

*Watson Studio Desktop 2.0 User Guide*



**Note**

Before using this information and the product that it supports, read the information in “Notices” on page 52.

---

# Tables of Contents

<b>Watson Studio Desktop documentation</b>	<b>1</b>
<b>Overview</b>	<b>1</b>
What's new	5
Watson Studio Desktop offerings	39
Limitations and known issues	40
Feature differences between deployments	44
Supported frameworks	45
Data source connections	45
Security	46
GDPR	47
Accessibility features in Watson Studio Desktop content and documentation	52
Notices	52
<b>Getting started</b>	<b>55</b>
<b>Installation and setup (Subscription)</b>	<b>56</b>
System requirements	56
Purchasing Watson Studio Desktop Subscription	57
Managing your subscription	58
Logging in and downloading updates	60
Installing Watson Studio Desktop	60
Installing on Windows	60
Network access or other permissions	61
Installing on macOS	62
Environments for notebooks	63
Managing add-ons	64
Locating files	64
Checking for updates	65
Uninstalling	66
<b>Installation and setup (Version 2.0)</b>	<b>66</b>
System requirements	67
Installing Watson Studio Desktop	68
Installing on Windows	68
Network access or other permissions	70
Installing on macOS	72
Environments for notebooks	73
Managing add-ons	74
Locating files	74
Uninstalling on Windows	75
Uninstalling on macOS	76
Updating from a previous version	76
Connecting to IBM Watson Machine Learning Server	77

<b>Installation and setup (Version 1.1)</b>	78
System requirements	78
Installing Watson Studio Desktop	79
Installing on Windows	80
Network access or other permissions	80
Installing on macOS	81
Environments for notebooks	82
Locating files	83
Uninstalling	84
Connecting to IBM Watson Machine Learning Server	84
<b>Projects</b>	85
Example projects	86
Deployment spaces	86
Exporting a project	88
<b>Adding data to a project</b>	89
Adding connections to projects	90
Connection types	90
IBM Cognos Analytics connection	92
IBM Planning Analytics connection	93
OData connection	94
SAP OData connection	95
<b>Refining data</b>	95
Adding data to Data Refinery	98
Specifying the format of your data	99
Validating your data	100
Visualizing your data	100
Managing Data Refinery flows	104
GUI operations	106
Interactive code templates	111
Tutorial: Shape raw data	116
<b>Creating SPSS Modeler flows</b>	124
Nodes palette	130
Import	130
Data Asset node	131
User Input node	133
Sim Gen node	134
Extension Import node	135
Record Operations	136
Select node	137
Sample node	138
Sort node	138
Balance node	139
Distinct node	139
Aggregate node	141
Merge node	142



Append node	142
Streaming Time Series node	142
SMOTE node	143
RFM Aggregate node	143
Space-Time-Boxes node	143
Extension Transform node	145
CPLEX Optimization node	145
Field Operations	147
Auto Data Prep node	148
Type node	149
Viewing and setting information about types	151
Measurement levels	151
Geospatial measurement sublevels	152
Converting continuous data	153
What is instantiation?	153
Data values	154
Setting options for values	155
Specifying values and labels for continuous data	156
Specifying values and labels for nominal and ordinal data	157
Specifying values for a flag	157
Specifying values for collection data	157
Specifying values for geospatial data	157
Defining missing values	158
Checking type values	158
Setting the field role	159
Setting field format options	159
Filter node	160
Derive node	160
Filler node	161
Reclassify node	162
Binning node	162
RFM Analysis node	163
Ensemble node	164
Partition node	164
Set to Flag node	165
Restructure node	165
Transpose node	166
Field Reorder node	166
History node	166
Time Intervals node	166
Anonymize node	167
Reproject node	167
Graphs	168
Charts node	169
Plot node	169
Multiplot node	172
Time Plot node	172
Distribution node	172
Histogram node	172
Collection node	173
Web node	173

Evaluation node	174
Modeling	178
Auto Classifier node	182
Auto Numeric node	183
Auto Cluster node	185
TCM node	185
Bayes Net node	186
C5.0 node	188
C&R Tree node	189
CHAID node	190
QUEST node	190
Tree-AS node	191
Random Trees node	191
Random Forest node	192
Decision List node	193
Time Series node	194
GenLin node	195
GLMM node	196
GLE node	197
Linear node	198
Linear-AS node	198
Regression node	198
LSVM node	199
Logistic node	199
Neural Net node	200
KNN node	200
Cox node	201
PCA/Factor node	201
SVM node	202
Feature Selection node	202
Discriminant node	203
SLRM node	203
Spatio-Temporal Prediction (STP) node	204
Spatio-Temporal Prediction (STP) model nugget	204
Association Rules node	204
Apriori node	205
CARMA node	206
Sequence node	206
Kohonen node	207
Anomaly node	208
K-Means node	209
TwoStep cluster node	209
TwoStep-AS cluster node	210
Isotonic-AS node	210
XGBoost-AS node	210
K-Means-AS node	211
XGBoost Tree node	211
XGBoost Linear node	211
Gaussian Mixture node	211
KDE node	212
One-Class SVM node	212

MultiLayerPerceptron-AS node	213
HDBSCAN node	213
Extension Model node	213
Extension model nugget	214
Text Analytics	215
About text mining	217
How extraction works	220
How categorization works	222
Language Identifier node	223
Text Link Analysis node	223
Expert options	224
TLA node output	225
Text Mining node	226
Text Mining model nuggets	227
Interactive workbench mode	228
The Categories and concepts view	229
The Clusters view	230
The Text Link Analysis view	231
The Resource Editor view	232
Setting options	233
Advanced linguistic settings	236
Advanced frequency settings	238
Generating a model nugget	239
Outputs	239
Table node	240
Matrix node	240
Analysis node	240
Data Audit node	241
Transform node	241
Statistics node	242
Means node	242
Report node	242
Set Globals node	243
Sim Fit node	243
KDE Simulation node	245
Extension Output node	245
Export	246
Data Asset Export node	246
Extension Export node	248
Working with your data	248
Missing data values	249
Handling missing values	250
Handling records with missing values	250
Handling fields with missing values	251
Handling records with system missing values	251
Functions available for missing values	253
Saving and running models on Watson Machine Learning Server	254
Flow scripting	255
Flow scripting example	256
Flow and SuperNode parameters	256
SQL optimization	257

How does SQL pushback work?	259
Tips for maximizing SQL pushback	260
Nodes supporting SQL pushback	261
CLEM expressions and operators supporting SQL pushback	264
Generating SQL from model nuggets	266
Disabling or caching nodes in a flow	266
Disabling nodes in a flow	266
Caching options for nodes	267
Importing an SPSS Modeler stream	268
Extension nodes	269
R scripts	270
Python for Spark scripts	270
Scripting with Python for Spark	270
Analytic Server Context	272
Data metadata	275
Date, time, timestamp	276
Exceptions	276
Examples	277
Setting properties for flows	280
Expression Builder	282
Selecting functions	283

## Notebooks

Creating notebooks	284
Parts of a notebook	285
Libraries and scripts	287
Importing scripts into a notebook	288
Coding and running notebooks	288
Loading and accessing data in a notebook	289
Adding data from a Planning Analytics connection	289
Using project-lib for Python	290
Markdown cheatsheet	292

## AutoAI (Version 2.0)

AutoAI tutorial	293
Building an AutoAI model	299
Saving an AutoAI generated notebook (Tech preview)	301
Using autoai-lib for Python (Tech preview)	302
Selecting an AutoAI model	310
AutoAI implementation details	313

## Deploying assets (Version 2.0)

Deployment spaces	318
Adding data to a space	320
Collaborator permissions for spaces	321
Deploying from a space	322
Creating an online deployment	322
Creating a batch deployment	325
Batch deployment details	327
Managing deployment jobs	332

Creating a Core ML deployment	333
Deploying using the Python client	333
Deploying functions	338
Deploying scripts with the Python client	343
<b>Deploying models (Version 1.1)</b>	347
Deployment spaces	348
Deploying from a space	349
Saving to a project using the Python client	351
Deploying from the Python client	354
Deploying functions	359
<b>SPSS Modeler tutorials</b>	364
Introduction to modeling	365
Building the flow	366
Browsing the model	369
Evaluating the model	373
Scoring records	375
Summary	376
Automated modeling for a flag target	376
Historical data	377
Building the flow	378
Generating and comparing models	379
Summary	384
Automated modeling for a continuous target	384
Training data	385
Building the flow	385
Comparing the models	387
Summary	388
Automated data preparation	389
Building the flow	389
Comparing the models	391
Drug treatment - exploratory graphs	392
Reading in text data	393
Creating a distribution chart	394
Creating a scatterplot	394
Creating a web chart	395
Creating advanced visualizations	396
Deriving a new field	397
Building a model	399
Browsing the model	399
Using an Analysis node	400
Screening predictors	401
Building the flow	401
Building the models	403
Reducing input data string length	403
Reclassifying the data	403
Classifying telecommunications customers	406
Building the flow	407
Browsing the model	411

Telecommunications churn	412
Building the flow	412
Browsing the model	416
Forecasting bandwidth utilization	417
Forecasting with the Time Series node	418
Creating the flow	419
Examining the data	420
Defining the dates	423
Defining the targets	424
Setting the time intervals	425
Creating the model	426
Examining the model	428
Summary	432
Forecasting catalog sales	432
Creating the flow	433
Examining the data	434
Exponential smoothing	435
ARIMA	440
Summary	443
Making offers to customers (self-learning)	443
Building the flow	444
Browsing the model	447
Retail sales promotion	450
Examining the data	450
Learning and testing	451
Condition monitoring	452
Examining the data	453
Data preparation	453
Learning	454
Testing	454
Hotel satisfaction example for Text Analytics	455
Text Mining node	457
Using the Interactive Workbench	459
Building and deploying the model	464
Text Link Analysis node	466
<b>Troubleshooting</b>	469
Viewing logs	469
Setting the log level	469
<b>Glossary</b>	471

# Watson Studio Desktop documentation

---

Welcome to the documentation for Watson Studio Desktop, where you can find information about using the Desktop deployment environment of Watson Studio. This documentation contains information about Watson Studio Desktop 1.1, Watson Studio Desktop 2.0 and Watson Studio Desktop Subscription. All topics apply to all offerings, unless otherwise specified.

## Getting started

- [Product overview](#)
- [Getting started with Watson Studio Desktop](#)
- [Installation \(Subscription\)](#)
- [Installation \(Version 2.0\)](#)
- [Installation \(Version 1.1\)](#)
- [What's new?](#)
- [Example projects](#)

## Common tasks

- [Working with projects](#)
- [Adding data to a project](#)
- [Refining data](#)
- [Creating SPSS Modeler flows](#)
- [Importing SPSS Modeler streams](#)
- [Developing applications with notebooks](#)
- [Building models with AutoAI](#)

## Troubleshooting and support

- [Troubleshooting Watson Studio Desktop](#)
- [🔗 Watson Studio Desktop support](#)
- [🔗 Stack overflow](#)

## Other deployment environments and components

- [🔗 Cloud Pak for Data as a Service](#)
- [🔗 Cloud Pak for Data](#)
- [Deployment environments feature matrix](#)
- [🔗 Watson Machine Learning Server](#)

## More information

- [🔗 Watson Studio portal](#)
- [🔗 Watson Studio community](#)
- [🔗 Pricing](#)
- [🔗 Key features and use cases](#)
- [🔗 Cognitive Class](#)

© Copyright IBM Corporation 2016, 2020

# Overview of Watson Studio Desktop

---

IBM Watson Studio Desktop is a desktop client tool for solving your business problems by analyzing data with artificial intelligence. With Watson Studio Desktop, you can prepare data and build models on your desktop with visual drag and drop tools.

## Projects

You organize your resources for data analysis tasks in projects. Each [project](#) has its own directory on your computer. You can choose a standard project or to import a project that was previously exported from Watson Studio Desktop.

You can add these resources to a project:

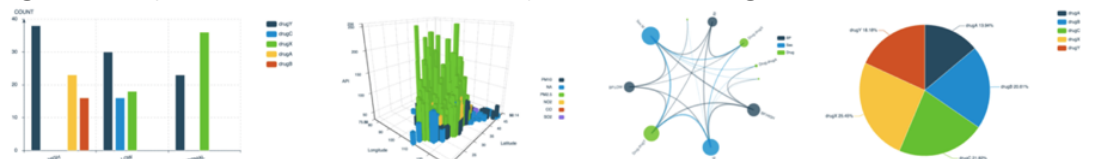
### Data assets

You can [load data](#) into a project, link to data files on your computer, or create a connection to access data from a remote data source.

### Data Refinery flows

Click Refine on a data asset to create a Data Refinery flow to [refine your data](#). The data set opens with three tabs. On the Data tab, you can view the data and apply operations to cleanse, shape, and enrich your data. On the Profile tab, you can see the frequency and statistics for each column of data. Click the Visualizations tab to explore data from different perspectives, and identify patterns, connections, and relationships within that data.

Figure 1. Sample visualizations available when you refine data or right-click a node in a flow

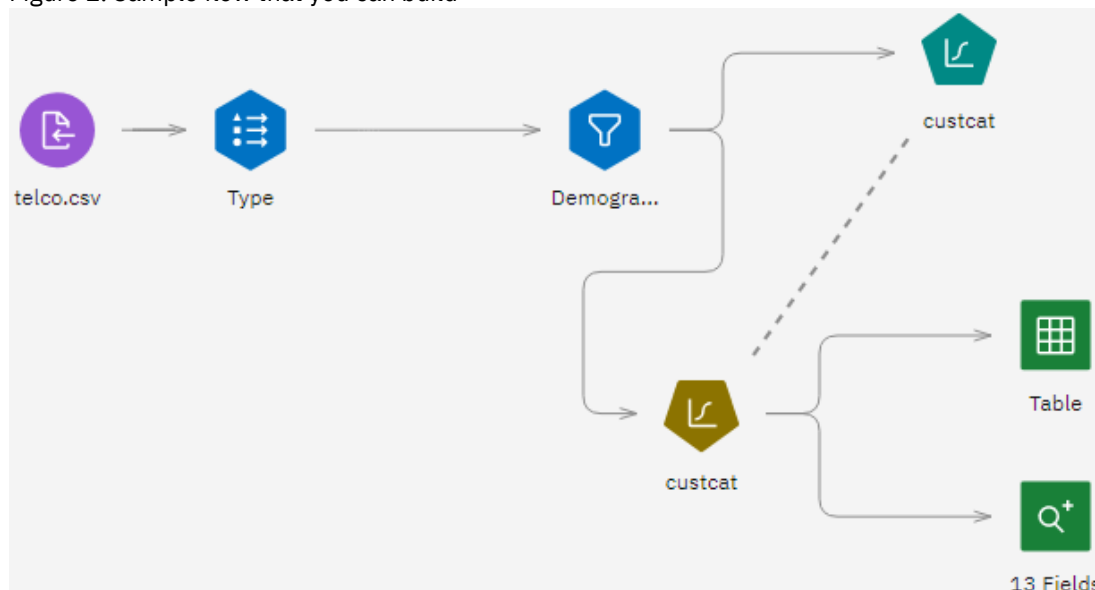


Run the steps in the Data Refinery flow to create a new data asset in your project that contains the refined data.

### SPSS Modeler flows

To analyze your data, you can build, train, and test models with the [SPSS Modeler flows tool](#).

Figure 2. Sample flow that you can build

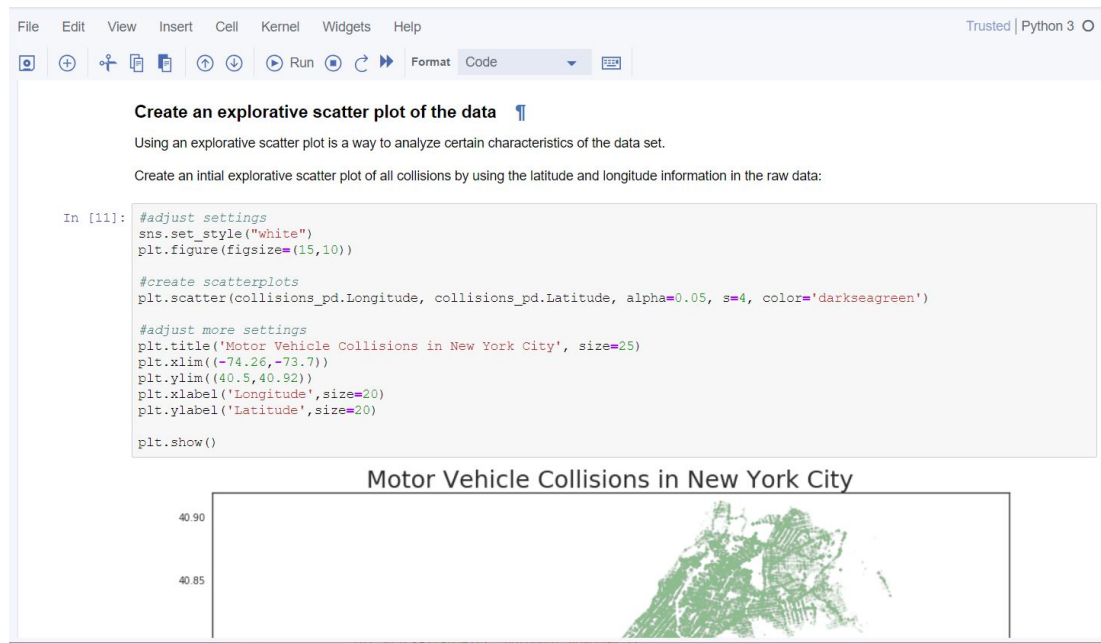


### Notebooks

To understand your data and form business insights that you share with collaborators, you can use notebooks. [Notebooks](#) give you a lot of flexibility combining code, statistics, visualizations, and explanatory text.

Figure 3. Sample notebook





### AutoAI Experiments

If you are connected to a Watson Machine Learning Server, you can automatically build and train machine learning models using your structured data. AutoAI analyzes the data, chooses the model types, applies algorithms, and optimizes a set of model candidate pipelines designed to solve classification or regression problems. You can review the pipelines, save them as models, deploy in a deployment space and score them to generate predictions. See [AutoAI overview](#).

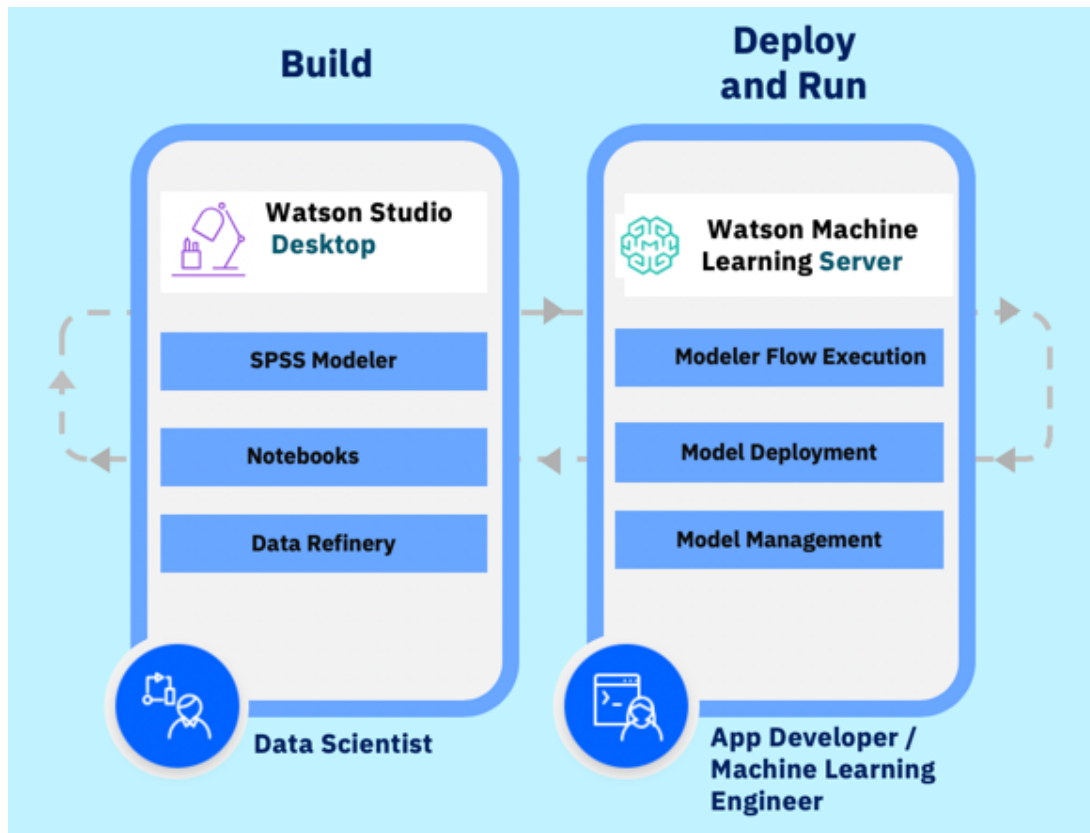
### Deployment spaces

If you are connected to a Watson Machine Learning Server, you can associate deployment spaces with your project. A deployment space is where you configure and deploy your models. After you finish developing your models in your project, you create a deployment space for that project and move your models into it. You can easily move assets between projects and deployment spaces. See [Deployment spaces](#).

## Watson Machine Learning Server

IBM Watson Machine Learning Server is a separate offering that integrates with Watson Studio Desktop 1.1 or Watson Studio Desktop 2.0. When you connect to a Watson Machine Learning Server, you can use its processing power to run model flows. After you create and train an SPSS model, you promote the model from Watson Studio Desktop to a deployment space on the Watson Machine Learning Server, where you can configure, monitor, and deploy the model. You deploy and manage Python notebooks programmatically.

Figure 4. Watson Studio Desktop and Watson Machine Learning Server



## Community

To ask questions and get answers from other Watson Studio users and Watson Studio experts, check out the [Watson Studio questions page in Stack Overflow](#).

You can also access the [Watson Studio Community](#), which contains resources to help you learn about data science:

- Read articles from many sources to keep current with data science trends.
- Read tutorials for multiple skill levels to learn how to do specific data science tasks. Some of the tutorials do not apply to Watson Studio Desktop, which has a subset of features and asset types.

## Ready to get started?

Download and install Watson Studio Desktop. See [Installing Watson Studio Desktop \(Subscription\)](#), or [Installing Watson Studio Desktop 1.1](#) or [Installing Watson Studio Desktop 2.0](#).

- **What's new in Watson Studio Desktop**  
Check out what's new for Watson Studio Desktop! The available features correspond to the latest update available for your offering.
- **Watson Studio Desktop offerings**  
IBM Watson Studio Desktop is available as two offerings. Both offerings include the major SPSS Modeler features, Python notebooks, and Data Refinery.
- **Limitations and known issues**  
Use the following information to help troubleshoot issues with IBM Watson Studio Desktop.
- **Feature differences between deployments**
- **Supported frameworks**
- **Data source connections**
- **Security**
- **Accessibility features in Watson Studio Desktop content and documentation**  
IBM is committed to accessibility. Accessibility features that follow compliance guidelines are included in Watson Studio Desktop content and documentation to benefit users with disabilities. Parts of the user interface

of Watson Studio Desktop are accessible, but not entirely. Only documentation is compliant, with a subset of parts of the overall product.

- [Notices](#)

## What's new in Watson Studio Desktop

---

Check out what's new for Watson Studio Desktop! The available features correspond to the latest update available for your offering.

### 28 September 2020 (Version 2.0)

---

- [Updates to installation](#)
- [What's new in Watson Machine Learning Server](#)
- [What's new in SPSS Modeler](#)
- [New visualization charts for Data Refinery and SPSS Modeler](#)
- [New operations and enhancements for Data Refinery](#)
- [Load data from local files to your notebook](#)
- [View Watson Studio Desktop in your own language](#)

#### Updates to installation

##### New icon

You'll see a new icon for Watson Studio Desktop when you install the application.

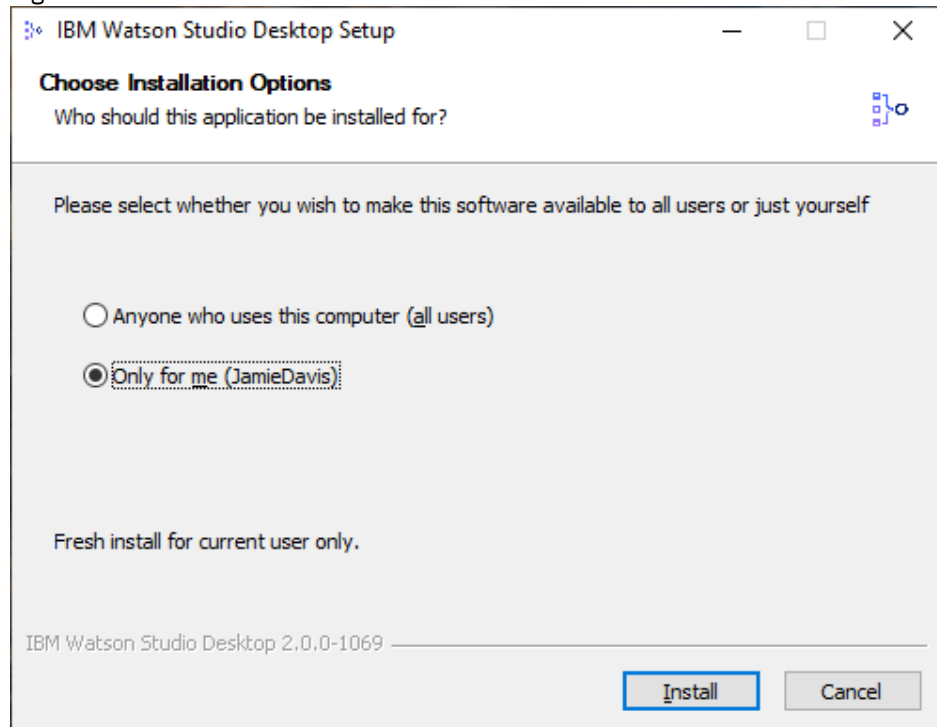
Figure 1. Watson Studio Desktop icon



##### Choices for Windows installation

You now have a choice of a *per-user* installation or a *per-machine* (*all users*) installation. The per-user installation is for a single user using Watson Studio Desktop on the computer and is the default installation. The per-machine installation is for multiple users.

Figure 2. Windows installation choices

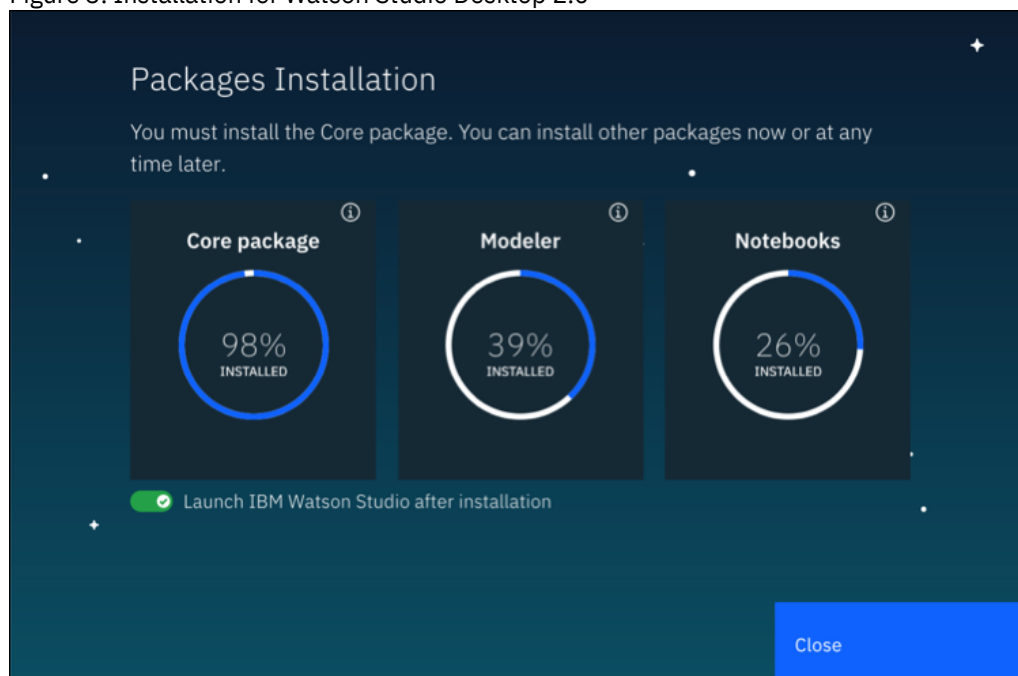


For Windows installation instructions, see [Installing on Windows \(Version 2.0\)](#).

#### Faster and more efficient installation

Watson Studio Desktop 2.0 has a new user interface to install the application. You can choose which add-ons (Modeler or Notebooks) to install and when.

Figure 3. Installation for Watson Studio Desktop 2.0



Choose the steps for your operating system: [Installing Watson Studio Desktop 2.0](#).

#### Only one instance of Watson Studio can be installed on a computer

You can install only one instance of Watson Studio Desktop or Watson Studio Desktop Subscription on a computer.

#### Minimum RAM increased from 8 GB to 16 GB

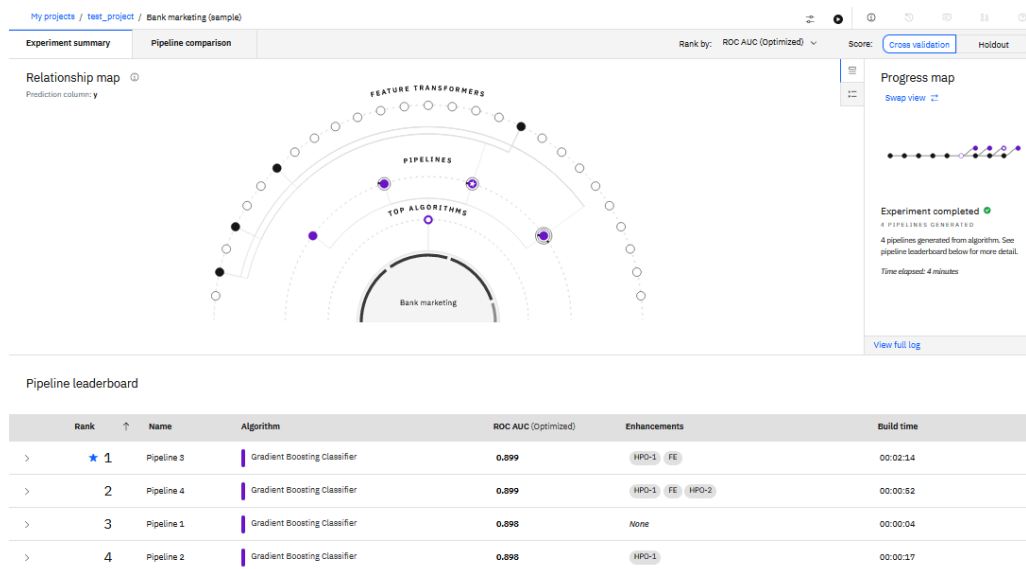
The minimum RAM for both Windows and macOS is now 16 GB. For the updated list of all system requirements, see [System requirements \(Version 2.0\)](#).

#### What's new in Watson Machine Learning Server

##### Build machine learning models with AutoAI

The AutoAI graphical tool in Watson Studio automatically analyzes your structured data and generates candidate model pipelines customized for your predictive modeling problem. These model pipelines are created iteratively as AutoAI analyzes your data set and discovers data transformations, algorithms, and parameter settings that work best for your problem setting.

Figure 4. AutoAI trained pipelines



This feature requires Watson Machine Learning Server. See [AutoAI](#) for more information.

### Save an AutoAI model as a notebook

After training an AutoAI experiment, you can save a pipeline to your project as an automatically generated notebook. Use the notebook to review the data transformations applied to generate the model. You can also run the model from the notebook, and deploy the model to score it and generate predictions. This feature requires Watson Machine Learning Server. See [AutoAI notebook](#) for more information.

### Schedule batch deployment jobs

Schedule batch deployment jobs on Watson Machine Learning Server. From a deployment space, specify input data and write predictions to an output file. Run the deployment job and view the resulting batch predictions. See [Managing deployment jobs](#) for more information.

### Watson Machine Learning Server supports more input data assets

Promote or add data sources to a deployment space to use with batch deployment jobs. Data can be:

- A data file such as a .csv file
- A connection to data that resides in a repository such as a Db2 database
- Connected data that resides in a storage bucket, such as a data file in a Cloud Object Storage bucket.

### Import or export a Watson Machine Learning Server deployment space

With Watson Machine Learning Server, you can export the contents of a deployment space to a file. You can also create a new space by importing a space from a file. This provides an efficient way to archive a space, use a space as a template, or share a space. See [Deployment spaces](#) for more information.

### Deploy Python scripts on Watson Machine Learning Server

Deploy Python scripts on Watson Machine Learning Server. Scripts are not a supported project asset, so you must store and deploy them using the Watson Machine Learning Python client. See [Deploying scripts](#) for more information.

### Watson Machine Learning Python client installed by default

The latest version of the Watson Machine Learning Python client is installed by default when you create a notebook in Watson Studio Desktop. You no longer need to install the client library. See [Deploying using the Python client](#) for more information.

## Export Watson Machine Learning Server assets from a project

With Watson Studio Desktop, you can export the contents of a project to a file. The exportable assets now include your machine learning models and Python functions saved in notebooks. You can also import machine learning assets when you create a new project from a saved project file. This provides an efficient way to archive a project, use a project as a template, or share a project.

## What's new in SPSS Modeler

### New SPSS Modeler nodes

- With the new CPLEX Optimization node in SPSS Modeler, you can use complex mathematical (CPLEX) based optimization via an Optimization Programming Language (OPL) model file. See [CPLEX Optimization node](#).

Figure 5. CPLEX Optimization node



- The new Kernel Density Estimation (KDE) Simulation node uses the Ball Tree or KD Tree algorithms for efficient queries, and walks the line between unsupervised learning, feature engineering, and data modeling. See [KDE Simulation node](#).

Figure 6. KDE Simulation node



- The Data Asset Export node has been redesigned. Use the node to write to remote data sources using connections, write to a data file on your local computer, or write data to your project. See [Data Asset Export node](#).

### Database functions supported in SPSS Modeler desktop streams

You can now run an SPSS Modeler desktop stream file (.str) that contains database functions. But they aren't yet available in the Expression Builder user interface.

### Write to any directory

Previously, you could only write to your projects directory. Now you can write to any directory on your system that you have access to.

### New Restart session button

In some cases, a task may fail to complete in SPSS Modeler and you'll be prompted to continue running or restart your session. You can also force your session to restart at any time by clicking the Restart session button (↺) in the right-hand Information panel.

### Deploy Text Analytics models

You can now deploy Text Analytics models to a Watson Machine Learning Server as you can with other model types. See [Deploying assets](#) for more information.

### Profile right-click option

The Profile right-click option for SPSS Modeler nodes has been removed. To launch the chart builder and create advanced visualizations, you can use the Charts node.

#### New visualization charts for Data Refinery and SPSS Modeler

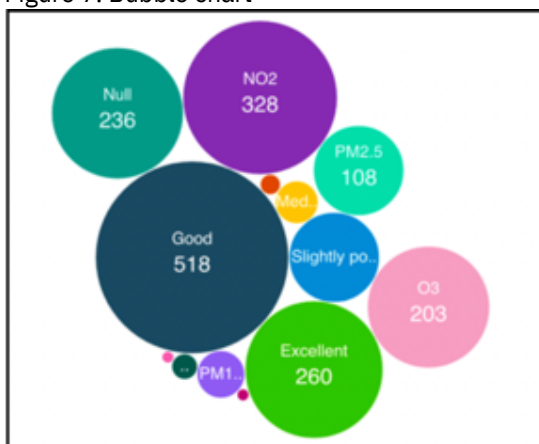
To access the charts in Data Refinery, click the Visualizations tab and then select the columns to visualize. The chart automatically updates as you refine the data.

To access the charts in SPSS Modeler, use a Charts node. The Charts node is available under the Graphs section on the node palette. Double-click the Charts node to open the properties pane. Then click Launch Chart Builder to open the chart builder and create one or more chart definitions to associate with the node.

For the full list of available charts, see [Visualizing your data](#).

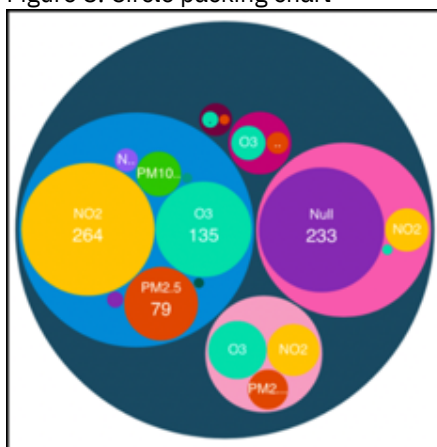
- Bubble charts display each category in the groups as a bubble.

Figure 7. Bubble chart



- Circle packing charts display hierarchical data as a set of nested areas.

Figure 8. Circle packing chart



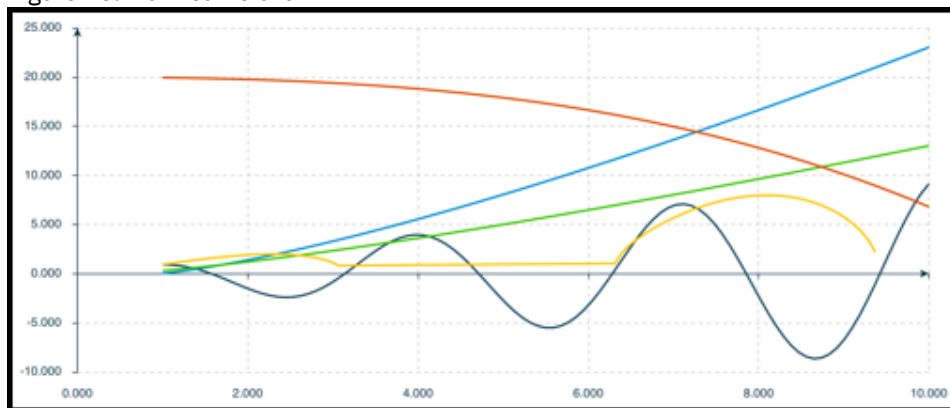
- Evaluation charts are combination charts that measure the quality of a binary classifier. You need three columns for input: actual (target) value, predict value, and confidence (0 or 1). Move the slider in the Cutoff chart to dynamically update the other charts. The ROC and other charts are standard measurements of the classifier.

Figure 9. Evaluation chart



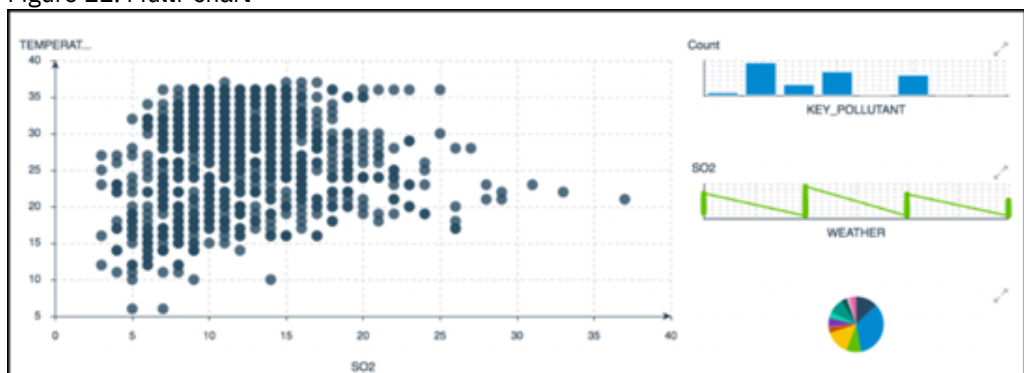
- Math curve charts display a group of curves based on equations that you enter. You do not use a data set with this chart. Instead, you use it to compare the results with the data set in another chart, like the scatter plot chart.

Figure 10. Math curve chart



- Multi-charts display up to four combinations of Bar, Line, Pie, and Scatter plot charts. You can show the same kind of chart more than once with different data. For example, two pie charts with data from different columns.

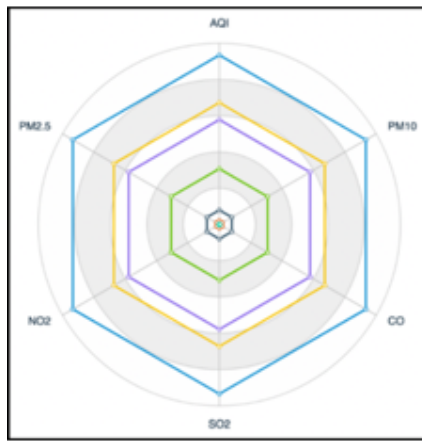
Figure 11. Multi-chart



- Radar charts integrate three or more quantitative variables that are represented on axes (radii) into a single radial figure. Data is plotted on each axis and joined to adjacent axes by connecting lines. Radar charts are useful to show correlations and compare categorized data.

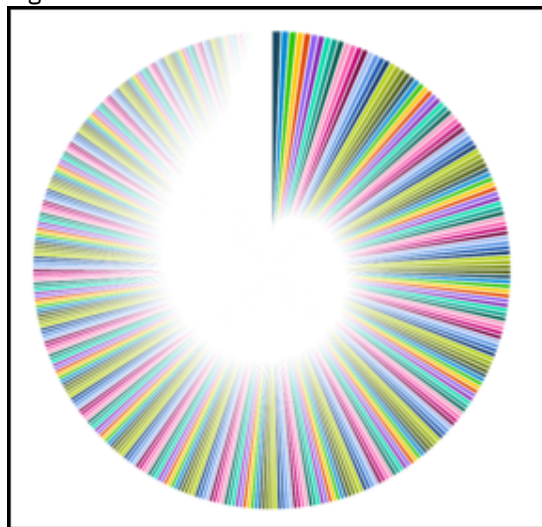
Figure 12. Radar chart





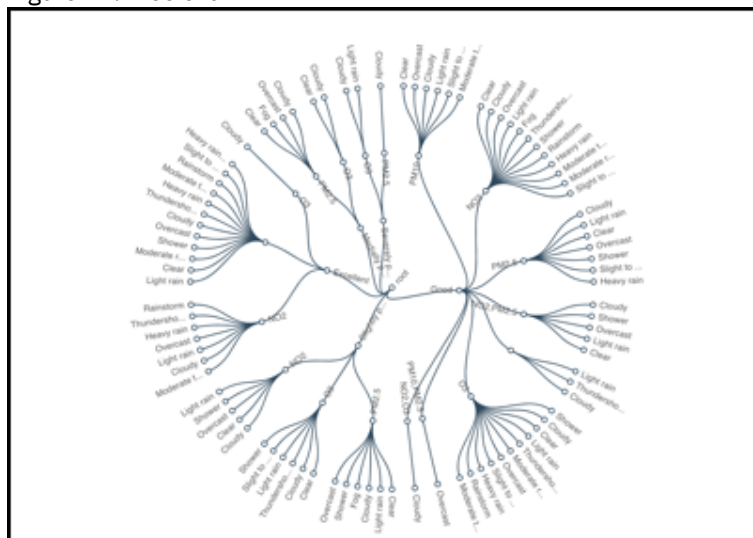
- Sunburst charts display different depths of hierarchical groups. The Sunburst chart was formerly an option in the Treemap chart.

Figure 13. Sunburst chart



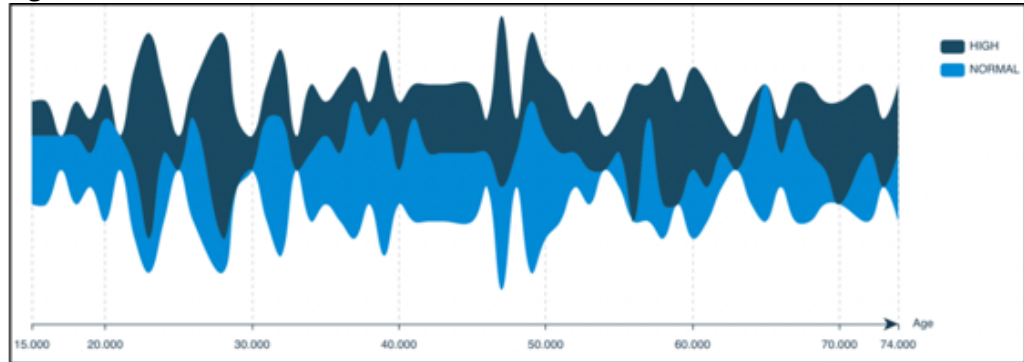
- Tree charts represent a hierarchy in a tree-like structure. The Tree chart consists of a root node, line connections called branches that represent the relationships and connections between the members, and leaf nodes that do not have child nodes. The Tree chart was formerly an option in the Treemap chart.

Figure 14. Tree chart



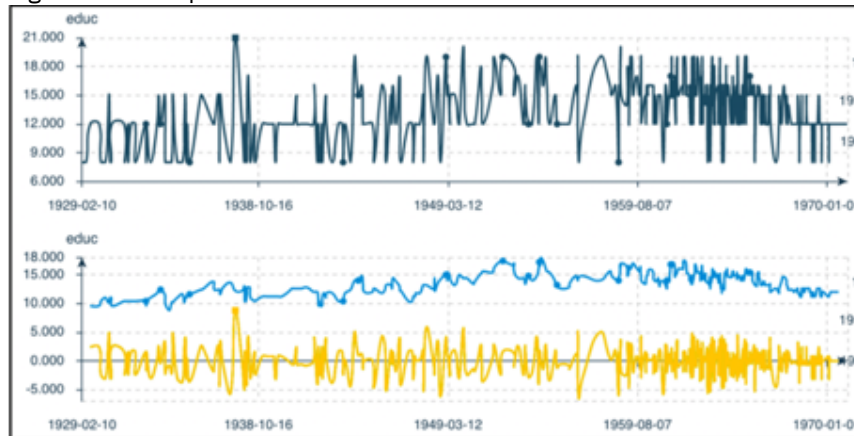
- Theme river charts use a specialized flow graph that shows changes over time.

Figure 15. Theme river chart



- Time plot charts illustrate data points at successive intervals of time.

Figure 16. Time plot chart



## New operations and enhancements for Data Refinery

### New Data Refinery GUI operations

These operations are in the ORGANIZE category:

#### Join operation

The **Join** operation can join two data sets in a variety of ways. You can perform a full join, inner join, left join, right join, semi join, or anti join. You can also select the columns you want to see in the result set, and if there are same-named columns between the two data sets, you can specify unique suffixes to differentiate them.

Figure 17. Data Refinery Join operation

[<](#) Join

Combine data from two data sets based on a comparison of the values in specified key columns.

Left join
 [v](#)

Returns all rows in the original data set and returns only matching rows in the joining data set. Returns one row in the original data set for each matching row in the joining data set.

The default suffix for each data set will be used to differentiate any duplicate column names in the resulting data set.

Source
 

mydata.csv
 

\*Suffix
 

\_x

Data set to join
 

+ Add data set

\*Suffix
 

\_y

#### Union operation

Use the **Union** operation to combine the rows from two data sets that share the same schema.

Figure 18. Data Refinery Union operation



The screenshot displays the Data Refinery interface with two aggregation operations.   
**AGGREGATION 1** is configured with the column `UniqueCarrier` and the operator `Count unique values`. The resulting output is `unique-airlines`.   
**AGGREGATION 2** is configured with the column `ArrDelay` and the operator `Mean`.   
 Red boxes highlight the column selection dropdowns in both aggregation configurations.

The **Aggregate** operation is in the ORGANIZE category.

#### Automatically detect and convert date and timestamp data types

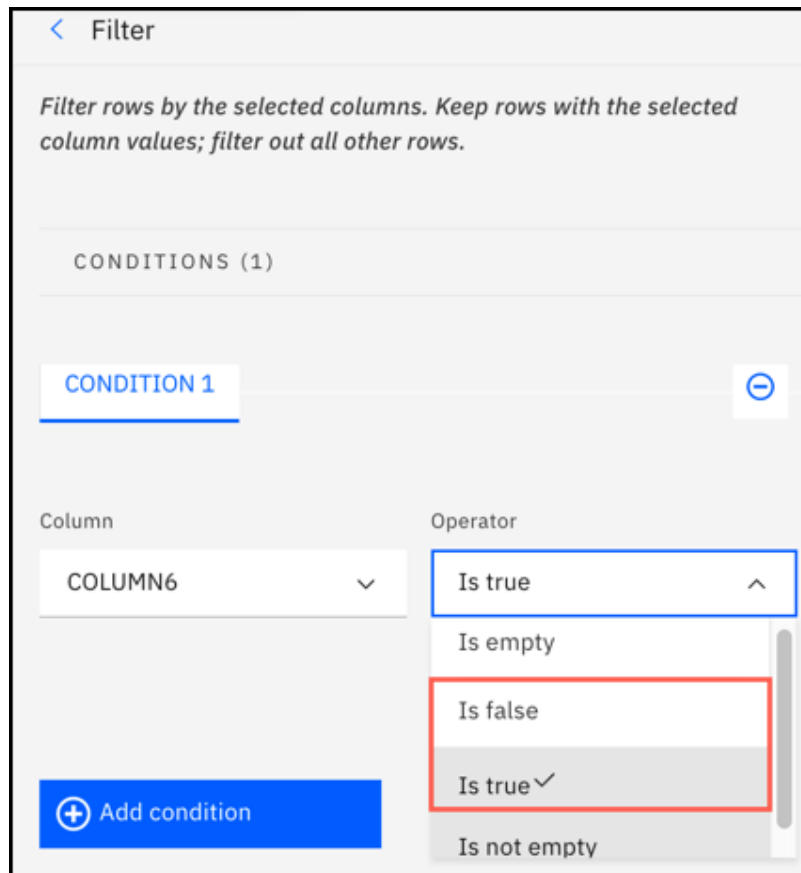
When you open a file in Data Refinery, the **Convert column type** GUI operation is automatically applied as the first step if it detects any non-string data types in the data. Now date and timestamp data are detected and are automatically converted to inferred data types. You can change the automatic conversion for selected columns or undo the step. For information about the supported inferred date and timestamp formats see [Convert column type in GUI operations in Data Refinery](#), under the FREQUENTLY USED category.

#### Filter values in a Boolean column

You can now use these operators in the **Filter** GUI operation to filter Boolean (logical) data:

- `Is false`
- `Is true`

Figure 20. Data Refinery Filter operation with Boolean selection



The **Filter** operation is in the FREQUENTLY USED category.

In addition, a new template for filtering by Boolean values has been added to the **filter** coding operation.

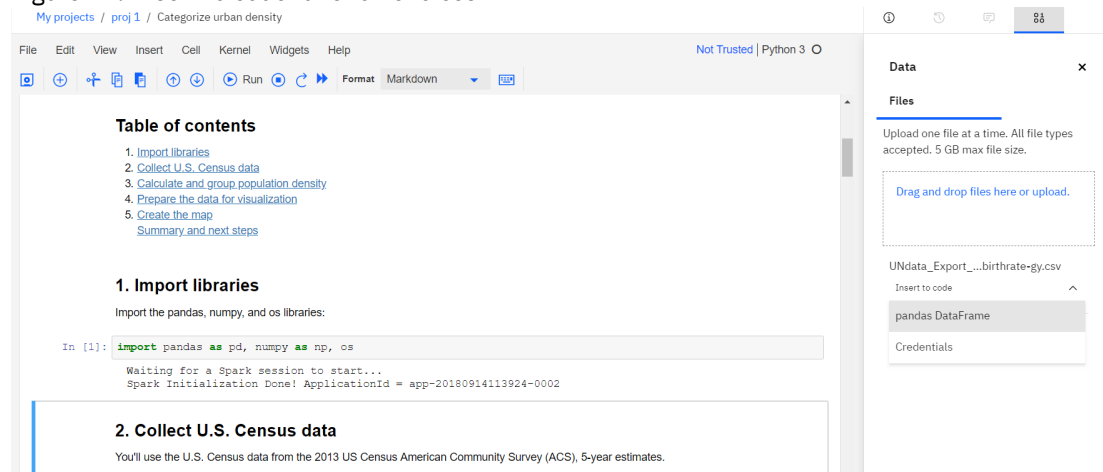
```
filter(`<column>`== <logical>)
```

For more information about the **filter** templates, see [Interactive code templates in Data Refinery](#).

Load data from local files to your notebook


You can now load a file directly from within a notebook and access the data by using the *Insert to code* function. The generated code to access the data serves as a quick start to begin working with the data in a file.

Figure 21. Insert to code function choices



See [Loading and accessing data in a notebook](#) for more information.

View Watson Studio Desktop in your own language

To change the language, click Settings (  ) on the left side of the window. Alternatively, you can change the language from the menu: View > Language.

You can view the Watson Studio Desktop user interface in the following languages:

- Simplified Chinese
- Traditional Chinese
- English
- French
- German
- Italian
- Japanese
- Brazilian Portuguese
- Russian
- Spanish

In addition, the following features and changes from the **29 May 2020 (Subscription)** release are included:

- [Save your example projects if you modified them](#)
- [New connections](#)
- [Name changes for four connections](#)
- [SAV file support](#)
- [SPSS Modeler Extension nodes](#)
- [SPSS Modeler flow properties](#)
- [SPSS Modeler node tooltips](#)
- [SPSS Modeler Type node icons](#)
- [New options for the SPSS Modeler Auto Classifier and Auto Numeric nodes](#)
- [New SQL icon](#)
- [Maximum file size restriction removed for Data Refinery](#)
- [Specify format options for your data in Data Refinery](#)
- [Data Refinery automatically detects and converts data types](#)
- [New SPSS Modeler tutorials](#)

## 29 May 2020 (Subscription)

---

New name for the Subscription offering

The Subscription offering name is changed from "IBM Watson Studio" to "IBM Watson Studio Desktop Subscription."

New icon

Watson Studio Desktop Subscription has a new icon.

Figure 22. Watson Studio Desktop Subscription icon



Save your example projects if you modified them

The Example Project installed with the product has been replaced to fix an issue with one of the flows. Note that the flows included are intended to be for demonstration purposes only. They'll be replaced periodically when the product is updated. If you modified any of the examples and want to preserve your work, you should export and import the resources as desired to avoid them being replaced. For details about the example projects, see [Example projects](#).

New connections

Watson Studio Desktop now supports connections to the following data sources:

- [IBM Cognos Analytics](#)
- [IBM Planning Analytics](#) (formerly known as "IBM TM1")
- [OData](#)
- [SAP OData](#)

#### Name changes for four connections

These connections have new names:

- "BigInsights HDFS" is renamed to "Analytics Engine HDFS"
- "Compose for PostgreSQL" is renamed to "Databases for PostgreSQL"
- "Hortonworks HDFS" is renamed to "Apache HDFS"
- "PureData System for Analytics" is renamed to "Netezza (PureData System for Analytics)"

Your previous settings for the connections remain the same. Only the connection names have changed.

#### SAV file support

You can now import or export SPSS Statistics.sav data files in SPSS Modeler. You can refine .sav files in Data Refinery.

#### SPSS Modeler Extension nodes

To complement SPSS Modeler and its data mining abilities, several Extension nodes are now available to enable expert users to input their own R scripts or Python for Spark scripts to carry out data processing, model building, and model scoring. See [Extension nodes](#).

#### SPSS Modeler flow properties

You can set flow properties. See [Setting properties for flows](#).

#### SPSS Modeler node tooltips

New tooltips are available in the nodes palette. You can hover over any node to see a helpful description before adding it to your flow canvas.

#### SPSS Modeler Type node icons

New icons in the Type node properties quickly indicate the data type of each field, such as string, date, double integer, or hashtag.

Figure 23. New Type node icons



#### New options for the SPSS Modeler Auto Classifier and Auto Numeric nodes

New cross-validation options are available for testing the effectiveness of machine learning models or evaluating a model if you have limited data. See [Auto Classifier node](#) or [Auto Numeric node](#).

Figure 24. New cross-validation options



☒ Cross-validate

Number of folds ⓘ

5

☐ Repeatable Cross Validation partition assignment

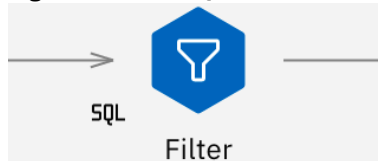
Random seed [Generate](#)

1234567

#### New SQL icon

When running a flow, nodes that push back to your database used to be highlighted with a small purple icon beside the node. Now a new small SQL icon is used instead.

Figure 25. New SQL icon



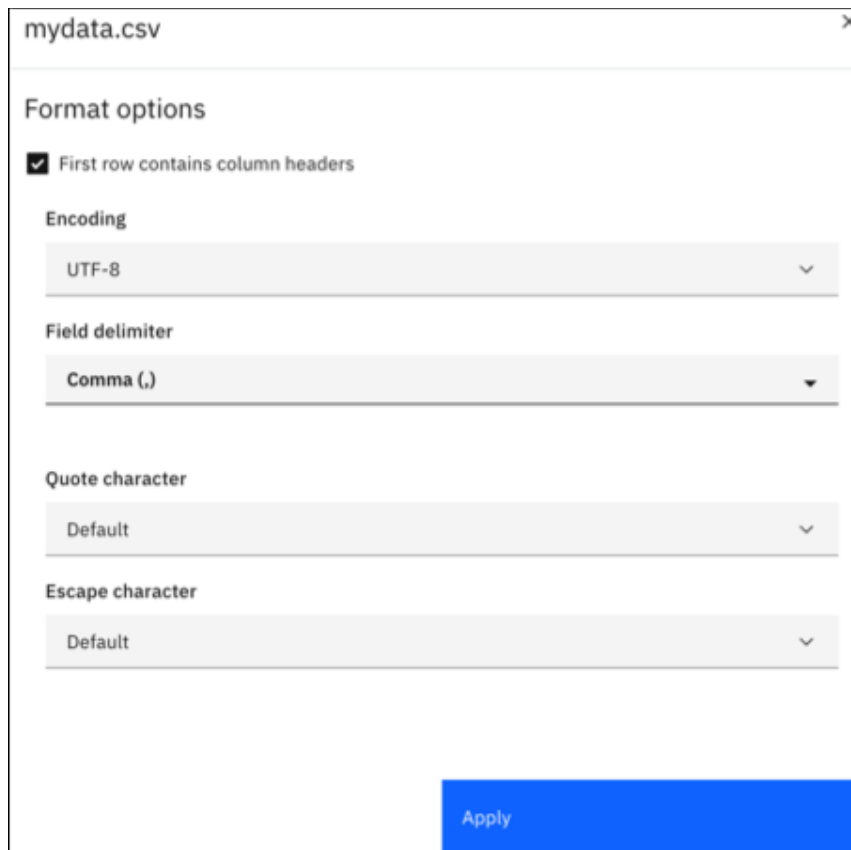
#### Maximum file size restriction removed for Data Refinery


Previously, the maximum file size for Data Refinery depended on the amount of RAM in your computer.

#### Specify format options for your data in Data Refinery

You can now modify the format options in Data Refinery such as the encoding, the field delimiter, and the quote and escape characters. You can also specify if the first row contains the column headers. Use this feature to fix problems when the data does not display in a tabular form.

Figure 26. Format options in Data Refinery

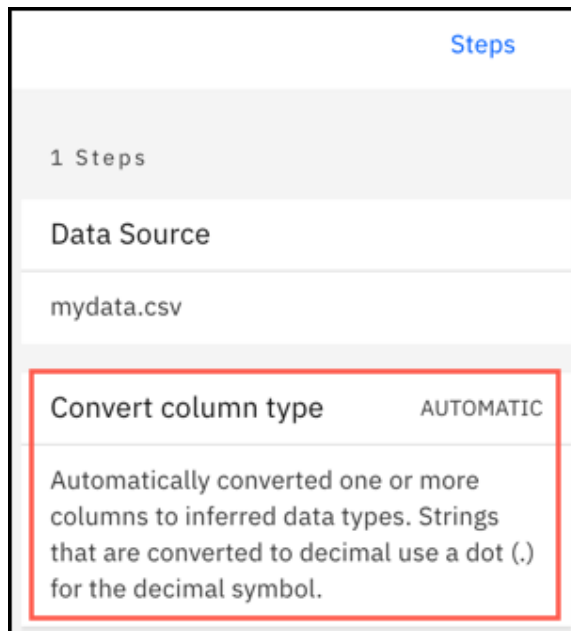
The image shows a dialog box titled 'mydata.csv' with a close button in the top right corner. Inside the dialog, under the heading 'Format options', there is a checked checkbox labeled 'First row contains column headers'. Below this are four dropdown menus: 'Encoding' set to 'UTF-8', 'Field delimiter' set to 'Comma (,)', 'Quote character' set to 'Default', and 'Escape character' set to 'Default'. Each dropdown menu has a small downward arrow on its right side. At the bottom right of the dialog is a blue button labeled 'Apply'.

To access the format options in Data Refinery, go to the Data tab, scroll down to the SOURCE FILE information at the bottom of the page, and click the "Specify data format" icon . For more information, see [Specifying the format of your data in Data Refinery](#).

#### Data Refinery automatically detects and converts data types

Previously, you had to manually apply the Convert column type GUI operation to detect and convert the data types. Now the Convert column type GUI operation is automatically applied as the first step in the Data Refinery flow. The operation automatically detects and converts the data types to inferred data types (for example, to Integer, Boolean, etc.) as needed. This enhancement will save you a lot of time, particularly if the data has many columns. It is easy to undo the automatic conversion or to edit the operation for selected columns.

Figure 27. Automatic detection and conversion of data type



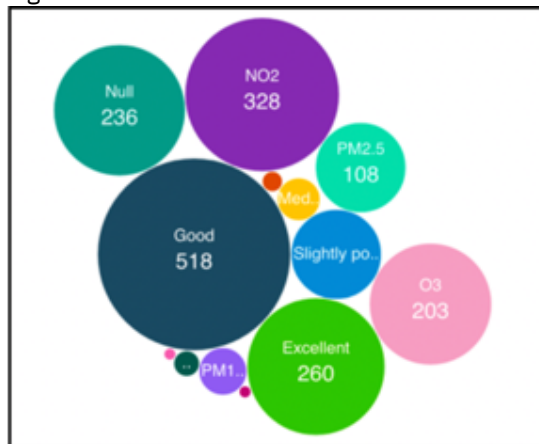
For more information, see Convert column type in [GUI operations in Data Refinery](#), under the FREQUENTLY USED category.

#### New visualization charts in Data Refinery

Data Refinery introduces six new charts. To access the charts, click the Visualizations tab in Data Refinery, and then select the columns to visualize. The chart automatically updates as you refine the data.

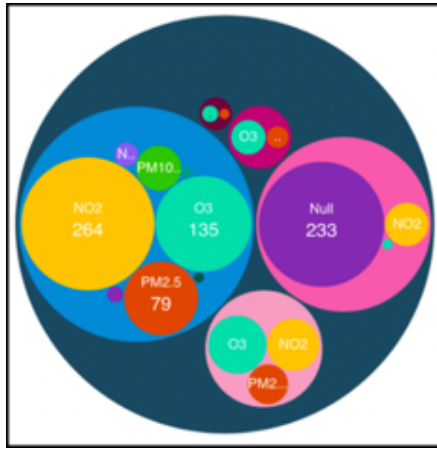
- Bubble charts display each category in the groups as a bubble.

Figure 28. Bubble chart



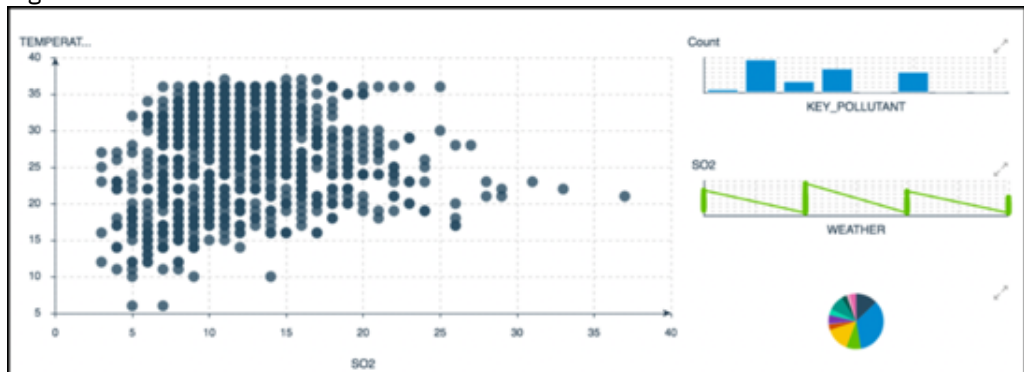
- Circle packing charts display hierarchical data as a set of nested areas.

Figure 29. Circle packing chart



- Multi-charts display up to four combinations of Bar, Line, Pie, and Scatter plot charts. You can show the same kind of chart more than once with different data. For example, two pie charts with data from different columns.

Figure 30. Multi-chart



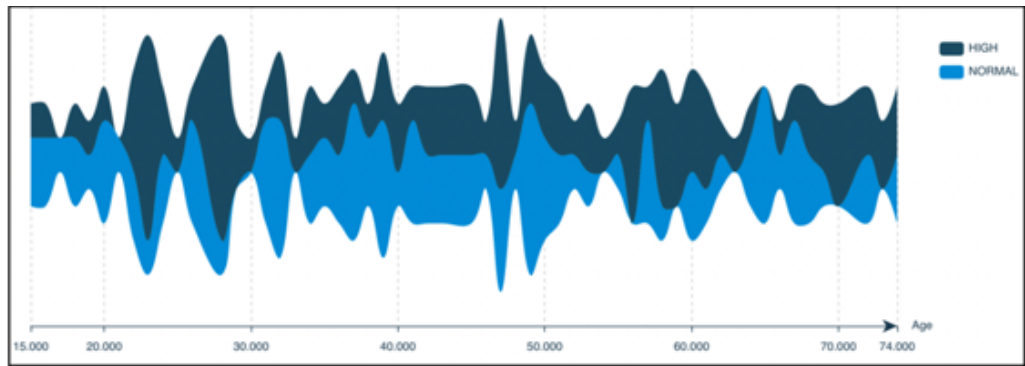
- Radar charts integrate three or more quantitative variables that are represented on axes (radii) into a single radial figure. Data is plotted on each axis and joined to adjacent axes by connecting lines. Radar charts are useful to show correlations and compare categorized data.

Figure 31. Radar chart



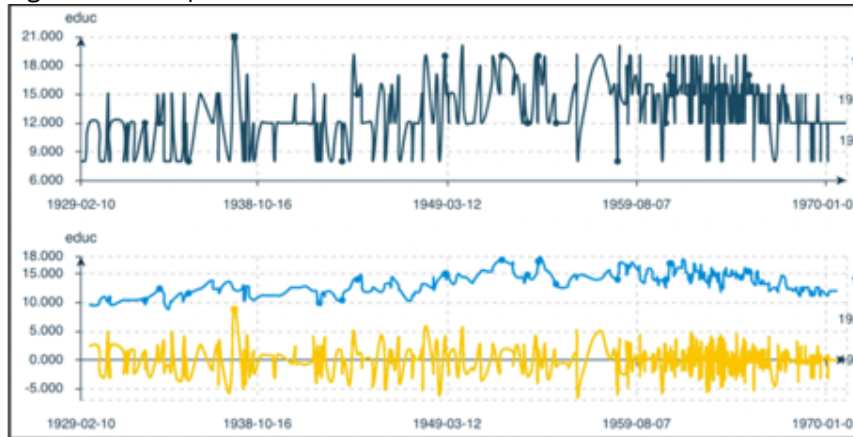
- Theme river charts use a specialized flow graph that shows changes over time.

Figure 32. Theme river chart



- Time plot charts illustrate data points at successive intervals of time.

Figure 33. Time plot chart

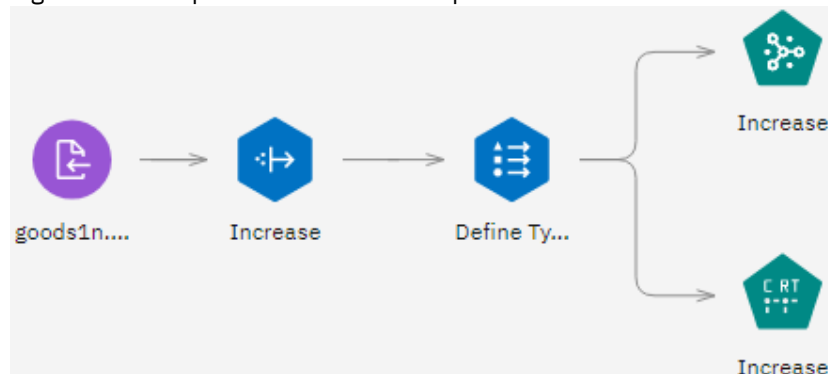


#### New SPSS Modeler tutorials

The following two new tutorials are available to accompany the thirteenth and fourteenth flows in the example project that's installed with the product. See [SPSS Modeler tutorials](#).

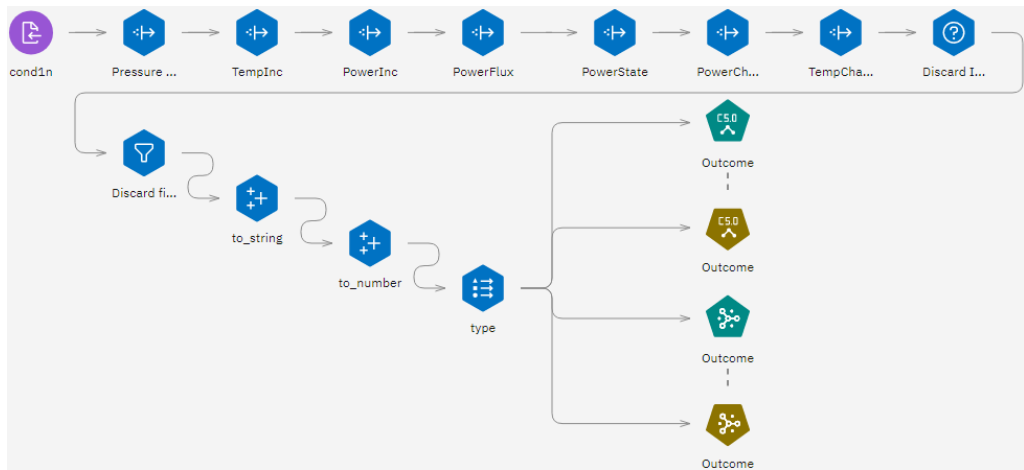
- [Retail sales promotion](#)

Figure 34. Example flow for retail sales promotion



- [Condition monitoring](#)

Figure 35. Example flow for condition monitoring



In addition, the following features and changes from the **31 January 2020 (Version 1.1)** release are included:

- [Support for Google BigQuery SQL pushback for Watson Studio Desktop](#)
- [New right-click options for nodes in SPSS Modeler flows](#)
- [New SPSS Modeler documentation](#)

## 31 January 2020 (Version 1.1)

What's new in Watson Machine Learning Server

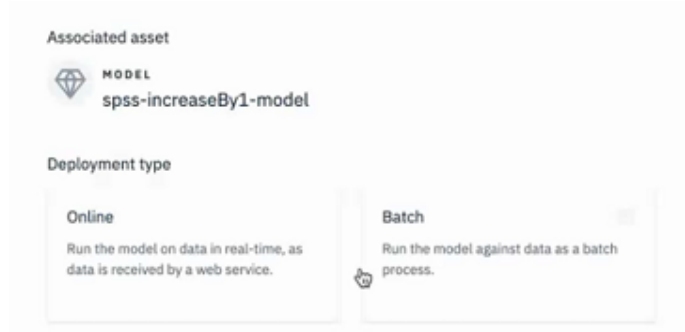
### Saving models

Previously, you saved models to a deployment space. Now you save models to your projects where you can then access your model and create deployments from the Models section under Assets. (You can still save models to a space with the Python client.) For more information, see [Saving and running models on Watson Machine Learning Server](#), [Deploying models](#), and [Deployment spaces](#).

### Improved support for deployment types

Previously, you had to create batch and virtual deployments programmatically. You can now create them from a deployment space. Additionally, you can promote the data assets required for a batch deployment from a project to a deployment space, or upload them directly to the space.

Figure 36. Online and Batch deployment types

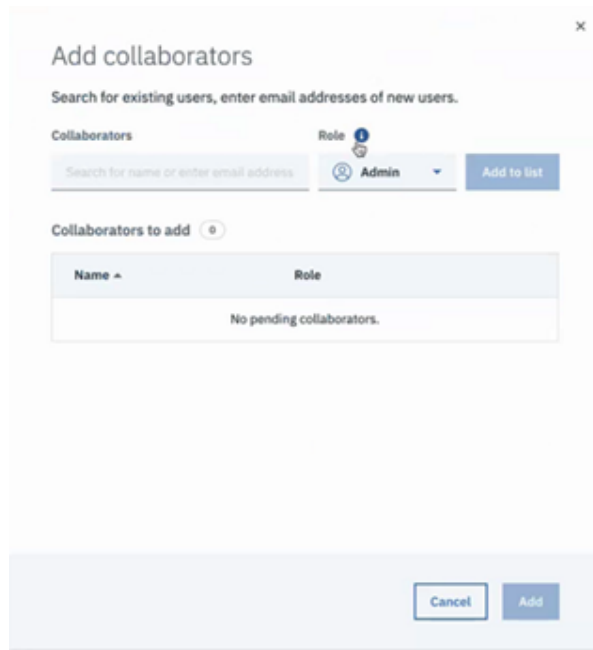


For more information, see [Deploying models](#).

### Access control for deployment spaces

Invite users with Watson Machine Learning Server accounts to collaborate in a deployment space.

Figure 37. Add collaborators



For more information, see [Deployment spaces](#).

#### Support for self-signed certificates

A self-signed certificate provides a certificate to enable SSL sessions between clients and the server. If you have a certificate, you can enter it when you connect to the Watson Machine Learning Server. For more information, see [Connecting to IBM Watson Machine Learning Server](#).

#### Support for macOS 10.15 (Catalina) operating system for Watson Studio Desktop 1.1

You can install Watson Studio Desktop 1.1 on macOS 10.15.

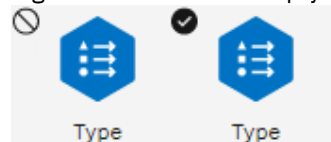
#### Support for Google BigQuery SQL pushback for Watson Studio Desktop

SQL pushback is now supported for the Google BigQuery database on Windows and Mac, for flows running on your local computer and not on a remote Watson Machine Learning Server. You must install a specific ODBC driver. See [SQL optimization](#).

#### New right-click options for nodes in SPSS Modeler flows

You can disable a node so it's ignored when the flow runs. And you can set up a cache on a node. For details, see [Disabling nodes in a flow](#) and [Caching options for nodes](#).

Figure 38. Node with empty cache vs. node with full cache



#### New SPSS Modeler documentation

##### SQL optimization

New documentation is available related to SQL pushback. See the new subsections under [SQL optimization](#).

##### Type node

New documentation is available for the Type node, one of the most frequently used nodes in SPSS Modeler. See the new subsections under [Type node](#).

In addition, the following features and changes from the **17 December 2019 (Subscription)** release are included:

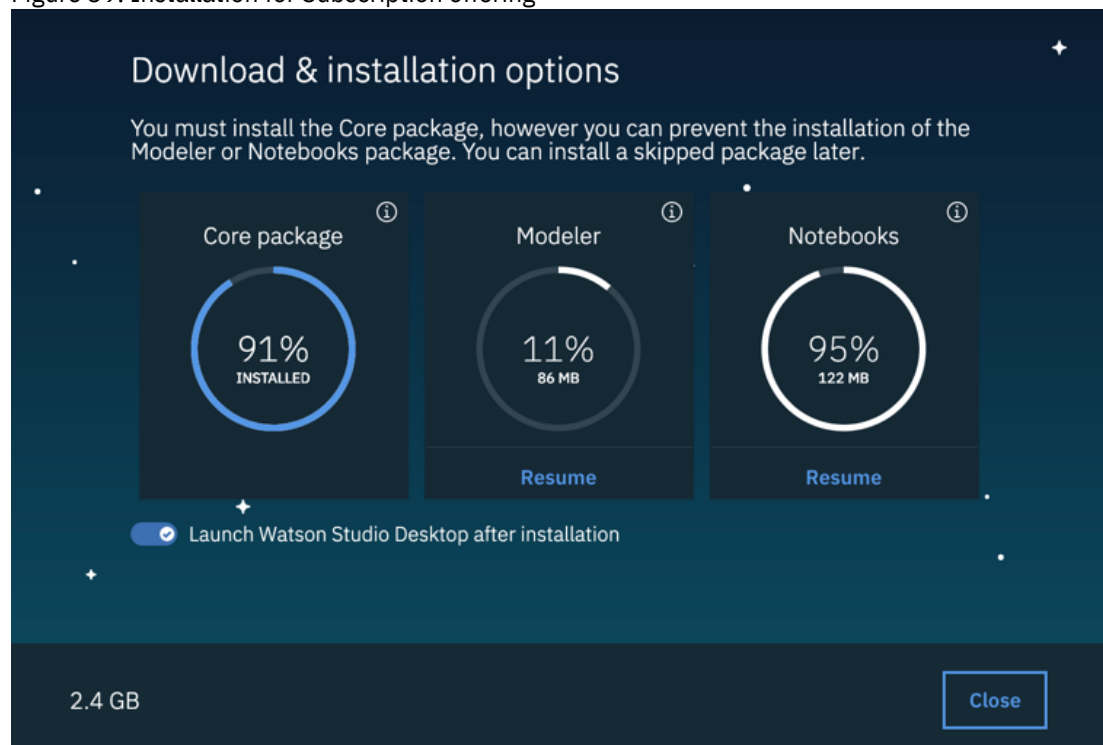
- [Project export and import](#)
- [New Text Analytics example project and tutorial](#)
- [Use a SQL SELECT statement to import data](#)
- [New SPSS Modeler nodes](#)
- [Use Data Refinery to change the decimal and thousands grouping symbols in all applicable columns](#)
- [Change the source of a Data Refinery flow](#)
- [Previous macOS and Windows operating system versions no longer supported](#)
- ["Object Storage OpenStack Swift \(Infrastructure\)" connection is discontinued](#)
- [New SPSS Modeler tutorial](#)
- [New SPSS Modeler documentation](#)

## 17 December 2019 (Subscription)

Faster and more efficient installation for Subscription

Watson Studio Desktop Subscription introduces a new user interface to install the application. You can choose which add-ons (Modeler or Notebooks) to install and when.

Figure 39. Installation for Subscription offering



Choose the steps for your operating system: [Installing Watson Studio Desktop \(Subscription\)](#).

Project export and import

You can now share your project assets with others by exporting your project. You export a project by clicking the Export project icon from the project toolbar.

Figure 40. Export project icon



The project assets that you select are downloaded as a project ZIP file to your desktop. You can then share this ZIP file with other IBM Watson Studio Desktop users who can import your ZIP file and use your assets in their new project.

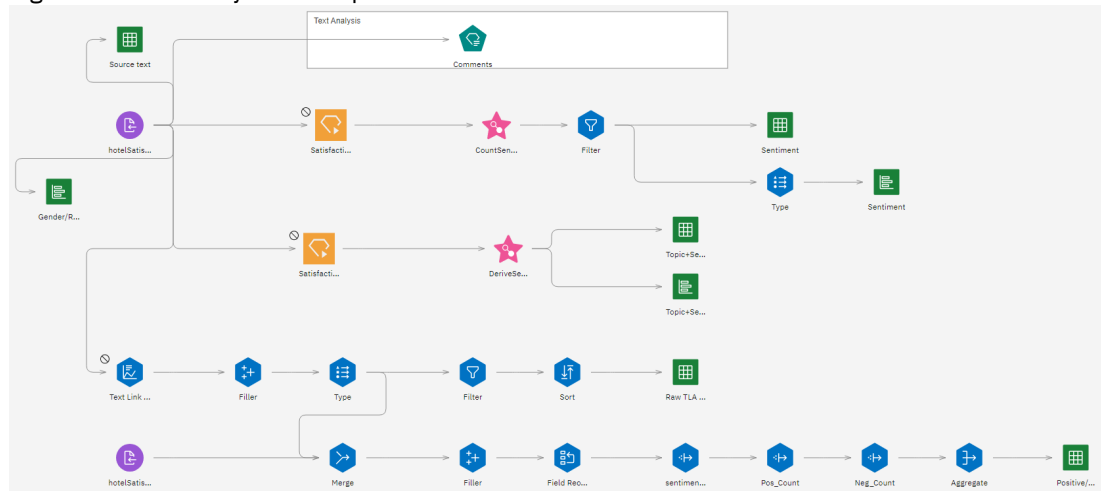
Currently, you can only import projects that are exported from IBM Watson Studio Desktop. For example, you can't import a project in IBM Watson Studio Desktop that was exported from IBM Watson Studio Cloud. Also note that SPSS Modeler flows aren't currently included. See [Exporting a project](#).



## New Text Analytics example project and tutorial

A new example project for Text Analytics is now installed with the product. Go to your projects, open the Example Project for Text Analytics, and open the HotelSatisfaction flow. For details about the flow, see the new [Hotel satisfaction example for Text Analytics](#) tutorial.

Figure 41. Text Analytics example



## Use a SQL SELECT statement to import data

In the Data Asset import node properties, you can now use SQL to import data from your database. See [Data Asset node](#).

Figure 42. Custom SQL

**Mode**

☐ Table ☒ SQL Query

**Source query**

```
select * from GOSALES.ORDER_DETAILS
where UNIT_COST > 40.000 LIMIT 4
```

## New SPSS Modeler nodes

The following nodes have been added for SPSS Modeler flows:

- [MultiLayerPerceptron-AS node](#)
- [HDBSCAN node](#)

## Use Data Refinery to change the decimal and thousands grouping symbols in all applicable columns

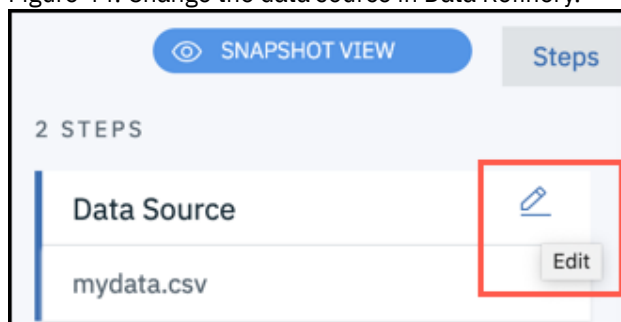
When you use the Convert column type GUI operation to detect and convert the data types for all the columns in a data asset, you can now also choose the decimal symbol and the thousands grouping symbol if the data is converted to an Integer or to a Decimal data type. Previously you had to select individual columns to specify the symbols.

Figure 43. Automatic conversion for all applicable columns.

Change the source of a Data Refinery flow

To change the source of a Data Refinery flow, click the Edit icon next to Data Source in the Steps panel.

Figure 44. Change the data source in Data Refinery.



For best results, the new data set should have a schema that is compatible to the original data set (for example, column names, number of columns, and data types). If the new data set has a different schema, operations that won't work with the schema will show errors. You can edit or delete the operations, or change the source to one that has a more compatible schema.

Support for macOS 10.15 (Catalina) operating system for Watson Studio Desktop Subscription

Watson Studio Desktop Subscription is supported on macOS 10.15 for both new installations and for in-app updates.

Previous macOS and Windows operating system versions no longer supported

Watson Studio Desktop (both Subscription and Watson Studio Desktop 1.#) is no longer supported on these operating systems:

- macOS High Sierra 10.13

- macOS Sierra 10.12
- Windows 7

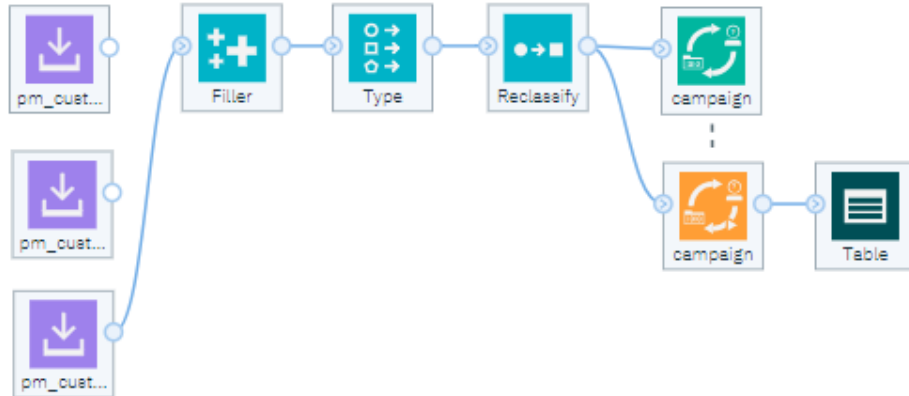
### "Object Storage OpenStack Swift (Infrastructure)" connection is discontinued

Support for the Object Storage OpenStack Swift (Infrastructure) connection is discontinued. For information, see [Object Storage OpenStack Swift - End of Support](#).

### New SPSS Modeler tutorial

A new [Making offers to customers \(self-learning\)](#) tutorial is available to accompany the twelfth flow in the example project that's installed with the product. As they become available, more tutorials will be added for the rest of the example flows. See [SPSS Modeler tutorials](#).

Figure 45. Example flow for making offers to customers (self-learning)



### New SPSS Modeler documentation

New documentation is available that may be helpful in handling missing values in your data. See [Missing data values](#).

## 14 November 2019 (Version 1.0.1 and Subscription)

### Use SPSS Modeler to change the field delimiter and decimal symbols in your source data

Different countries use different symbols to separate the integer part from the fractional part of a number and to separate fields in data. For example, you might use a comma instead of a period to separate the integer part from the fractional part of numbers. And, rather than using commas to separate fields in your data, you might use colons or tabs. Now you can use the new properties of the [Data Asset import node](#) and the [Data Asset Export node](#) to specify these symbols for field delimiter and decimal symbol. Available delimiters are comma, tab, or colon. Or you can specify your own custom delimiter.

Figure 46. New Data Asset node properties.

### Use Data Refinery to change the decimal and thousands grouping symbols in your source data

Different countries use different symbols to separate the integer part from the fractional part of a number and to group the thousands of a number. Now you can use the Convert column type GUI operation to specify these

symbols for Decimal and Integer data types.

Figure 47. Decimal and thousands grouping symbols.

**Operation** Code an operation to cleanse and shape your data

< Convert column type

Convert the data type of the columns to a different data type.

☐ Automatically detect and convert data types

Provide the columns and types to convert.

CONVERSION 1

number most closely matches the column's data.

Column	Type
euro_dec_grp	Decimal

Decimal symbol

Comma (,)

Thousands grouping symbol

Dot (.)

☐ Create new column for results ⓘ

+ Select column

euro_dec_grp
String
3.123,4
2.341,23
3,123
4,323
3.456,781
123
2,345

For more information, see Convert column type in [GUI operations in Data Refinery](#), under the FREQUENTLY USED category.

#### Data Refinery detects and converts data types

When you open a file in Data Refinery, for most file types all the columns are interpreted as the String data type. Now you can use the Convert column type GUI operation to detect and convert the data types for all the columns in a data asset.

Figure 48. Automatically detect and convert data types.

**Operation** Code an operation to cleanse and shape your data

< Convert column type

Convert the data type of the columns to a different data type.

☐ Automatically detect and convert data types

Provide the columns and types to convert.

euro_dec_grp
String
3.123,4
2.341,23
3.123

For more information, see Convert column type in [GUI operations in Data Refinery](#), under the FREQUENTLY USED category.

#### New videos

Watch the new videos for a [time series modeling example](#) and a [visual classification modeling example](#) in SPSS Modeler and how to use [visualization charts](#) in Data Refinery.

## 30 September 2019 (Version 1.0 and Subscription)

---

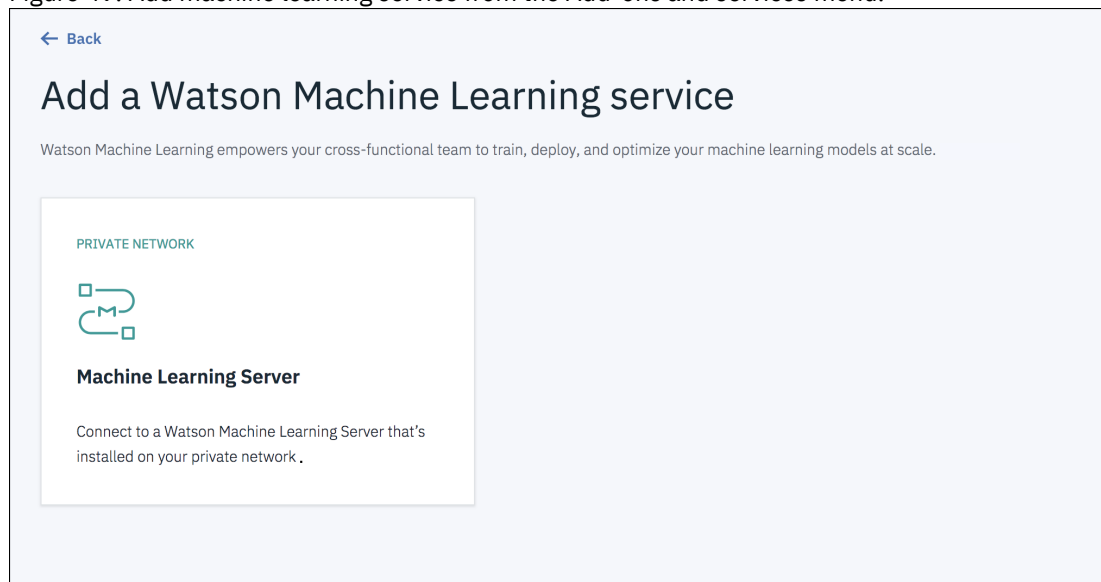
#### New IBM Watson Studio Desktop offering

If you prefer to purchase software instead of subscribing, you now have the choice to purchase Watson Studio Desktop at a one-time fee. Like the Subscription offering, Watson Studio Desktop 1.0 includes all the major SPSS Modeler features, Python notebooks, and Data Refinery. For descriptions of the two offerings, see [Watson Studio Desktop offerings](#).

#### Connect to Watson Machine Learning Server

IBM Watson Machine Learning Server is available as a separate offering that integrates with Watson Studio Desktop 1.0. When you [connect to a Watson Machine Learning Server](#) on your private network, you can use its processing power to run model flows. After you create and train a model in SPSS Modeler or in a Python notebook, you can promote the model from Watson Studio Desktop to a deployment space on the Watson Machine Learning Server, where you can configure, monitor, and deploy the model.

Figure 49. Add machine learning service from the Add-ons and services menu.



See [Saving and running models on Watson Machine Learning Server](#) and [Deploying using the Python client](#).

#### New SPSS Modeler nodes

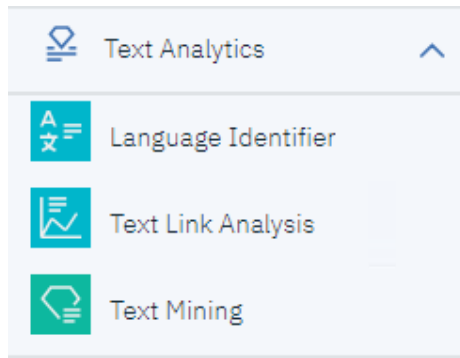
The following nodes have been added for SPSS Modeler flows:

- [Charts node](#)
- [Gaussian Mixture node](#)
- [GLMM node](#)
- [KDE node](#)
- [Sim Gen node](#)
- [Text Analytics nodes](#)

#### Text Analytics

Advanced text analysis functionality is now available for SPSS Modeler flows. See [Text Analytics](#).

Figure 50. New Text Analytics nodes



### SQL pushback

You can now push many data preparation and mining operations directly in the database. See [SQL optimization](#).

### Flow and SuperNode parameters

You can now set flow parameters in a flow script or in a flow's properties dialog box. You can also set parameters for SuperNodes, in which case they're visible only to nodes encapsulated within that SuperNode. See [Flow and SuperNode parameters](#).

### Run Data Refinery flows from Excel files that have the `.xlsx` format

Previously, Watson Studio Desktop supported creating Data Refinery flows only from Excel files with the `.xls` format. Now Excel files with the `.xlsx` format are supported. As before, only the first Excel sheet is read.

### Set a log level to troubleshoot problems

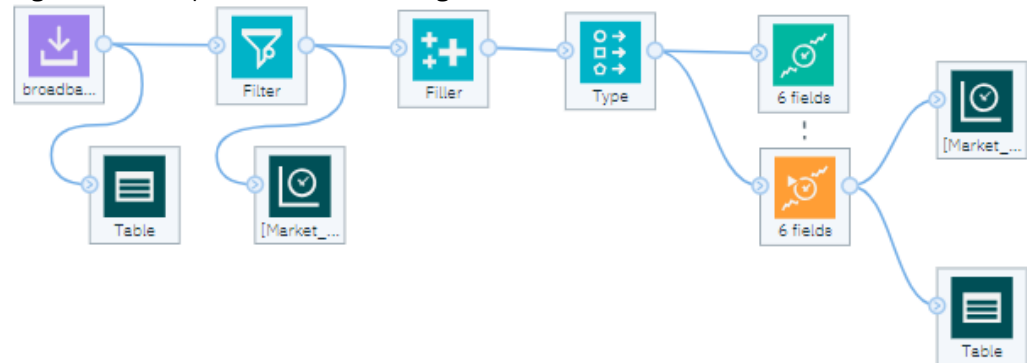
You can now run a command to set a log level so that you can see different amounts of diagnostic information. See [Setting the log level at runtime](#).

### New SPSS Modeler tutorials

Two new tutorials are available to accompany the tenth and eleventh flows in the example project that's installed with the product. As they become available, more tutorials will be added for the rest of the example flows. See [SPSS Modeler tutorials](#).

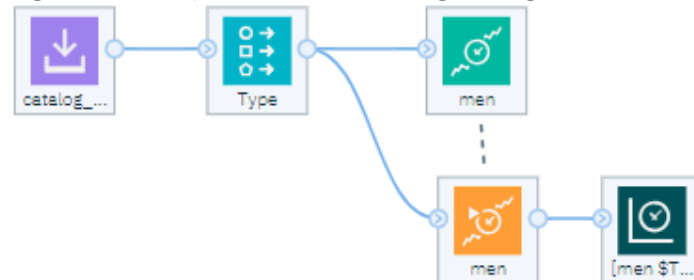
- [Forecasting bandwidth utilization](#)

Figure 51. Example flow for forecasting bandwidth utilization



- [Forecasting catalog sales](#)

Figure 52. Example flow for forecasting catalog sales



## New Data Refinery tutorial

The Data Refinery tutorial shows you how to build a Data Refinery flow to prepare data. See [Data Refinery tutorial: Shape raw data](#).

## 11 June 2019 (Subscription)

---

### New SPSS Modeler nodes

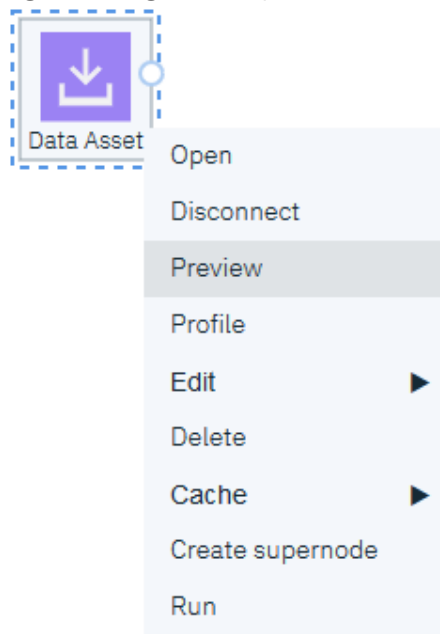
The following nodes have been added for SPSS Modeler flows:

- [Reproject node](#)
- [Space-Time-Boxes node](#)
- [Spatio-Temporal Prediction \(STP\) node](#)

### New right-click options for SPSS Modeler nodes

Previously, when you right-clicked a node and selected Preview, a Data tab, Profile tab, and Visualizations tab opened - allowing you to examine your flow's data in various ways. Now when you select Preview, you get a snapshot of your data that loads more quickly. Use the new right-click option called Profile to work with the full features such as the Visualizations tab.

Figure 53. Right-click options



### Export node conversion

Previously, when you imported a stream (.str) that was created in SPSS Modeler Subscription or SPSS Modeler client, Watson Studio Desktop converted your import nodes. Now you can also convert your export nodes. For details about configuring export nodes to export to your project or to a connection, see [Importing an SPSS Modeler stream](#).

Figure 54. Converting export nodes from a stream (.str)

## Migrate import and export nodes



Node	Name	Location	Target Path	Reset
Flat File	output.csv	My Project	<a href="#">Browse</a>	<a href="#">↺</a>
Excel	Excel	My Project	<a href="#">Browse</a>	<a href="#">↺</a>
SAS	SAS	My Project	<a href="#">Browse</a>	<a href="#">↺</a>
Analytic S...	Analytic Server	My Project	<a href="#">Browse</a>	<a href="#">↺</a>
IBM Cogn...	IBM Cognos A	My Project	<a href="#">Browse</a>	<a href="#">↺</a>

[Back to import](#)

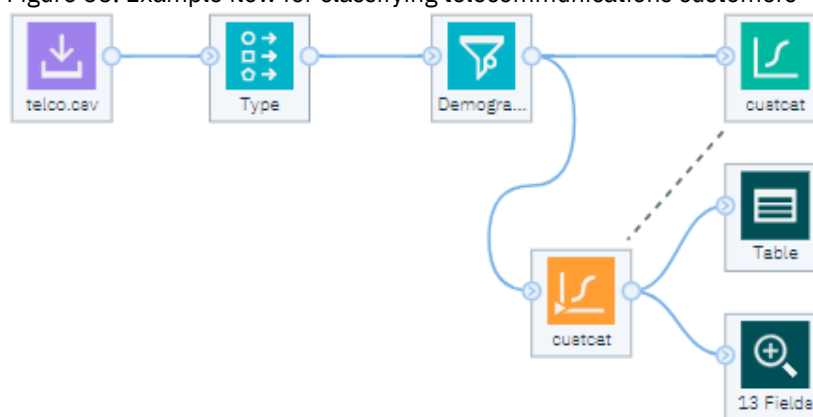
[Migrate](#)

### New SPSS Modeler tutorials

Two new tutorials are available to accompany the eighth and ninth flows in the example project that's installed with the product. As they become available, more tutorials will be added for the rest of the example flows. See [SPSS Modeler tutorials](#).

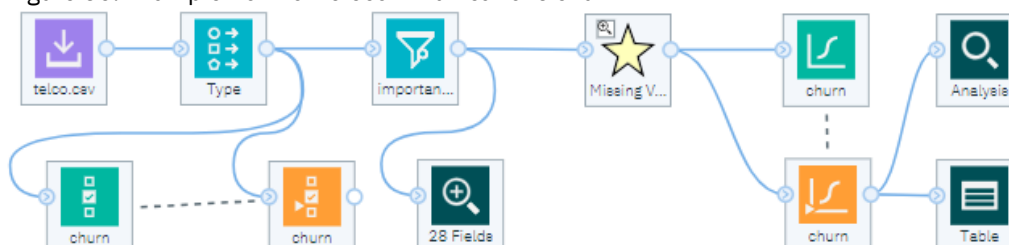
- [Classifying telecommunications customers](#)

Figure 55. Example flow for classifying telecommunications customers



- [Telecommunications churn](#)

Figure 56. Example flow for telecommunications churn

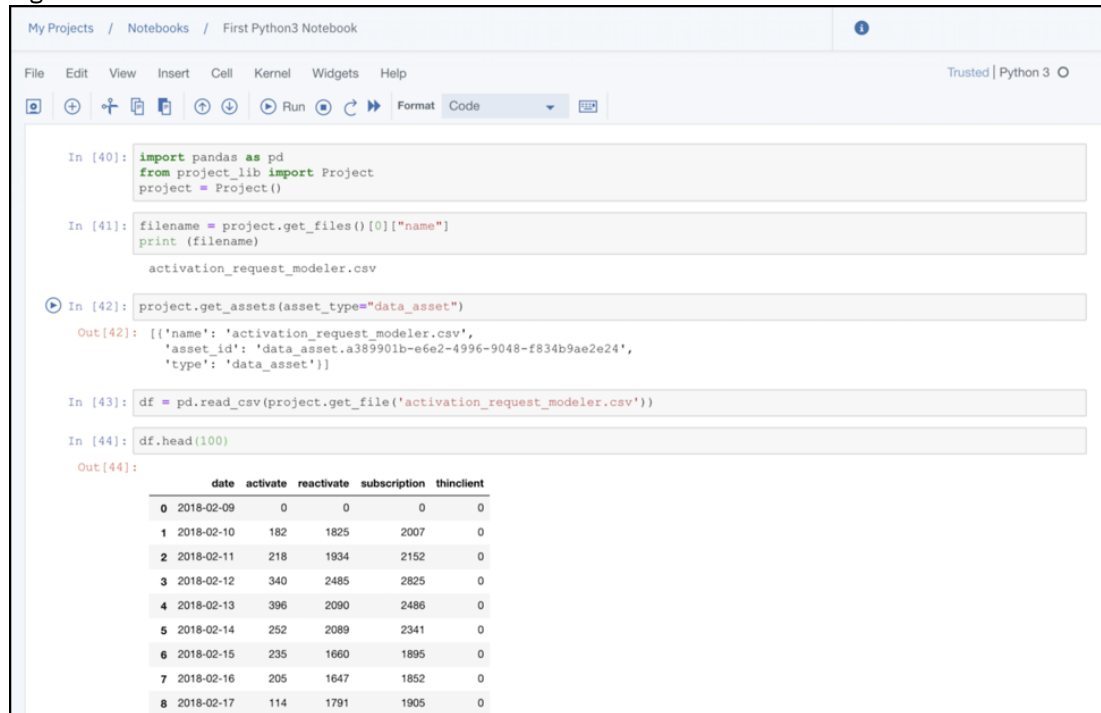




## Notebooks

You can now use Jupyter notebooks to analyze your data and form business insights that you share with collaborators. The first time you add a notebook to a project, you are prompted to install the [environments for running notebooks](#). After you have installed the environments, you can start creating your own notebooks.

Figure 57. Notebook



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [40]: import pandas as pd
         from project_lib import Project
         project = Project()

In [41]: filename = project.get_files()[0]["name"]
         print (filename)
         activation_request_modeler.csv

In [42]: project.get_assets(asset_type="data_asset")
Out[42]: [{"name": 'activation_request_modeler.csv',
              'asset_id': 'data_asset.a389901b-e6e2-4996-9048-f834b9ae2e24',
              'type': 'data_asset'}]

In [43]: df = pd.read_csv(project.get_file('activation_request_modeler.csv'))

In [44]: df.head(100)
Out[44]:
```

	date	activate	reactivate	subscription	thinclient
0	2018-02-09	0	0	0	0
1	2018-02-10	182	1825	2007	0
2	2018-02-11	218	1934	2152	0
3	2018-02-12	340	2485	2825	0
4	2018-02-13	396	2090	2486	0
5	2018-02-14	252	2089	2341	0
6	2018-02-15	235	1660	1895	0
7	2018-02-16	205	1647	1852	0
8	2018-02-17	114	1791	1905	0

Learn about working with [notebooks](#).

## Analyze data from remote data sources

You can now access data from a remote data source to build your models. Connect to the internet, and then click Add to project > Connection and enter the connection details. The connection is then stored in your project as a data asset.

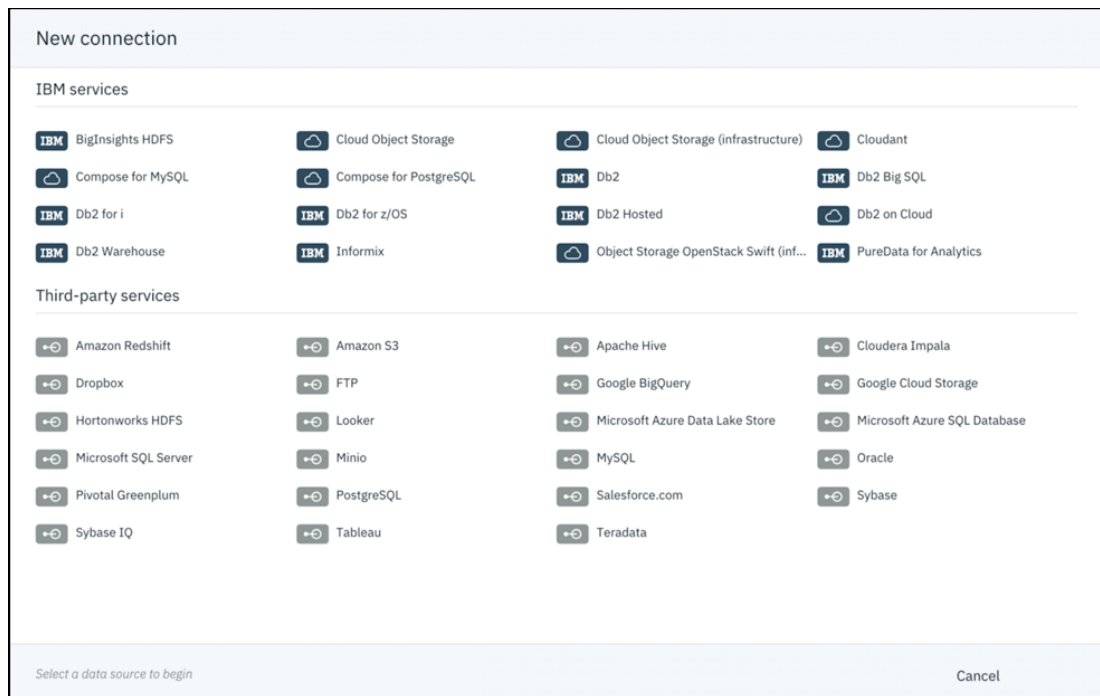
### For Data Refinery

Select the data asset as a source from Connections when you create the Data Refinery flow.

### For SPSS Modeler

Select the data asset from Connections when you pull in data with the Data Asset import node or when you write data with the Data Asset export node.

Figure 58. Data sources for connections

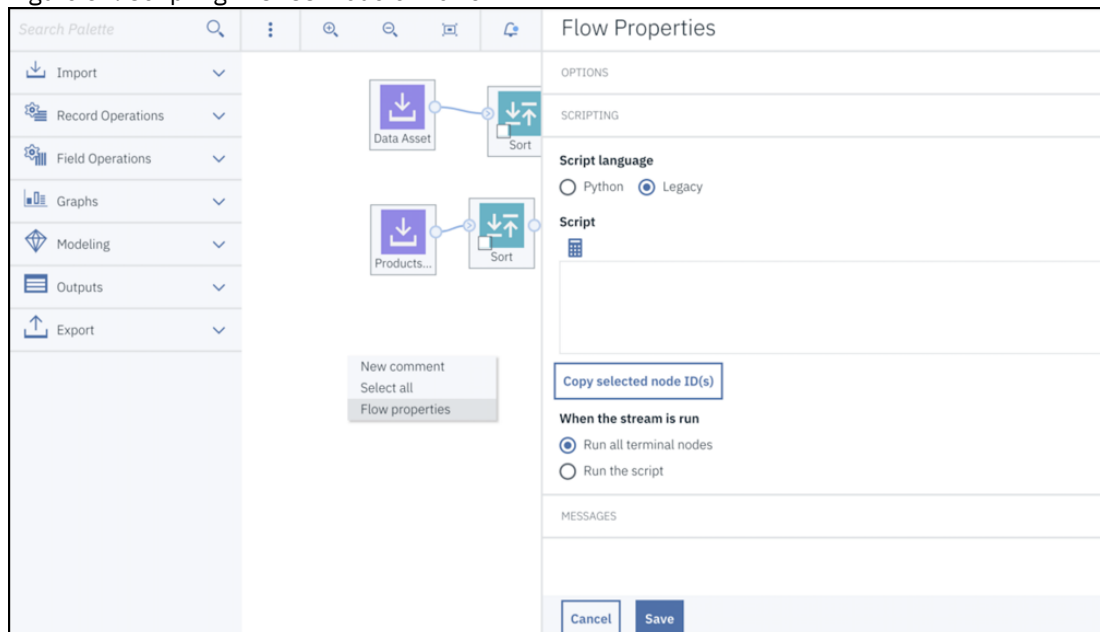


See [Connection types](#) for the list of on-prem databases and cloud services that you can connect to.

#### Add scripting to your SPSS Modeler flows

With the new scripting capability in SPSS Modeler flows, you can use scripts to customize operations within your flow. For example, you might want to specify a particular run order for terminal nodes. You can write scripts in Python or in the legacy SPSS scripting language, and a script editor is available to help you with script authoring and syntax. The scripts are saved with the flow.

Figure 59. Scripting in SPSS Modeler flows



See [Flow scripting](#).

Watch this short video for a demo of these features.

Figure 60. Video that introduces notebooks, connections, and scripting.

## What's New in IBM Watson Studio Desktop



<https://www.youtube.com/watch?v=DxQi4YZS4vM>

### New SPSS Modeler tutorials

Three new tutorials are available to accompany the fifth, sixth, and seventh flows in the example project that's installed with the product. As they become available, more tutorials will be added for the rest of the example flows. See [SPSS Modeler tutorials](#).

- [Drug treatment - exploratory graphs](#)
- [Screening predictors](#)
- [Reducing input data string length](#)

## 27 February 2019 (Subscription)

---

### New SPSS Modeler tutorial

A new [Automated data preparation](#) tutorial is available to accompany the fourth flow in the example project that's installed with the product. As they become available, more tutorials will be added for the rest of the example flows. See [SPSS Modeler tutorials](#).

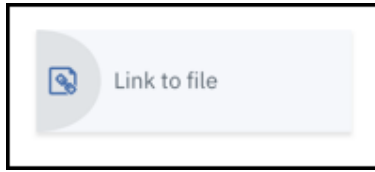
## 05 February 2019 (Subscription)

---

### Link to data assets

Now you can link to a file instead of loading it into a project. Unlike loading a file, you can reference the file from a directory path on your computer. Linking to a file saves disk space, and you can link to the same file from multiple projects. To link to a file click Add to project > Link to file.

Figure 61. Link to file



#### Work with Microsoft Excel files

- Refine data from a Microsoft Excel file. Excel files must have the `.xls` file extension. Only the first sheet is read.
- Pull Excel data (`.xls` or `.xlsx`) into your SPSS Modeler flows by using a Data Asset import node. Only the first sheet is imported.

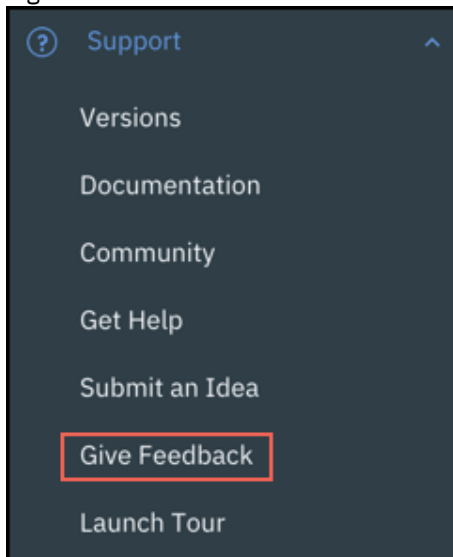
#### New SPSS Modeler tutorial

A new [Automated modeling for a continuous target](#) tutorial is available to accompany the third flow in the example project that's installed with the product. As they become available, more tutorials will be added for the rest of the example flows. See [SPSS Modeler tutorials](#).

#### Give us your feedback!

Let us know what you think of Watson Studio Desktop. Click the new Give Feedback link from the Support menu.

Figure 62. Give Feedback



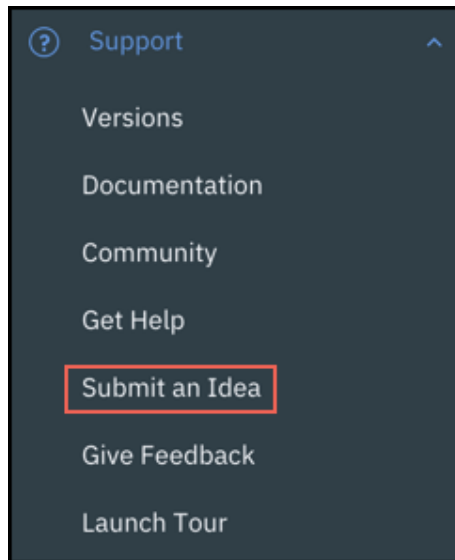
## 24 January 2019 (Subscription)

---

#### Submit your ideas!

Do you have an idea for a feature or product improvement? Click the new Submit an Idea link from the Support menu.

Figure 63. Submit an Idea



#### New nodes

The following nodes have been added for SPSS Modeler flows.

- [Anonymize node](#)
- [Time Intervals node](#)
- [SLRM node](#)

#### SPSS Modeler tutorials

Two new tutorials have been added to the documentation to accompany the first two flows in the example project that's installed with the product. As they become available, more tutorials will be added for the rest of the example flows. See [SPSS Modeler tutorials](#).

## 11 December 2018 (Subscription)

---

#### Watson Studio Desktop is generally available

Watson Studio Desktop gives you desktop data analysis using the features of the Watson Studio platform. Get started quickly by dropping your data files onto the project's Load pane. Refine your data to remove outliers and shape it for analysis. Then use the fully integrated SPSS Modeler to blend, graph, and develop predictive models. Export to deploy them into business operations to improve decision making.

Start your subscription today. See [Purchasing or starting a trial of Watson Studio Desktop \(Subscription\)](#).

**Parent topic:** [Overview of Watson Studio Desktop](#)

## Watson Studio Desktop offerings

---

IBM Watson Studio Desktop is available as two offerings. Both offerings include the major SPSS Modeler features, Python notebooks, and Data Refinery.

### Watson Studio Desktop Subscription

---

With Watson Studio Desktop Subscription, you rent the software as a monthly subscription. You regularly download application updates that have the new features and functionality. By default, the application automatically informs you when the updates are available. Watson Studio Desktop Subscription requires access to the internet for the updates. This offering does not support connections to Watson Machine Learning Server.

Also note that Watson Studio Desktop Subscription includes access to IBM SPSS Modeler Subscription, which includes features such as Text Analytics and SQL pushback. For more information about SPSS Modeler Subscription, see the [SPSS Modeler Subscription Knowledge Center](#).

[Get started with Watson Studio Desktop Subscription.](#)

## Watson Studio Desktop 1.1 or 2.0

---

Watson Studio Desktop 1.1 or Watson Studio Desktop 2.0 are a one-time purchase that does not require access to the internet beyond the software download. This offering supports connections to Watson Machine Learning Server on a private network. Watson Studio Desktop 1.1 or Watson Studio Desktop 2.0 are available in two license types:

- Single fee license covers the cost of the software at time of purchase and the first year's maintenance. New features and functionality are not available. At the end of the year, you can renew your maintenance contract or you can let it lapse. If you let it lapse, you will have access to the software in perpetuity, but you will not have access to support or maintenance.
- Maintenance (also known as S&S or "Subscription & Support") allows you to upgrade to future versions to get the latest features and functionality.

[Get started with Watson Studio Desktop 1.1](#)

[Get started with Watson Studio Desktop 2.0](#)

**Parent topic:** [Overview of Watson Studio Desktop](#)

## Limitations and known issues

---

Use the following information to help troubleshoot issues with IBM Watson Studio Desktop.

All issues apply to all offerings (Watson Studio Desktop Subscription and all versions of Watson Studio Desktop), unless otherwise specified.

- [General](#)
- [Activation \(Subscription\)](#)
- [Installation and updates](#)
- [Connections](#)
- [SPSS Modeler flows](#)
- [Data Refinery](#)
- [Notebooks](#)

### General

---

SAV file support

- SPSS Statistics .sav data files are supported in SPSS Modeler and Data Refinery.
- Encrypted .sav files aren't supported.
- .zsav files aren't supported.
- When using the Data Export node to export to a .sav file, spaces in variable names of the source data aren't supported.

Preview unavailable for linked files outside of the "C:/Users" directory on Windows (Subscription)

If you link to a file outside of the C:/Users directory or its subdirectories, you might not be able to preview the data when you click the data asset on the Project page. However, you can still refine the data with Data Refinery, import the file with the Modeler's Data Asset import node, or add the file to a notebook.

Preview unavailable for CSV files that have more than 5000 columns

If you click a data asset that is a CSV file with more than 5000 columns, the preview will not be available.

### Activation (Subscription)

---

The trial subscription cannot be activated on more than one computer (Subscription)

If you already installed the trial software on one computer and activated the subscription, you cannot activate the trial subscription on another computer.

#### Registration activation delay on the My IBM dashboard (Subscription)

After you register for the Watson Studio trial, you might need to wait a moment before you see the trial that is activated on the My IBM dashboard.

## Installation and updates

---

#### "Windows protected your PC" message when you install Watson Studio Desktop on Windows (Subscription)

During installation, if you receive a message that states that Microsoft Defender SmartScreen prevented an unrecognized app from starting, click Run anyway.

#### "EBUSY" error when you install Watson Studio Desktop on Windows

When you install Watson Studio Desktop on Windows, if you encounter an error that includes the text "Error: EBUSY: resource busy or locked, rmdir", restart the installation with an environment variable:

1. Close the installation.
2. Open the command prompt with "Run as Administrator."
3. Enter the following command:

```
set WSD_CLEAN_INSTALL=true&&"C:\Program Files\IBM Watson Studio\IBM Watson Studio.exe
```

4. If the error persists, restart your computer.

#### "ERR\_TIME\_OUT" error when you update Watson Studio Desktop (Subscription)

If you receive a JavaScript `ERR_TIME_OUT` error when you update Watson Studio Desktop, it is caused by a temporary network outage. Try updating later. Alternatively, go to <http://ibm.com>, log in to My IBM, and re-download the latest version of Watson Studio Desktop.

#### Installation time on Windows (Version 1.1)

Installation time for Watson Studio Desktop might take over 10 minutes to complete on some Windows machines. The time depends on the machine's performance.

## Connections

---

#### Google BigQuery connections and SPSS Modeler Data Asset export node

In the Data Asset export node, if the data set you're exporting to already exists, the option Replace the data set isn't supported for Google BigQuery.

#### Connection details cannot be changed in the "Edit Connection" dialog box

After you create a connection, you cannot edit the connection details (username, hostname, etc.). But you can change the connection name or description. If you need to change the connection details, delete and re-create the connection.

## SPSS Modeler flows

---

#### Exporting to a SAV file

When using the Data Asset Export node to export to an SPSS Statistics SAV file (.sav), the Replace data asset option won't work if the input schema doesn't match the output schema. The schema of the existing file you want to replace must match.

#### Data Asset Export node (Subscription)

When exporting to a Connection, the Connection must contain an existing file.

#### Oracle database

When using the Data Asset node to pull in data from an Oracle database, you may encounter errors. To work around this issue, perform the following steps:

1. Add or modify the following line in sqlnet.ora of your Oracle database.

```
SQLNET.ALLOWED_LOGON_VERSION_SERVER=8
```

2. Restart the database.
3. Create a new user and set a password for it, or change the password of an existing user.
4. Use the following query to verify that the user has 10G in its PASSWORD\_VERSIONS. If it doesn't, add it.

```
select USERNAME,ACCOUNT_STATUS,PASSWORD_VERSIONS from dba_users;
```

5. In sqlnet.ora, change the value from 8 to 11 in the following line.

```
SQLNET.ALLOWED_LOGON_VERSION_SERVER=11
```

6. Verify that the user can still connect to the database.

#### Graphs in Data Audit node (Subscription)

After running a Data Audit node, your output may not show the graphs in the Graph column.

#### Delimiter and decimal options

You can only set field delimiter and decimal options for Data Asset nodes (.csv). These options aren't available for connections at this time.

#### Chart node

If you uninstalled a previous version of Watson Studio Desktop and then installed this latest version, the Chart node may not work properly due to some folders not being cleaned up. To work around this issue, you can delete the common-canvas-custom-ctrl folder from the installation directory and then reinstall Watson Studio Desktop.

#### Running flows on a Watson Machine Learning Server

- By default, flows run on your local computer. If you switch to run a flow on a Watson Machine Learning Server, note that local project assets aren't currently supported. Only data connections can be used when running a flow on the server.
- If a flow references a connection, the connection must be accessible on both the local machine (to configure the connection) and the remote Watson Machine Learning Server (to run the flow).
- If a flow contains Text Analytics nodes, you can't run it on a Watson Machine Learning Server. You must run the flow locally.
- Don't import a stream file (.str) that requires migration (see [Importing an SPSS Modeler stream](#) for more information) to a flow that runs on the Watson Machine Learning Server.

#### Special characters in descriptions (Subscription)

When creating a new flow, using special characters in the Description field may result in an error.

#### Migrating Import nodes

If you import a stream (.str) to your flow that was created in SPSS Modeler and contains one or more unsupported Import nodes, you'll be prompted to migrate the Import nodes to data assets. If the stream contains multiple Import nodes that use the same data file, then you must first add that file to your project as a data asset before migrating because the migration can't upload the same file to more than one Import node. After adding the data asset to your project, reopen the flow and proceed with the migration using the new data asset.

#### Text Analytics

- When you click Generate new model in the Interactive Workbench, the new model isn't connected to any nodes. For example, if you click Generate new model ten times, you'll get ten new modeling nodes.
- Editing of linguistic resources isn't available in this version.
- If a flow contains Text Analytics nodes, you can't run it on a Watson Machine Learning Server. You must run the flow locally.
- In the Interactive Workbench, when you click Generate new model, a new model nugget is created in your flow. If you generate multiple models, they all have the same name, so it may be difficult to



differentiate them. One recommendation is to use annotations to help identify them (double-click a model nugget to open its properties, then go to ANNOTATIONS).

## Data Refinery

---

Data Refinery flow not saved when you click outside the Data Refinery user interface

If you apply an operation in Data Refinery, and then you click a menu item on the left-side menu, your changes are not saved. Be sure to save your Data Refinery flow before you click outside of the Data Refinery user interface.

Cannot run a Data Refinery flow if project name includes a comma and data asset exceeds 20 MB (Version 2.0)

If the project name includes a comma (,) and the data asset exceeds 20 MB, the Data Refinery flow will fail. If the data asset exceeds 20 MB, change the project name to not include a comma.

CSV file limitations

Be sure that CSV files are correctly formatted and conform to the following rules:

- Files can't have some rows ending with null values and some columns containing values enclosed in double quotation marks.
- Two consecutive commas in a row indicate an empty column.
- If a row ends with a comma, an additional column is created.

Column name limitations

Be sure that column names conform to the following rules:

- The column names are unique within the data set.
- The column names are not [reserved words](#) for R.
- The column names are not numbers. A workaround is to enclose the column names in double quotation marks ("").

White space characters are considered as part of the data

If your data includes columns that contain white space (blank) characters, Data Refinery considers those white space characters as part of the data, even though you can't see them in the grid. Be aware that some database tools might pad character strings with white space characters to make all the data in a column the same length and this change will affect the results of Data Refinery operations that compare data.

Maximum file size (Version 1.1)

For Data Refinery, the maximum file size depends on the amount of RAM in your computer:

- The maximum file size is 100 MB for 16 GB RAM.
- The maximum file size is 50 MB for 8 GB RAM.

## Notebooks

---

Data is not automatically saved when leaving a notebook on Windows

If you create a notebook and want to leave, you must save (File > Save) your latest updates before leaving your notebook.

It is also recommended that you stop the notebook kernel when leaving your notebook (File > Stop kernel).

Notebooks cannot be downloaded from the notebook environment

If you created a notebook and want to share it, you cannot download your notebook file when the notebook is open from File > Download as in the notebook environment. To retrieve your notebook files (the .ipynb files), navigate to the following directory:

- For Windows: %APPDATA%\IBM Watson Studio\projects\<project-name>\assets\notebook
- For macOS: \$HOME/Library/Application Support/IBM Watson Studio/projects/<project-name>/assets/notebook

## Feature differences between Watson Studio deployments

This table describes the feature differences between the Watson Studio service in Cloud Pak for Data as a Service, in Cloud Pak for Data 2.5, in Cloud Pak for Data 3.0.1, and in Watson Studio Desktop. Watson Studio Local 2.1 has the same features as the service in Cloud Pak for Data 2.5.

\* indicates that the feature is limited or not available in some offering plans.

+ indicates that the feature requires an additional service that might cost extra.

Feature	Cloud Pak for Data as a Service	Cloud Pak for Data 2.5	Cloud Pak for Data 3.0.1	Watson Studio Desktop
Collaboration in projects	✓	✓	✓	
Jupyter notebooks	✓	✓	✓	✓
JupyterLab		✓	✓	
Scripts		✓	✓	
Data Refinery	✓	✓	✓	✓
Project import and export	✓	✓	✓	✓
Git integration		✓	✓	
Anaconda Repository integration			+	
Crowd sourced annotation	*			
RStudio	✓	+	+	
Remote data sources	*	✓	✓	✓
Personal credentials		✓	✓	
Upload data files	✓	✓	✓	✓
Link to data files				✓
Environment runtimes	✓	✓, +	✓, +	
GPU environments	✓	+	+	
Job scheduling	✓	✓	✓	
SPSS Modeler flows	✓	+	+	✓
Decision Optimization	✓	+	+	
Visual Recognition	+			
Natural Language Classification	+			
Dashboards	+	+	+	
Streams flows	+	+	+	
Hadoop integration		+	+	
Watson Knowledge Catalog integration	+	✓	✓	

## Watson Machine Learning

This table describes the differences in features between the Watson Machine Learning service on Cloud Pak for Data as a Service, Watson Machine Learning Server 2.x for Watson Studio Desktop, the Watson Machine Learning service for Cloud Pak for Data 2.5, and the Watson Machine Learning service for Cloud Pak for Data 3.0.1. The Watson Machine Learning service on Watson Studio Local 2.1 has the same functionality as on Cloud Pak for Data 2.5.

+ indicates that the feature requires an additional service that might cost extra.

\* indicates that the feature is limited or not available in some offering plans.

Feature	Cloud Pak for Data as a Service	Cloud Pak for Data 2.5	Cloud Pak for Data 3.0.1	Watson Machine Learning Server 1.x	Watson Machine Learning Server Version 2.0
Experiments	✓	✓	✓		
AutoAI	✓	✓	✓		✓
Collaboration in deployment spaces		✓	✓	✓	✓
Deploy models	✓	✓	✓	✓	✓
Deploy functions	✓	✓	✓	✓	✓
Deploy scripts			✓		
Deploy Shiny apps			✓		
Continuous learning	✓				
Performance monitoring	✓	✓	✓		
v4 APIs		✓	✓	✓	✓
Batch deployments	✓	✓	✓	✓	✓
Online deployments	✓	✓	✓	✓	✓
Core ML deployments	✓	✓	✓	✓	✓
Evaluation jobs	✓	✓	✓		
Deploy from projects	✓	✓	✓		
Import and export deployment spaces			✓		✓
Watson Machine Learning Accelerator integration		+	+		

## Supported machine learning frameworks

Watson Studio Desktop includes these popular tools, libraries, and frameworks to build machine learning models. This topic lists supported versions.

**Note:** If you are connecting to a Watson Machine Learning Server, see [Supported frameworks for Watson Machine Learning Server](#) for the frameworks and restrictions that apply to Watson Machine Learning Server.

Framework	Version
IBM SPSS Modeler	18.2.2
Keras (Included with Python 3.6.)	2.2.4
Predictive Model Markup Language (PMML)	PMML 3.0 to 4.3
Scikit-Learn (Included with Python 3.6.)	0.20.3
TensorFlow (Included with Python 3.6.)	1.12
XGBoost (Included with Python 3.6.)	0.90

Other machine learning libraries can be installed by using `conda` or `pip` commands from inside a notebook.

## Data source connections

Watson Studio Desktop and Watson Machine Learning Server provide many ways to use and access data from a remote data source connection.

## Project connection assets

---

Create a connection asset for a data source so that you can load data from it. A connection asset contains the information necessary to create a connection to a data source. Most connection assets are supported for Data Refinery flows, Notebooks, and SPSS Modeler flows. See the following information:

- [Creating a connection asset](#)
- [Supported project connections](#)

## SQL optimization

---

Many of the project connection assets support SPSS Modeler SQL optimization (SQL pushback). For a list of supported connections and instructions, see the following information:

- [Supported connections and instructions for SQL optimization](#)

## Connections for running SPSS Modeler flows on a Watson Machine Learning Server

---

You can run SPSS Modeler flows on a Watson Machine Learning Server for better performance. For a list of supported connections and instructions, see the following information:

- [Supported connections and instructions for running flows on a server](#)

## Batch deployment on remote data sources with Watson Machine Learning Server

---

If you're connected to a Watson Machine Learning Server, you can promote or add data sources to a deployment space to use with batch deployment jobs. For more information, see:

- [Instructions](#)
- [Supported interfaces and data sources](#)

## Security in Watson Studio Desktop

---

Watson Studio Desktop uses these security features.

### Authentication

---

- To activate Watson Studio Desktop Subscription, you must log in with your IBM account (also known as an IBMid).
- Watson Studio Desktop 1.1 or Watson Studio Desktop 2.0 are purchased through Passport Advantage. License information is not stored on your computer.
- Credentials for connections to Watson Machine Learning Server are stored in Base64 format on the client's machine.

### Data at rest

---

Watson Studio Desktop uses an AES 256 algorithm to encrypt values that are sensitive including keys such as passwords, client ID secrets, etc. When you create a connection for an SPSS Modeler flow or a Data Refinery flow, any credentials used by that connection are encrypted with an AES 256 algorithm and are stored with the connection asset in the project folder in which the asset resides.

### Data in transit

---

Watson Studio Desktop uses the Transport Layer Security (TLS) protocol to encrypt data in transit. TLS ensures privacy between communicating applications and their users on the internet. TLS is the successor to the Secure Sockets Layer (SSL) protocol.

## IBM Watson Studio Desktop considerations for GDPR readiness

---

Information about features of Watson Studio Desktop that you can configure, and aspects of the product's use, that you should consider to help your organization with GDPR readiness.

### For PID(s): 5737-B93

---

#### Notice:

---

This document is intended to help you in your preparations for GDPR readiness. It provides information about features of Watson Studio Desktop that you can configure, and aspects of the product's use, that you should consider to help your organization with GDPR readiness. This information is not an exhaustive list, due to the many ways that clients can choose and configure features, and the large variety of ways that the product can be used in itself and with third-party applications and systems.

Clients are responsible for ensuring their own compliance with various laws and regulations, including the European Union General Data Protection Regulation. Clients are solely responsible for obtaining advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulations that may affect the client's business and any actions the clients may need to take to comply with such laws and regulations.

The products, services, and other capabilities described herein are not suitable for all client situations and may have restricted availability. IBM does not provide legal, accounting, or auditing advice or represent or warrant that its services or products will ensure that clients are in compliance with any law or regulation.

### Table of contents

---

1. [GDPR Overview](#)
2. [Product Configuration for GDPR](#)
3. [Data Life Cycle](#)
4. [Data Collection](#)
5. [Data Storage](#)
6. [Data Access](#)
7. [Data Processing](#)
8. [Data Deletion](#)
9. [Data Monitoring](#)
10. [Responding to Data Subject Rights](#)

### GDPR Overview

---

General Data Protection Regulation (GDPR) has been adopted by the European Union and will apply from May 25, 2018.

Why is GDPR important?

GDPR establishes a stronger data protection regulatory framework for processing of personal data of individuals. GDPR brings:

- New and enhanced rights for individuals
- Widened definition of personal data
- New obligations for processors
- Potential for significant financial penalties for non-compliance
- Compulsory data breach notification

Read more about GDPR:

- [EU GDPR Information Portal](#)
- [ibm.com/GDPR web site](https://ibm.com/GDPR)

**Parent topic:** [Security](#)

## Product configuration - considerations for GDPR readiness

---

### Data Life Cycle

GDPR requires that personal data is:

- Processed lawfully, fairly and in a transparent manner in relation to individuals.
- Collected for specified, explicit and legitimate purposes.
- Adequate, relevant and limited to what is necessary.
- Accurate and, where necessary, kept up to date. Every reasonable step must be taken to ensure that inaccurate personal data are erased or rectified without delay.
- Kept in a form which permits identification of the data subject for no longer than necessary.

### What is the end-to-end process through which personal data go through when using Watson Studio Desktop?

What types of data flow through Watson Studio Desktop?

Watson Studio Desktop is available as a base deployment environment that includes a graphical drag and drop interface, extensive data preparation capabilities including automatic data preparation, interactive decision trees, a large library of algorithms, geospatial analysis, extensibility into R, Python, Spark, and much more.

Watson Studio Desktop is comprised of the license server (cloud) and the client software (installed on end user machines). The license server stores user licensing information and personal data that includes each user's subscription ID, IBM ID, First Name, Last Name, machine ID, and IP address. The information is used to validate each user's license entitlement when they log in to the client software.

The client software acts as on-premise software and does not store any personal data. Users have total control over how their data is handled, used, or stored. The data may, or may not, include personal user data. Watson Studio Desktop acts as a general tool for processing data based on the user's business requirements. User data is transparent to the software and is not used in any way outside of the application.

Note: When you create a connection for an SPSS Modeler flow or a Data Refinery flow, any credentials used by that connection are encrypted with an AES 256 algorithm and are stored with the connection asset in the project folder in which the asset resides.

Where in the process?

For the license server, data processing includes registration, authentication, backup, logging, and monitoring. Some processes require the handling of personal data.

The client software acts as a general tool that is used to manipulate and analyze data (based on user's business requirements). The data may, or may not, include personal user data. Users have total control over how their data is handled, used, or stored.

The following sections provide more detail about how the license server processes data (does not apply to the client software).

- **Registration:** The license server works together with other IBM IDaaS services to process each user's subscription registration. A pre-condition is that users must register an IBM ID, via the IBM IDaaS system, before they can register Watson Studio Desktop. After users have acquired an IBM ID, they can register to use the product from the online IBM MarketPlace. During the Watson Studio Desktop registration process, new

subscription IDs are created and passed through the license server (along with each user's IBM ID, first name, and last name from the IBM IDaaS system).

- **Authentication:** The license server leverages the IBM IDaaS system to perform user authentication. The IBM IDaaS system requires user credentials (user ID and password) when authenticating each user. Assuming a successful authentication, the user's IBM ID and machine ID information is passed to the license server for license validation. During a user's successful login, the user's machine ID is collected and stored on the license server (for future license validation purposes).
- **Backup:** The license server backup process relies on the IBM Db2 Warehouse on Cloud service. According to the IBM DB2 Warehouse documentation, each backup is encrypted and retained for 14 days. The license server is also configured for high availability (two servers are configured as a highly available pair).
- **Logging:** The license server transaction logs are used for recovering data after a failure. The logs contain personal information that includes the user's IBM ID, first name, last name, machine ID, and IP address. The logs are stored in the IBM cloud logging system, which encrypts data during transaction and at rest. Log files are stored for 14 days.
- **Monitoring:** The license server monitoring interface checks and records the health status of each sub service. The health status data consists of simple on/off information and does not contain any personal or transactional data.

For what purpose?

On the license server, the stored and processed data includes the IBM ID, first name, last name, and machine ID information for each user and is only used for license validation purposes. Other data, such as user IP addresses, is handled by the underlying services (backup, logging, monitoring) which is essential for supporting the main licensing service.

The client software acts as a general tool that is used to manipulate and analyze data (based on the user's business requirements). The data may or may not include personal user data. Users have total control over how their data is handled, used, or stored.

## Data Collection

The license server requires that account information pass through the IBM IDaaS system. Users must first register an IBM ID from the IBM IDaaS system. After a user has an IBM ID, and registers for the Watson Studio Desktop software, the account information is passed through the IBM IDaaS system to the license server. During a user's successful login, the user's machine ID is collected and stored on the license server (for future license validation purposes).

## Data Storage

The following Data Storage mechanisms are used by Watson Studio Desktop. Users may wish to consider this information when assessing their GDPR readiness.

- **Storage of account data:** On the license server, user licensing entitlement data is used to authenticate each individual license entitlement. Data is stored in the license server database, which is a Db2 Warehouse on the IBM Cloud service. User authentication data is stored in the IBM IDaaS system. Data is encrypted both at rest and during each transaction. Users can request to update or delete their account data.

The client software does not store account data.

- **Storage of client data:** The license server is used for user license authentication and stores only user licensing entitlement data (which includes the user's IBM ID, first name, last name, and machine ID). The license server does not store any other type of client data. The logging system, which is essential for supporting the main licensing service, stores the user's IP address.

The client software acts as a general tool that is used to manipulate and analyze data (based on the user's business requirements). Users have total control over how their data is handled, used, or stored.

- **Storage in backups:** The license server backup relies on the IBM Db2 Warehouse on Cloud service. Backup data is stored in a security facility on IBM Cloud. According to the IBM Db2 Warehouse documentation, each backup is encrypted and retained for 14 days.

The client software does not store backup data. Users must manage their own backup data and storage solution.

- **Storage in archives:** License server transaction logs are stored in a security facility on the IBM Cloud. The logs are stored in the IBM logging system, which is encrypted at rest. Log files are stored for 14 days. License server monitoring data is stored in the IBM Monitoring system. No personal data is stored in the Monitoring system (only the service health status data).

The client software does not store archived data.

## Data Access

- **Clients do not have access to personal data on the license server.** License server administrators are the only individuals who have access to personal data. Users can log on to the IBM IDaaS system to update their personal information (such as IBM ID, first name, last name, etc.). Users cannot access the supporting system data (such as backup data, log files, and monitoring data).
- **Roles and access rights:** Privileged administrators are the only individuals who have access to personal data on the license server. The privileged administrator role requires the approval from both the SPSS License Server development and DevOps managers. Does not apply to the client software.
- **Separation of duties:** License server privileged access rights are assigned to users on an as-needed basis and based on the minimum requirements for the role. Only DevOps personnel who work on deployment have access to the production hosts, know dashDB passwords, and have access to the Softlayer Object Store. Only personnel involved in SME have access to, and know details about, the SSL certificates. Does not apply to the client software.
- **Privileged Administrators:** The development manager and operation manager assign license server privileged access rights to users on an as-needed basis and based on the minimum requirements for the role. Only DevOps personnel who work on deployment have access to the production hosts, know dashDB passwords, and have access to the Softlayer Object Store. Only personnel involved in SME have access to, and know details about, the SSL certificates. Does not apply to the client software.
- **Administrators:** The development manager and operation manager assign license server privileged access rights to users on an as-needed basis and based on the minimum requirements for the role. Only DevOps personnel who work on deployment have access to the production hosts, know dashDB passwords, and have access to the Softlayer Object Store. Only personnel involved in SME have access to, and know details about, the SSL certificates. Does not apply to the client software.
- **Activity logs:** The license server does not currently include a method for generating audit logs. Does not apply to the client software.

## Data Processing

- **Encryption in motion:** To protect all personal data on the license server, communication is encrypted between the client software, the IBM IDaaS, and the license server using Transport Layer Security (TLS). Logging also uses the same encryption method to protect personal data between the license server and IBM cloud logging. Monitoring data does not include any personal information. Does not apply to the client software.
- **Encryption at rest:** To protect personal information, in the license server database and backups at rest and on the IBM Cloud, the database and backup data is encrypted using AES. Does not apply to the client software.
- **Encryption key ownership:** Encryption key access on the license server is restricted. The encryption keys enable individuals to read or decrypt personal data on a network or storage device. A trusted, privileged administrator is responsible for the safekeeping and maintenance (rotation) of the encryption keys. Does not apply to the client software.

## Data Deletion

- **Client data deletion:** The license server is used only for customer license authentication. The license server stores only the customer license entitlement data and does not store other types of client data.
- **Account data deletion:** The license server uses user account information to authenticate individual license entitlement (the information for which is supplied by the IBM IDaaS system). Users can request to delete their



accounts from the IBM IDaaS system. A user must cancel their licensing entitlement in order to delete the licensing entitlement information from the license server. Does not apply to the client software.

## Data Monitoring

You should regularly test, assess, and evaluate the effectiveness of your technical and organizational measures to comply with GDPR. These measures should include ongoing privacy assessments, threat modeling, centralized security logging, and monitoring, among others.

- **Log monitoring:** The license Server transaction logs (which are used for recovering data after a failure) contain personal information such as the user's IBM ID, first name, last name, machine ID, and IP address. The log files are stored in the IBM cloud logging system which is encrypted both during transaction and at rest. Log files are retained for 14 days. Does not apply to client software.
- **Data monitoring:** The license server stores data from Db2 Warehouse on Cloud, which provides monitoring functions that allow administrators to track the processing of personal data. Does not apply to the client software.
- **Activity monitoring:** The license server does not currently include a method for generating audit logs. Does not apply to the client software.

## Responding to Data Subject Rights

- **Right to Access:** Users cannot directly access their licensing entitlement data on the license server. Authentication from the client software is used to validate licensing entitlement. After a successful authentication, the user's first and last name are displayed. User account-related information (including the user's IBM ID, first name, last name, etc.) can be accessed and updated from the IBM IDaaS system. Other supporting system data (such as backup data, logs, and monitoring data) cannot be accessed. Does not apply to the client software.
- **Right to Modify:** Users cannot edit their individual licensing entitlement data on the license server. Authentication from the client software validates the user licensing entitlement. After a successful authentication, the user's first and last name are displayed. Users can unsubscribe subscriptions from the IBM Products and Services web site. User account-related information (including the user's IBM ID, first name, last name, etc.) can be accessed and updated from the IBM IDaaS system. Other supporting system data (such as backup data, logs, and monitoring data) cannot be accessed. Does not apply to the client software.
- **Right to Restrict Processing:** Users cannot edit their individual licensing entitlement data on the license server. Authentication from the client software validates the user licensing entitlement. After a successful authentication, the user's first and last name are displayed. Users can unsubscribe subscriptions from the IBM Products and Services web site. User account-related information (including the user's IBM ID, first name, last name, etc.) can be accessed and updated from the IBM IDaaS system. Other supporting system data (such as backup data, logs, and monitoring data) cannot be restricted. Does not apply to the client software.
- **Right to Object:** Users cannot edit their individual licensing entitlement data on the license server. Authentication from the client software validates the user licensing entitlement. After a successful authentication, the user's first and last name are displayed. Users can unsubscribe subscriptions from the IBM Products and Services web site. User account-related information (including the user's IBM ID, first name, last name, etc.) can be accessed and updated from the IBM IDaaS system. Other supporting system data (such as backup data, logs, and monitoring data) cannot be restricted. Does not apply to the client software.
- **Right to Be Forgotten:** Users cannot edit their individual licensing entitlement data on the license server. Authentication from the client software validates the user licensing entitlement. After a successful authentication, the user's first and last name are displayed. Users can unsubscribe subscriptions from the IBM Products and Services web site. User account-related information (including the user's IBM ID, first name, last name, etc.) can be accessed and updated from the IBM IDaaS system. Other supporting system data (such as backup data, logs, and monitoring data) cannot be restricted. Does not apply to the client software.
- **Right to Data Portability:** Users cannot edit their individual licensing entitlement data on the license server. Authentication from the client software validates the user licensing entitlement. After a successful authentication, the user's first and last name are displayed. Users can unsubscribe subscriptions from the IBM Products and Services web site. User account-related information (including the user's IBM ID, first name, last name, etc.) can be accessed and updated from the IBM IDaaS system. Other supporting system data (such as backup data, logs, and monitoring data) cannot be restricted. Does not apply to the client software.

# Accessibility features in Watson Studio Desktop content and documentation

---

IBM is committed to accessibility. Accessibility features that follow compliance guidelines are included in Watson Studio Desktop content and documentation to benefit users with disabilities. Parts of the user interface of Watson Studio Desktop are accessible, but not entirely. Only documentation is compliant, with a subset of parts of the overall product.

Watson Studio Desktop documentation uses the latest W3C Standard, [WAI-ARIA 1.0](#), to ensure compliance with the [United States Access Board Section 508 Standards](#) and the [Web Content Accessibility Guidelines \(WCAG\) 2.0](#).

The Watson Studio Desktop online product documentation is enabled for accessibility. Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use information technology products successfully. Documentation is provided in HTML so that it is easily accessible through assistive technology. With the accessibility features of Watson Studio Desktop, you can perform the following tasks:

- Use screen-reader software and digital speech synthesizers to hear what is displayed on the screen. Consult the product documentation of the assistive technology for details about using assistive technologies with HTML-based information.
- Use screen magnifiers to magnify what's displayed on the screen.
- Operate specific or equivalent features by using only the keyboard.

For more information about the commitment that IBM has to accessibility, see [IBM Accessibility](#).

## TTY service

---

In addition to standard IBM help desk and support web sites, IBM has established a TTY telephone service for use by deaf or hard of hearing customers to access sales and support services:

800-IBM-3383 (800-426-3383) within North America

## Additional interface information

---

The Watson Studio Desktop user interfaces do not have content that flashes 2 - 55 times per second.

The Watson Studio Desktop web user interfaces rely on cascading stylesheets to render content properly and to provide a usable experience. If you are a low-vision user, you can adjust your operating system display settings, and use settings such as high contrast mode. You can control font size by using the device or web browser settings.

**Parent topic:** [Overview of Watson Studio Desktop](#)

## Notices

---

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
US*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan Ltd.  
19-21, Nihonbashi-Hakozakicho, Chuo-ku  
Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

This requirement arises from the IBM CDPI (Clearly Differentiated Programming Interfaces) policy. It is a function of general antitrust law and therefore still relevant. To avoid antitrust claims, especially with respect to products where IBM might have large market share, IBM avoids using secret interfaces for one IBM software product to connect to another (unless there is an equivalent public way for other party's products to interact with our product).

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
US*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

**Parent topic:** [Overview of Watson Studio Desktop](#)

## Trademarks

---

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

## Terms and conditions for product documentation

---

Permissions for the use of these publications are granted subject to the following terms and conditions.

### Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

### Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

### Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

### Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

## Creative Commons

---

The Offering includes some or all of the following that IBM provides under the [Creative Commons Attribution 4.0 International license](#):

- **Package name:** caniuse-lite(node) 1.0.30000835

The Offering includes some or all of the following that IBM provides under the [Creative Commons ShareAlike 3.0 License](#):

- **Package name:** QT 5.9.7

## Getting started

---

Get started quickly with Watson Studio Desktop.

1. Install Watson Studio Desktop. The steps depend on which offering you're installing. If you plan to use notebooks, you must also install the environments for notebooks.
  - [Watson Studio Desktop Subscription](#)
  - [Watson Studio Desktop 1.1](#)
  - [Watson Studio Desktop 2.0](#)

2. [Create a project](#). A project is how you organize your data and resources to achieve a goal. Project resources can include Data Refinery flows, connections, and analytic assets like notebooks and SPSS Modeler flows.

You can create a new empty project or import a project that was previously exported from Watson Studio Desktop. Watson Studio Desktop also includes an [example project](#) with many preloaded SPSS Modeler flows and their associated data sets to help you learn about the application.

3. [Add data](#) to the project.
4. Depending on the state of your source data, you might need to [use Data Refinery to cleanse and shape the data](#).
5. Use analytic tools to analyze your data and solve your business problems:
  - [SPSS Modeler flows](#): Develop predictive models and deploy them into business operations.
  - [Notebooks](#): Run small pieces of code that process your data. Immediately view the results of your computation.
6. If you are using Watson Studio Desktop 1.1 or Watson Studio Desktop 2.0 and you connect to a [Watson Machine Learning Server](#) (available as a separate offering), you can use the server's processing power to run model flows or build and train machine learning models using AutoAI. You can also use Watson Machine Learning Server to configure, monitor, and deploy the model.

- [Watson Studio overview](#)

## Installation and setup (Subscription)

---

Let's get started with the installation of Watson Studio Desktop Subscription.

Note: If updates are available after installation, you can select to check for updates or wait until you are prompted.

- **System requirements (Subscription)**  
Ensure that your computer meets the minimum hardware and software requirements for the installation and deployment of Watson Studio Desktop Subscription.
- **Purchasing or starting a trial of Watson Studio Desktop (Subscription)**  
With Watson Studio Desktop Subscription, you have no worries about software licenses or version updates. You can download application updates regularly.
- **Installing Watson Studio Desktop (Subscription)**  
Install Watson Studio Desktop Subscription on Windows or macOS.
- **Installing the environments for notebooks (Subscription)**  
To start using notebooks you first have to install the environments that you need to create and run your notebooks in Watson Studio Desktop.
- **Managing add-ons (Subscription)**  
Install or uninstall add-ons from the Add-ons and services page.
- **Locating files in Watson Studio Desktop (Subscription)**  
By default, Watson Studio Desktop stores files in the following locations.
- **Checking for updates (Subscription)**  
By default, IBM Watson Studio automatically informs you whenever new updates or fixes are available.
- **Uninstalling Watson Studio Desktop (Subscription)**  
If you decide to uninstall IBM Watson Studio Desktop Subscription, your files are not deleted, but you cannot access them anymore.

## System requirements (Subscription)

---

Ensure that your computer meets the minimum hardware and software requirements for the installation and deployment of Watson Studio Desktop Subscription.

These system requirements are for installing the Core package and all add-ons.

- [Microsoft Windows](#)
- [macOS](#)

## Microsoft Windows

---

Table 1. Minimum system requirements for running on Microsoft Windows

Requirement	Details
Operating system Microsoft Windows 64 bits	<ul style="list-style-type: none"><li>• Windows 10 Professional</li><li>• Windows 10 Enterprise</li></ul>
Computer and processor	<ul style="list-style-type: none"><li>• 1.6 gigahertz (GHz) or faster</li><li>• 2-core or more</li></ul>
Memory	<ul style="list-style-type: none"><li>• 8 GB of RAM</li></ul>
Display	<ul style="list-style-type: none"><li>• 1280 x 720 or higher resolution</li></ul>

Requirement	Details
Free disk space required for core package	<ul style="list-style-type: none"> <li>5 GB</li> </ul>
Notebooks add-on and Notebook environments (Miniconda3)	<ul style="list-style-type: none"> <li>8.1 GB</li> </ul>
SPSS Modeler add-on	<ul style="list-style-type: none"> <li>1.8 GB</li> </ul>
Free disk space required for installation core package and all add-ons	<ul style="list-style-type: none"> <li>19.1 GB</li> </ul>
Final installation size of core package and all add-ons	<ul style="list-style-type: none"> <li>15.1 GB</li> </ul>

## macOS

Table 2. Minimum system requirements for running on macOS

Requirement	Details
Operating system macOS	<ul style="list-style-type: none"> <li>Catalina 10.15</li> <li>Mojave 10.14</li> </ul>
Supported file systems	<ul style="list-style-type: none"> <li>Apple File System (APFS)</li> <li>Mac OS Extended (HFS+)</li> </ul>
Computer and processor	<ul style="list-style-type: none"> <li>1.6 gigahertz (GHz) or faster</li> <li>2-core or more</li> </ul>
Memory	<ul style="list-style-type: none"> <li>8 GB of RAM</li> </ul>
Display	<ul style="list-style-type: none"> <li>1280 x 720 or higher resolution</li> </ul>
Free disk space required for core package	<ul style="list-style-type: none"> <li>5 GB</li> </ul>
Notebooks add-on and Notebook environments (Miniconda3)	<ul style="list-style-type: none"> <li>8.1 GB</li> </ul>
SPSS Modeler add-on	<ul style="list-style-type: none"> <li>1.8 GB</li> </ul>
Free disk space required for installation core package and all add-ons	<ul style="list-style-type: none"> <li>20.1 GB</li> </ul>
Final installation size of core package and all add-ons	<ul style="list-style-type: none"> <li>15.2 GB</li> </ul>

## Next steps

- [Installing Watson Studio Desktop \(Subscription\)](#)
- [Installing the environments for notebooks \(Subscription\)](#)

**Parent topic:** [Installation and setup \(Subscription\)](#)

## Purchasing or starting a trial of Watson Studio Desktop (Subscription)

With Watson Studio Desktop Subscription, you have no worries about software licenses or version updates. You can download application updates regularly.

Because Watson Studio Desktop Subscription doesn't rely on a traditional licensing model, you are provided more flexibility with your purchasing options. You can purchase the software as a monthly subscription without any term commitments. The simplified packaging makes purchasing the right options to suit your needs a breeze.

The following steps provide guidance on purchasing Watson Studio Desktop Subscription.

1. Go to [IBM Watson Studio Desktop: Pricing](#). You are presented with options for downloading the trial version or purchasing Watson Studio Desktop Subscription. Students can click Get student edition for an academic trial.
2. Click Free trial or Purchase now.
3. When you purchase the subscription offering, you're presented with options for purchasing multiple subscription licenses and the subscription term. Select the desired number of authorized users and choose monthly auto-renewal or the 12-month term.
4. Review the total price and click Continue to checkout.
5. Sign in to your IBM account with your IBMid. Enter your IBMid credentials, or click Create an account to create a new IBM account. You are presented with a check-out step that requires you to enter your contact and billing information.
6. Review your contact information, review your order details, and then click Submit your order. You are provided an order number and sent an order confirmation email.
7. You are also sent a second email that requires you to validate your email address. Clicking Validate Email Address in the email sends you to the IBM Products and Services portal, which is explained in [Managing your subscription to Watson Studio Desktop](#).
8. See [Installing Watson Studio Desktop \(Subscription\)](#) to download and install the product.

After you download and install the product, you are required to log in with your IBMid credentials to activate your subscription.

- **Managing your subscription to Watson Studio Desktop**

Use the My IBM website to manage IBM product purchases, subscriptions, users, and services.

- **Logging in and downloading updates (Subscription)**

When you first open IBM Watson Studio, you're prompted to log in with your IBMid. Your IBMid provides access to IBM applications, services, communities, support channels, online purchasing, and more.

**Parent topic:** [Installation and setup \(Subscription\)](#)

## Managing your subscription to Watson Studio Desktop

---

Use the My IBM website to manage IBM product purchases, subscriptions, users, and services.

1. Go to <https://myibm.ibm.com/dashboard/>, and log in with your IBM account (IBMid). The Products page displays a list of IBM products to which you are entitled.
2. Go to the Watson Studio Desktop tile.
  - Click Download to download the product.
  - Click Manage to view the product management options:

### Overview

The Overview section provides links to upgrade or change your plan, a link to cancel the subscription at the end of the current billing cycle, and the following subscription plan details:

#### Owner

Displays the product subscription owner name and associated email address.

#### Plan

Identifies the product subscription plan.

#### Number of authorized users

Lists the total number of authorized users, including the current number of assigned users, and available subscriptions.

#### Billing frequency

Identifies how the product subscription is billed (monthly, yearly).

#### Subscription start date

Identifies the subscription start date.

#### Subscription ID

Provides the unique subscription ID number.



## Downloads

The Downloads section displays all the IBM products and services to which you are entitled. The available downloads are sorted by their respective operating system.

The Related Downloads section provides links to downloads that support the main product.

## Manage devices

The Manage devices section lists the device upon which the subscribed product is installed and active. The information includes the computer name, operating system, subscription activation date.

The Action column includes a Deactivate link that deactivates the product subscription for the associated computer name. Use this feature to install and activate a subscribed product on another computer. For example, you can deactivate a subscription for your desktop computer, and then activate the subscription on your laptop computer. However, you will typically deactivate a managed device by exiting and logging out of Watson Studio Desktop from the managed device. With the Deactivate link, you can remotely deactivate a device (for example, when you forget to log out of Watson Studio Desktop from a managed device).

## Manage users

The Manage users section provides options for administrators to manage Watson Studio Desktop users. Administrators can add new users and edit existing users.

### Adding new users

The dialogue displays the current number of assigned subscriptions alongside the remaining number of available subscriptions.

1. Click the drop-down arrow next to Add Users. The Add a single user dialog allows administrators to define the new user name, email address, IBMid, and role (Administrator, license user, or both).
2. Click Add user.

When the administrator enters a user email address, and the new user does not have an IBMid, the new user is sent an email message that provides instructions on creating an IBMid. After the new user creates an IBMid, they use their IBMid credentials to log in to Watson Studio Desktop.

It might take some time for newly added users to display in the Manage users section.

### Searching for and filtering existing users

The Manage users section provides search and filter options that allow administrators to search for existing users and filter the results based on the user role, and status.

Use the Search and Filter fields to define the search criteria.

### Managing existing users

The user record displays the current number of assigned subscriptions alongside the remaining number of available subscriptions.

1. Click the down arrow next to an existing user record. The record expands to provide the following options:
  - Name: This non-modifiable field displays the user name and email address.
  - Role: Provides options for defining the user roles.
  - Status: Displays the user status options. Administrators can activate or remove users.
2. Click Save to apply the changes.

## Assign alias

Use the Assign alias section to assign an alias to the official product name that is displayed on the Products page. An alias can help differentiate between different product versions. The alias appears beneath the official product name.

## Product support

The Product support section provides options for contacting IBM support through forums, support tickets, telephone, or email.

**Parent topic:** [Purchasing or starting a trial of Watson Studio Desktop \(Subscription\)](#)

## Logging in and downloading updates (Subscription)

---

When you first open IBM Watson Studio, you're prompted to log in with your IBMid. Your IBMid provides access to IBM applications, services, communities, support channels, online purchasing, and more.

If you don't yet have an IBMid, follow the online instructions.

After logging in, you can download and install a trial or purchase the full version of IBM Watson Studio.

For details about purchasing and subscribing, see [Purchasing or starting a trial of Watson Studio Desktop \(Subscription\)](#). For details about managing an existing subscription, see [Managing your subscription to Watson Studio Desktop](#).

**Parent topic:** [Purchasing or starting a trial of Watson Studio Desktop \(Subscription\)](#)

## Installing Watson Studio Desktop (Subscription)

---

Install Watson Studio Desktop Subscription on Windows or macOS.

When you install Watson Studio Desktop Subscription, you install the Core package and optionally install add-ons. Add-ons extend the functionality of Watson Studio Desktop Subscription.

1. Ensure that your system meets the [System requirements \(Subscription\)](#).
2. Refer to [Purchasing or starting a trial of Watson Studio Desktop \(Subscription\)](#) for information about downloading and subscribing to Watson Studio Desktop.
3. Follow the installation steps for your operating system:
  - [Installing on Windows \(Subscription\)](#)
  - [Installing on macOS \(Subscription\)](#)

Note: The first time that you start Watson Studio Desktop Subscription after installation, you must log in with your IBMid credentials to activate your subscription.

4. If you want to create or run notebooks, refer to [Installing the environments for notebooks \(Subscription\)](#).
- [Installing on Windows \(Subscription\)](#)  
Install Watson Studio Desktop Subscription on Windows.
  - [Installing on macOS \(Subscription\)](#)  
Install IBM Watson Studio Subscription on macOS.

**Parent topic:** [Installation and setup \(Subscription\)](#)

## Installing on Windows (Subscription)

---

Install Watson Studio Desktop Subscription on Windows.

### Prerequisites

---

- Ensure that your system meets the [System requirements \(Subscription\)](#).
- [Download](#) IBM Watson Studio Subscription.
- Enable User Account Control (UAC) and Administrator mode. You must use "Run as Administrator" not only for installing the Core package, but also for installing add-ons and future automatic downloads and installations.
- Make sure that you are connected to the internet.

### Installation steps

---

The installation steps differ for an administrator versus a user who is not an administrator. Depending on how privileges are set up on your computer, you might need to install as a user (not the administrator) even though you're

listed as an Administrator of your computer.

1. Browse to the saved .exe file location.
2. Right-click the .exe file and select Run as administrator.
3. At the Completing IBM Watson Studio Setup screen, do one of the following actions:
  - If you are NOT an administrator, deselect Run IBM Watson Studio.
  - If you are installing as the administrator, select Run IBM Watson Studio. (Default)
4. Click Finish.
5. If you are NOT installing as an administrator, right-click the IBM Watson Studio icon on the desktop, and select Run as administrator.
6. Read and agree to the Terms, and then click Continue. The Download & installation options page opens.

You must install the Core package. You can install the Modeler and Notebook packages now or later. See [Managing add-ons \(Subscription\)](#). White-outlined circles with a number of MB below the percentage show the download progress. Blue-outlined circles show the installation progress.

By default, IBM Watson Studio opens automatically after the Core package is installed and the Modeler and Notebooks add-on packages are either installed or skipped.

If you install the Notebooks add-on, the first time you open a notebook or create a notebook, you will be prompted to [install the notebook environments](#) if they are not already present on your computer.

7. Log in to IBM Watson Studio with your IBMid credentials. If you see a pop-up window that asks you to accept incoming network connections, click Accept.

Depending on your local setup, you are prompted to grant permissions or you might have to provide [network access and other permissions](#) to complete the installation.

The program, IBM Watson Studio Desktop Subscription, is installed by default in this path: C:\Program Files\IBM Watson Studio\IBM Watson Studio.exe.

## Next step

---

[Create a project](#)

- **[Network access or other permissions for Windows \(Subscription\)](#)**  
Depending on your local setup, you might have to provide network access or other permissions to complete the installation on Windows. Watson Studio requires the privilege to set up a local `http` server and establish network connections.

**Parent topic:** [Installing Watson Studio Desktop \(Subscription\)](#)

## Network access or other permissions for Windows (Subscription)

---

Depending on your local setup, you might have to provide network access or other permissions to complete the installation on Windows. Watson Studio requires the privilege to set up a local `http` server and establish network connections.

### General installation - allowlist

---

To ensure a successful installation, the firewall (for example, Windows Defender Firewall) of your local machine must allow connections for the following applications:

IBM Watson Studio.exe

By default, located in the following Windows path:

C:\Program Files\IBM Watson Studio\IBM Watson Studio.exe

nginx.exe

By default, located in the following Windows path:

C:\Program Files\IBM Watson Studio\resources\application\core\nginx\nginx.exe

Rscript.exe

By default, located in the following Windows path:

C:\Program Files\IBM Watson Studio\resources\application\runtimes\r\R-3.5.1\bin\Rscript.exe

or

C:\Program Files\IBM Watson Studio\resources\application\runtimes\r\R-3.5.1\bin\x64\Rscript.exe

java.exe

By default, located in the following Windows path:

C:\Program Files\IBM Watson Studio Desktop\resources\application\runtimes\modeler\server\jre\bin

node.exe

By default, located in the following Windows path:

C:\Program Files\IBM Watson Studio\resources\application\dependencies\node\node.exe

## Notebook environments - allowlist

---

To ensure a successful environments installation, the firewall of your local machine must allow connection privileges for the following applications in Windows:

jupyter.exe

By default, located in the following Windows path:

C:\ProgramData\WatsonStudioDesktop\miniconda3\envs\desktop\Scripts\jupyter.exe

python.exe

By default, located in the following Windows path:

C:\ProgramData\WatsonStudioDesktop\miniconda3\envs\desktop\python.exe

jupyter-notebook.exe

By default, located in the following Windows path:

C:\ProgramData\WatsonStudioDesktop\miniconda3\envs\desktop\Scripts\jupyter-notebook.exe

**Parent topic:** [Installing on Windows \(Subscription\)](#)

## Installing on macOS (Subscription)

---

Install IBM Watson Studio Subscription on macOS.

### Prerequisites

---

- Ensure that your system meets the [System requirements \(Subscription\)](#).
- [Download](#) IBM Watson Studio Subscription.
- Make sure that you are connected to the internet.

### Installation steps

---

1. Double-click the .dmg file.
2. In the pop-up window, drag the IBM Watson Studio icon to the Applications folder and close the window.
3. Double-click IBM Watson Studio in the Applications folder or use Spotlight to open the installation.
4. Read and agree to the Terms, and then click Continue. The Download & installation options page opens.

You must install the Core package. You can install the Modeler and Notebook packages now or later. See [Managing add-ons \(Subscription\)](#). White-outlined circles with a number of MB below the percentage show the

download progress. Blue-outlined circles show the installation progress.

By default, IBM Watson Studio opens automatically after the Core package is installed and the Modeler and Notebooks add-on packages are either installed or skipped.

If you install the Notebooks add-on, the first time you open a notebook or create a notebook, you will be prompted to [install the notebook environments](#) if they are not already present on your computer.

5. Log in to IBM Watson Studio with your IBMid credentials. If you see a pop-up window that asks you to accept incoming network connections, click Accept.

The program, IBM Watson Studio Desktop Subscription, is installed by default in this path: /Applications/IBM Watson Studio.app.

## Next step

---

[Create a project](#)

**Parent topic:** [Installing Watson Studio Desktop \(Subscription\)](#)

## Installing the environments for notebooks (Subscription)

---

To start using notebooks you first have to install the environments that you need to create and run your notebooks in Watson Studio Desktop.

The standard Python 3.6 libraries are installed to your computer:

- 3 GB of free disk space is required to install the Python environment.
- By default, Miniconda3-4.5.12 is installed with Python 3. For details on the installation path refer to [Notebook environments](#).

The first time you add a notebook to a project, you are prompted to install the notebook environments. Connect to the internet and follow the instructions to finish your installation.

If you're using a Windows computer, make sure you run the installation as an administrator. If you're installing for a user other than the administrator, close the application, right-click the IBM Watson Studio icon on your desktop, and choose Run as administrator. After installation, restart the application so that the Notebooks metadata matches your login.

Depending on your local setup, you might have to [provide network access or other permissions to complete the installation](#).

Note: The installation can take from 15 to 35 minutes or longer. The installation time is also affected by the speed of your network connection. Stay on this page until the installation is complete. When the installation completes, you can close the installation window and start creating notebooks.

If you don't stay on the current page, the installation of environments is canceled.

For more information on notebooks, see [Notebooks](#).

## Uninstalling the notebook environments

---

To uninstall the notebook environment files, remove the miniconda3 directory:

Windows

\ProgramData\WatsonStudioDesktop\miniconda3

macOS

/Users/username/WatsonStudioDesktop/miniconda3

**Parent topic:** [Installation and setup \(Subscription\)](#)

## Managing add-ons (Subscription)

---

Install or uninstall add-ons from the Add-ons and services page.

### Prerequisites

---

- Make sure you are connected to the internet.
- For Windows, you must install or uninstall add-ons as an administrator.

### Install or uninstall an add-on

---

1. Expand the side IBM Watson Studio pane.
2. Click Add-ons and services.
3. Select an add-on to install or uninstall.
  - You can install or uninstall only one add-on at a time.
  - You must remain on the Add-ons and services page until the installation or uninstall completes.
4. Restart the application for the change to take effect.

If you install the Notebooks add-on, the first time you open a notebook or create a notebook, you will be prompted to [install the notebook environments](#) if they are not already present on your computer.

### Restrictions when add-ons are not installed

---

All the assets that are in your project from a previous installation or that you imported remain in your project. However, if you choose not to install an add-on, be aware of the following restrictions:

- You will not be able to use the asset associated with the add-on (Modeler flow or Notebook).
- On the Projects page, if you click an asset (Modeler flow or Notebook), you will be directed to the Add-ons and services page to install it.
- The associated asset will not show up in the AVAILABLE ASSET TYPES list in the Add to project page.

**Parent topic:** [Installation and setup \(Subscription\)](#)

## Locating files in Watson Studio Desktop (Subscription)

---

By default, Watson Studio Desktop stores files in the following locations.

### Root installation folder

---

By default, files are stored in the following paths:

Windows

C:\Program Files\IBM Watson Studio

macOS

/Applications/IBM Watson Studio.app

### Default folder location

---

By default, files are stored in the following paths:

Windows

C:\Users\username\AppData\Roaming\IBM Watson Studio

macOS

/Users/username/Library/Application Support/IBM Watson Studio

## Project assets

---

By default, project assets are stored in the following paths:

Windows

C:\Users\username\AppData\Roaming\IBM Watson Studio\projects\

macOS

/Users/username/Library/Application Support/IBM Watson Studio/projects/

## Notebook files in projects

---

By default, notebooks that are created in your projects are stored in the following paths:

Windows

C:\Users\username\AppData\Roaming\IBM Watson Studio\projects\project-name\assets\notebook

macOS

/Users/username/Library/Application Support/IBM Watson Studio/projects/project-name/assets/notebook

## Notebook environments

---

By default, Miniconda3-4.5.12 is installed with Python 3. The default Python virtual environment is installed in the following paths:

Windows

C:\ProgramData\WatsonStudioDesktop\miniconda3\

macOS

/Users/username/WatsonStudioDesktop/miniconda3/

To use the Conda installation (Python runtime) you must be able to access:

<https://repo.anaconda.com/pkg/pro/noarch/repojson.json.bz2>

## Log files

---

By default, log files are stored in the following paths:

Windows

C:\Users\username\AppData\Roaming\IBM Watson Studio\logs\

macOS

/Users/username/Library/Application Support/IBM Watson Studio/logs/

**Parent topic:** [Installation and setup \(Subscription\)](#)

## Checking for updates (Subscription)

---

By default, IBM Watson Studio automatically informs you whenever new updates or fixes are available.

Click the question mark icon and select Versions or select the Check for Updates option from the IBM Watson Studio menu and choose to:

- Install the updates or fixes immediately
- Automatically download and install updates in the future
- Skip this update

Note: After you update IBM Watson Studio, you cannot downgrade it to a previous version of IBM Watson Studio.

**Parent topic:** [Installation and setup \(Subscription\)](#)

## Uninstalling Watson Studio Desktop (Subscription)

---

If you decide to uninstall IBM Watson Studio Desktop Subscription, your files are not deleted, but you cannot access them anymore.

The name of the Watson Studio Desktop Subscription application is *IBM Watson Studio*.

### Uninstall IBM Watson Studio Desktop Subscription on Windows

---

1. Sign in as an administrator.
2. Go to the Control Panel > Programs and Features and uninstall the IBM Watson Studio program.

Files that are installed under C:\Users\username\AppData\Roaming\IBM Watson Studio are not removed. These files include the project files and other metadata.

### Uninstall IBM Watson Studio Desktop Subscription on macOS

---

Select Finder > Applications and right-click the IBM Watson Studio.app and select Move to Trash.

Files that are installed under /Users/username/Library/Application Support/IBM Watson Studio are not removed. These files include the project files and other metadata.

### Cancel your subscription

---

If you uninstall IBM Watson Studio Desktop Subscription, the subscription is not automatically canceled. To cancel your subscription plan, log in to [My IBM account](#). On the Products page, click the Manage button next to your SPSS Modeler Subscription listing. Then, click the Cancel plan link on the Overview page.

### Uninstall an add-on

---

If you want to keep IBM Watson Studio Desktop Subscription on your computer, but remove an add-on, you can uninstall the add-on.

1. Expand the side IBM Watson Studio pane.
2. Click Add-ons and services.
3. Select an add-on to install or uninstall.
  - You can install or uninstall only one add-on at a time.
  - You must remain on the Add-ons and services page until the installation or uninstall completes.
4. Restart the application for the change to take effect.

**Parent topic:** [Installation and setup \(Subscription\)](#)

## Installation and setup (Version 2.0)

---

Let's get started with the installation of Watson Studio Desktop 2.0.

- **[System requirements \(Version 2.0\)](#)**  
Ensure that your computer meets the minimum hardware and software requirements for the installation and deployment of Watson Studio Desktop 2.0.
- **[Installing Watson Studio Desktop 2.0](#)**  
Install IBM Watson Studio Desktop 2.0 on Windows or macOS.
- **[Installing the environments for notebooks \(Version 2.0\)](#)**  
To start using notebooks, you first must install the environments that you need to create and run your notebooks in Watson Studio Desktop.



- [Managing add-ons \(Version 2.0\)](#)  
Install or uninstall add-ons from the Add-ons and services page.
- [Locating files \(Version 2.0\)](#)  
By default, Watson Studio Desktop 2.0 stores files in the following locations.
- [Uninstalling on Windows \(Version 2.0\)](#)  
Uninstall IBM Watson Studio Desktop 2.0 on Windows.
- [Uninstalling on macOS \(Version 2.0\)](#)  
Uninstall IBM Watson Studio Desktop 2.0 on macOS.
- [Updating from a previous Watson Studio Desktop version](#)  
To update a previous Watson Studio Desktop version, for example Watson Studio Desktop 1.1 to Watson Studio Desktop 2.0, uninstall the previous version and install the new version.
- [Connecting to IBM Watson Machine Learning Server](#)

## System requirements (Version 2.0)

Ensure that your computer meets the minimum hardware and software requirements for the installation and deployment of Watson Studio Desktop 2.0.

- [Microsoft Windows](#)
- [macOS](#)

### Microsoft Windows

Table 1. Minimum system requirements for running on Microsoft Windows

Requirement	Details
Operating system Microsoft Windows 64 bits	<ul style="list-style-type: none"> <li>• Windows 10 Professional</li> <li>• Windows 10 Enterprise</li> </ul>
Computer and processor	<ul style="list-style-type: none"> <li>• 1.6 gigahertz (GHz) or faster</li> <li>• 4-core or more</li> </ul>
Memory	<ul style="list-style-type: none"> <li>• 16 GB of RAM</li> </ul>
Display	<ul style="list-style-type: none"> <li>• 1280 x 720 or higher resolution</li> </ul>
Free disk space required for core package	<ul style="list-style-type: none"> <li>• 6.4 GB</li> </ul>
Notebooks add-on and Notebook environments (Miniconda3)	<ul style="list-style-type: none"> <li>• 8.99 GB</li> </ul>
SPSS Modeler add-on	<ul style="list-style-type: none"> <li>• 1.73 GB</li> </ul>
Free disk space required for installation core package and all add-ons	<ul style="list-style-type: none"> <li>• 17 GB</li> </ul>
Final installation size of core package and all add-ons	<ul style="list-style-type: none"> <li>• Approximately 12 GB</li> </ul>

### macOS

Table 2. Minimum system requirements for running on macOS

Requirement	Details
Operating system macOS	<ul style="list-style-type: none"> <li>• Catalina 10.15</li> <li>• Mojave 10.14</li> </ul>
Supported file systems	<ul style="list-style-type: none"> <li>• Apple File System (APFS)</li> <li>• Mac OS Extended (HFS+)</li> </ul>

Requirement	Details
Computer and processor	<ul style="list-style-type: none"> <li>1.6 gigahertz (GHz) or faster</li> <li>2-core or more</li> </ul>
Memory	<ul style="list-style-type: none"> <li>16 GB of RAM</li> </ul>
Display	<ul style="list-style-type: none"> <li>1280 x 720 or higher resolution</li> </ul>
Free disk space required for core package	<ul style="list-style-type: none"> <li>6.8 GB</li> </ul>
Notebooks add-on and Notebook environments (Miniconda3)	<ul style="list-style-type: none"> <li>9.81 GB</li> </ul>
SPSS Modeler add-on	<ul style="list-style-type: none"> <li>1.86 GB</li> </ul>
Free disk space required for installation core package and all add-ons	<ul style="list-style-type: none"> <li>17 GB</li> </ul>
Final installation size of core package and all add-ons	<ul style="list-style-type: none"> <li>Approximately 12 GB</li> </ul>

## Next steps

- [Installing Watson Studio Desktop 2.0](#)
- [Installing the environments for notebooks \(Version 2.0\)](#)

**Parent topic:** [Installation and setup \(Version 2.0\)](#)

## Installing Watson Studio Desktop 2.0

Install IBM Watson Studio Desktop 2.0 on Windows or macOS.

1. Ensure that your system meets the [System requirements \(Version 2.0\)](#).
  2. Download IBM Watson Studio Desktop from Passport Advantage. See [this download document](#) for details, including the Passport Advantage part numbers (in a table at the end of the document).
  3. Choose the steps for your operating system:
    - [Installing on Windows \(Version 2.0\)](#)
    - [Installing on macOS \(Version 2.0\)](#)
  4. If you want to create or run notebooks, refer to [Installing the environments for notebooks \(Version 2.0\)](#).
- **[Installing on Windows \(Version 2.0\)](#)**  
Install IBM Watson Studio Desktop 2.0 on Windows.
  - **[Installing on macOS \(Version 2.0\)](#)**  
Install IBM Watson Studio Desktop 2.0 on macOS.

**Parent topic:** [Installation and setup \(Version 2.0\)](#)

## Installing on Windows (Version 2.0)

Install IBM Watson Studio Desktop 2.0 on Windows.

Before you install IBM Watson Studio Desktop on Windows, determine which kind of installation you want to do. You can install only one instance of Watson Studio on your computer.

- **[Per-user installation:](#)** This installation is for a single user using Watson Studio on the computer. Use this installation type if you are not an administrator for the computer. This is the default type for new installations.
- **[Per-machine \(all users\) installation:](#)** This installation is for multiple users using Watson Studio on the computer. Users' project files are stored in individual folders. You must be the administrator to do this installation, to start

it the first time, and to install add-ons later.

## Per-user installation

---

### Prerequisites for per-user installation

- Ensure that your computer meets the [System requirements \(Version 2.0\)](#).
- It is recommended that you be member of the Administrators group. Alternatively, you'll be required to accept User Account Control prompts to allow the installation to continue.
- If you are connecting to a Watson Machine Learning Server, ensure that the version of Watson Studio Desktop and Watson Machine Learning Server are the same. (Watson Studio Desktop cannot access a Watson Machine Learning Server with a different version number.)
- Only one instance of Watson Studio can be installed on a computer.
  - If you have a *previous* version of IBM Watson Studio Desktop installed or a beta version installed, [uninstall](#) it.
  - If you have Watson Studio Desktop Subscription installed, [uninstall](#) it.

### Installation steps for per-user installation

1. Open the directory where you saved the .ISO file.
2. Double-click the .ISO file to access the .exe file and the .7z file.
3. Double-click the .exe file.
4. When prompted for Who should this application be installed for?, select Only for me.
5. At the Completing IBM Watson Studio Setup screen, select Run IBM Watson Studio
6. Click Finish.
7. Select to launch IBM Watson Studio Desktop and click Finish.
8. Read and agree to the Terms and the Privacy Policy, and then click Continue. The Packages installation page opens.

You must install the Core package. You can install the Modeler and Notebook packages now or later. See [Managing add-ons \(Version 2.0\)](#).

By default, IBM Watson Studio Desktop opens automatically after the Core package is installed and the Modeler and Notebooks add-on packages are either installed or skipped.

If you install the Notebooks add-on, the first time you open a notebook or create a notebook, you will be prompted to [install the notebook environments](#) if they are not already present on your computer.

9. Eject the ISO disk image: From Windows Explorer, navigate to the IBM\_Watson\_Studio DVD Drive. Right-click the drive, and select Eject.

When you first open IBM Watson Studio Desktop and you see a pop-up window that asks you to accept incoming network connections, click Accept.

The program, IBM Watson Studio Desktop, is installed by default in this path:

C:\Users\username\AppData\Local\Programs\IBM Watson Studio Desktop\IBM Watson Studio Desktop.exe.

## Per-machine (all users) installation

---

### Prerequisites for per-machine installation

- Ensure that your system meets the [System requirements \(Version 2.0\)](#).
- Ensure that you have Administrator access to the computer.
- Enable User Account Control (UAC) and Administrator mode.
- If you are connecting to a Watson Machine Learning Server, ensure that the version of Watson Studio Desktop and Watson Machine Learning Server are the same. (Watson Studio Desktop cannot access a Watson Machine Learning Server with a different version number.)
- Only one instance of Watson Studio can be installed on a computer.

- If you have a *previous* version that is installed of IBM Watson Studio Desktop installed or a beta version installed, [uninstall](#) it.
- If you have Watson Studio Desktop Subscription installed, [uninstall](#) it.

### Installation steps for per-machine installation

1. Open the directory where you saved the .ISO file.
2. Double-click the .ISO file to access the .exe file and the .7z file.
3. Right-click the .exe file and select Run as administrator.
4. When prompted for Who should this application be installed for?, select Anyone who uses this computer (all users).
5. At the Completing IBM Watson Studio Setup screen, select Run IBM Watson Studio
6. Click Finish.
7. Read and accept the terms and conditions, and then follow the online instructions until the message Installation complete! is displayed.
8. An administrator must start IBM Watson Studio the first time. Thereafter, if you intend to use it as a different user, close and restart IBM Watson Studio so that the projects and other metadata match your login.
9. Select to launch IBM Watson Studio Desktop and click Finish.
10. Read and agree to the Terms, and then click Continue. The Packages installation page opens.

You must install the Core package. You can install the Modeler and Notebook packages now or later. If you install a package later, you must install it as an administrator. See [Managing add-ons \(Version 2.0\)](#).

By default, IBM Watson Studio Desktop opens automatically after the Core package is installed and the Modeler and Notebooks add-on packages are either installed or skipped.

If you install the Notebooks add-on, the first time you open a notebook or create a notebook, you will be prompted to [install the notebook environments](#) if they are not already present on your computer.

11. Eject the ISO disk image: From Windows Explorer, navigate to the IBM\_Watson\_Studio DVD Drive. Right-click the drive, and select Eject.

The program, IBM Watson Studio Desktop, is installed by default in this path: C:\Program Files\IBM Watson Studio Desktop\IBM Watson Studio Desktop.exe.

When you first open IBM Watson Studio Desktop and you see a pop-up window that asks you to accept incoming network connections, click Accept.

If different users intend to use IBM Watson Studio Desktop, they must start the application with their individual logins so that the projects and other metadata match each login.

## Next steps

---

- [Installing the environments for notebooks \(Version 2.0\)](#)
- [Create a project](#)
- [Network access or other permissions for Windows \(Version 2.0\)](#)

Depending on your local setup, you might have to provide network access or other permissions to complete the installation on Windows. Watson Studio requires the privilege to set up a local `http` server and establish network connections.

**Parent topic:** [Installing Watson Studio Desktop 2.0](#)

## Network access or other permissions for Windows (Version 2.0)

---

Depending on your local setup, you might have to provide network access or other permissions to complete the installation on Windows. Watson Studio requires the privilege to set up a local `http` server and establish network connections.

## General installation - allowlist

---

To ensure a successful installation, the firewall (for example, Windows Defender Firewall) of your local machine must allow connections for the following applications. Default paths are listed here.

### IBM Watson Studio Desktop.exe

- Per-user installation:  
C:\Users\username\AppData\Local\Programs\IBM Watson Studio Desktop\IBM Watson Studio Desktop.exe
- Per-machine (all users) installation:  
C:\Program Files\IBM Watson Studio Desktop\IBM Watson Studio Desktop.exe

### nginx.exe

- Per-user installation:  
C:\Users\username\AppData\Local\IBMWS\Desktop\core\nginx\nginx.exe
- Per-machine (all users) installation:  
C:\Program Files\IBM Watson Studio Desktop\resources\application\core\nginx\nginx.exe

### Rscript.exe

- Per-user installation:  
C:\Users\username\AppData\Local\IBMWS\Desktop\runtimes\r\R-3.5.1\bin\Rscript.exe  
or  
C:\Users\username\AppData\Local\IBMWS\Desktop\runtimes\r\R-3.5.1\bin\x64\Rscript.exe
- Per-machine (all users) installation:  
C:\Program Files\IBM Watson Studio Desktop\resources\application\runtimes\r\R-3.5.1\bin\Rscript.exe  
or  
C:\Program Files\IBM Watson Studio Desktop\resources\application\runtimes\r\R-3.5.1\bin\x64\Rscript.exe

### java.exe

- Per-user installation:  
C:\Users\username\AppData\Local\IBMWS\Desktop\runtimes\modeler\server\jre\bin
- Per-machine (all users) installation:  
C:\Program Files\IBM Watson Studio Desktop\resources\application\runtimes\modeler\server\jre\bin

### node.exe

- Per-user installation:  
C:\Users\username\AppData\Local\IBMWS\Desktop\dependencies\node\node.exe
- Per-machine (all users) installation:  
C:\Program Files\IBM Watson Studio Desktop\resources\application\dependencies\node\node.exe

### Modeler service

- Per-user installation:  
C:\Users\username\AppData\Local\IBMWS\Desktop\runtimes\modeler\modelerserver.exe
- Per-machine (all users) installation:  
C:\Program Files\IBM Watson Studio Desktop\resources\application\runtimes\modeler\modelerserver.exe

## Notebook environments - allowlist

---

To ensure a successful environments installation, the firewall of your local machine must allow connection privileges for the following applications:

jupyter.exe

- Per-user installation:  
C:\Users\username\AppData\Local\IBMWS\miniconda3\envs\desktop\Scripts\jupyter.exe
- Per-machine (all users) installation:  
C:\ProgramData\WatsonStudioDesktop\miniconda3\envs\desktop\Scripts\jupyter.exe

python.exe

- Per-user installation:  
C:\Users\username\AppData\Local\IBMWS\miniconda3\envs\desktop\python.exe
- Per-machine (all users) installation:  
C:\ProgramData\WatsonStudioDesktop\miniconda3\envs\desktop\python.exe

jupyter-notebook.exe

- Per-user installation:  
C:\Users\username\AppData\Local\IBMWS\miniconda3\envs\desktop\Scripts\jupyter-notebook.exe
- Per-machine (all users) installation:  
C:\ProgramData\WatsonStudioDesktop\miniconda3\envs\desktop\Scripts\jupyter-notebook.exe

**Parent topic:** [Installing on Windows \(Version 2.0\)](#)

## Installing on macOS (Version 2.0)

---

Install IBM Watson Studio Desktop 2.0 on macOS.

### Prerequisites

---

- Ensure that your computer meets the [System requirements \(Version 2.0\)](#).
- If you are connecting to a Watson Machine Learning Server, ensure that the version of Watson Studio Desktop and Watson Machine Learning Server are the same. (Watson Studio Desktop cannot access a Watson Machine Learning Server with a different version number.)
- Only one instance of Watson Studio can be installed on a computer.
  - If you have a *previous* version of IBM Watson Studio Desktop installed or a beta version installed, [uninstall](#) it.
  - If you have Watson Studio Desktop Subscription installed, [uninstall](#) it.

### Installation steps

---

1. Double-click the wsd-2.0-macOS.dmg file to open it. This action will mount the file as a volume and will show a wsd-2.0-macOS icon on your desktop.  
Important: Do not drag the wsd-2.0-macOS.dmg file to the installpkgs folder.
2. Double-click the wsd-2.0-macOS icon to open a pop-up window that shows the content of the installation pack.
3. In the pop-up window, double-click the IBM Watson Studio Desktop.dmg file.
4. In the second pop-up window, drag the IBM Watson Studio Desktop.app icon to the Applications folder and close the window.
5. Double-click the IBM Watson Studio Desktop.app in the Applications folder, or use Spotlight to open the installation.
6. Read and agree to the Terms and the Privacy Policy, and then click Continue. The Packages installation page opens.

You must install the Core package. You can install the Modeler and Notebook packages now or later. See [Managing add-ons \(Version 2.0\)](#).

By default, IBM Watson Studio Desktop opens automatically after the Core package is installed and the Modeler and Notebooks add-on packages are either installed or skipped.

If you install the Notebooks add-on, the first time you open a notebook or create a notebook, you will be prompted to [install the notebook environments](#) if they are not already present on your computer.

7. Select to Launch IBM Watson Studio Desktop.
8. Eject the `wsd-2.0-macOS.dmg` volume on your desktop: Right-click the volume, and select Eject "`wsd-2.0-macOS`".

When you first open IBM Watson Studio Desktop and you see a pop-up window that asks you to accept incoming network connections, click Accept.

The program, IBM Watson Studio Desktop, is installed by default in this path: `/Applications/IBM Watson Studio Desktop.app`.

## Next steps

---

- [Installing the environments for notebooks \(Version 2.0\)](#)
- [Create a project](#)

**Parent topic:** [Installing Watson Studio Desktop 2.0](#)

## Installing the environments for notebooks (Version 2.0)

---

To start using notebooks, you first must install the environments that you need to create and run your notebooks in Watson Studio Desktop.

The standard Python 3.6 libraries are installed to your computer.

The first time that you add a notebook to a project, you are prompted to install the notebook environments. Similarly, when an update is available, you are prompted to install the update. Follow the instructions to finish the installation.

Note: A new installation can take from 15 to 35 minutes or longer. An update to a current installation takes only a few minutes. Stay on the page until the installation is complete. When the installation completes, you can close the installation window and start creating notebooks.

If you don't stay on the current page, the notebooks environments installation is canceled.

## Uninstalling the notebook environments

---

### Windows

- Per-user installation: To uninstall the notebook environment files, remove the `miniconda3` and the `minicondaInfo.json` directories:  
`C:\Users\username\AppData\Local\IBMWS\miniconda3`  
`C:\Users\username\AppData\Local\IBMWS\minicondaInfo.json`
- Per-machine (all users) installation. To uninstall the notebook environment files, remove the `miniconda3` directory:  
`C:\ProgramData\WatsonStudioDesktop\miniconda3`

### macOS

To uninstall the notebook environment files, remove the `WatsonStudioDesktop` directory:  
`/Users/username/WatsonStudioDesktop`

**Parent topic:** [Installation and setup \(Version 2.0\)](#)

## Managing add-ons (Version 2.0)

---

Install or uninstall add-ons from the Add-ons and services page.

### Prerequisite

---

Windows: If the installation is a "per-machine (all users)" installation, you must install or uninstall add-ons as an administrator.

### Install or uninstall an add-on

---

1. Expand the side IBM Watson Studio pane.
2. Click Add-ons and services.
3. Select an add-on to install or uninstall.
  - You can install or uninstall only one add-on at a time.
  - You must remain on the Add-ons and services page until the installation or uninstall completes.
4. Restart the application for the change to take effect.

If you install the Notebooks add-on, the first time you open a notebook or create a notebook, you will be prompted to [install the notebook environments](#) if they are not already present on your computer.

### Restrictions when add-ons are not installed

---

All the assets that are in your project from a previous installation or that you imported remain in your project. However, if you choose not to install an add-on, be aware of the following restrictions:

- You will not be able to use the asset associated with the add-on (Modeler flow or Notebook).
- On the Projects page, if you click an asset (Modeler flow or Notebook), you will be directed to the Add-ons and services page to install it.
- The associated asset will not show up in the AVAILABLE ASSET TYPES list in the Add to project page.

**Parent topic:** [Installation and setup \(Version 2.0\)](#)

## Locating files (Version 2.0)

---

By default, Watson Studio Desktop 2.0 stores files in the following locations.

### Root installation folder

---

By default, files are stored in the following paths:

Windows

- Per-user installation: C:\Users\username\AppData\Local\Programs\IBM Watson Studio Desktop\
- Per-machine (all users) installation: C:\Program Files\IBM Watson Studio Desktop\

macOS

/Applications/IBM Watson Studio Desktop.app

### Default folder location

---

By default, files are stored in the following paths:

Windows

C:\Users\username\AppData\Roaming\IBM Watson Studio



macOS  
/Users/*username*/Library/Application Support/IBM Watson Studio

## Project assets

---

By default, project assets are stored in the following paths:

Windows  
C:\Users\*username*\AppData\Roaming\IBM Watson Studio\projects\  
macOS  
/Users/*username*/Library/Application Support/IBM Watson Studio/projects/

## Notebook files in projects

---

By default, notebooks that are created in your projects are stored in the following paths:

Windows  
C:\Users\*username*\AppData\Roaming\IBM Watson Studio\projects\*project-name*\assets\notebook  
macOS  
/Users/*username*/Library/Application Support/IBM Watson Studio/projects/*project-name*/assets/notebook

## Notebook environments

---

By default, Miniconda3-4.5.12 is installed with Python 3. The default Python virtual environment is installed in the following paths:

Windows

- Per-user installation: C:\Users\*username*\AppData\Local\IBMWS\miniconda3\
- Per-machine (all users) installation: C:\ProgramData\WatsonStudioDesktop\miniconda3\

macOS  
/Users/*username*/WatsonStudioDesktop/miniconda3/

To use the Conda installation (Python runtime) you must be able to access:  
<https://repo.anaconda.com/pkg/pro/noarch/repojson.json.bz2>

## Log files

---

By default, log files are stored in the following paths:

Windows  
C:\Users\*username*\AppData\Roaming\IBM Watson Studio\logs\  
macOS  
/Users/*username*/Library/Application Support/IBM Watson Studio/logs/

**Parent topic:** [Installation and setup \(Version 2.0\)](#)

## Uninstalling on Windows (Version 2.0)

---

Uninstall IBM Watson Studio Desktop 2.0 on Windows.

**Prerequisite:** If the installation is a per-machine (all-users) installation, you must be an administrator to uninstall Watson Studio Desktop.

Important: During the uninstall, you will be asked if you want to delete the projects, data assets, and configuration-related files for your username. If you click Yes, the files will be permanently deleted. For a per-machine (all-users)

uninstall, only the files for the user who is doing the uninstall are deleted.

To uninstall Watson Studio Desktop, go to the Control Panel > Programs and Features and uninstall the IBM Watson Studio Desktop program.

**Parent topic:** [Installation and setup \(Version 2.0\)](#)

## Uninstalling on macOS (Version 2.0)

---

Uninstall IBM Watson Studio Desktop 2.0 on macOS.

There are two methods to uninstall Watson Studio Desktop from your macOS computer. Move the application to the Trash or use an uninstall script.

### Move the application to the Trash

---

Deletes all files under Applications/IBM Watson Studio.app. The project files, shared libraries (miniconda3, spark24), and artifacts are not removed. The next Watson Studio Desktop installation can use them.

Select Finder > Applications and right-click the IBM Watson Studio Desktop.app and select Move to Trash.

### Use an uninstall script

---

This method also deletes the Watson Studio Desktop files that are outside of the Applications folder.

1. Download the script file, [clean-wsd-mac.tar.gz](#), and extract its contents.
2. Select Finder > Applications. Right-click the IBM Watson Studio Desktop.app and select Move to Trash.
3. Empty the Trash.
4. Open Terminal: Select Finder > Go > Utilities. Double-click the Terminal.app.
5. Change to the directory where you downloaded and extracted the clean-wsd.sh file.

For example:

```
cd ~/Downloads
```

6. Set the execute permission for the shell script:

```
chmod +x clean-wsd.sh
```

7. Run the shell script. You have two choices:

- Delete the configuration-related files, but leave the projects metadata and data assets so that the next Watson Studio Desktop installation can use them.

```
sudo ./clean-wsd.sh
```

- Delete the configuration-related files and the projects metadata and data assets. This option completely removes the Watson Studio Desktop files from your computer.

```
sudo ./clean-wsd.sh delete-projects
```

**Parent topic:** [Installation and setup \(Version 2.0\)](#)

## Updating from a previous Watson Studio Desktop version

---

To update a previous Watson Studio Desktop version, for example Watson Studio Desktop 1.1 to Watson Studio Desktop 2.0, uninstall the previous version and install the new version.

Important: If you are connecting to IBM Watson Machine Learning Server, the Watson Machine Learning Server version and the Watson Studio Desktop version must be the same.

If you have projects that are connected to Watson Machine Learning Server 1.1 deployment spaces, you will need to update the project space associations to new deployment spaces on the Watson Machine Learning Server 2.0 instance.

1. [Uninstall Watson Studio Desktop 1.1.](#)
2. Install Watson Studio Desktop 2.0. Choose the steps for your operating system:
  - Windows: Use the same username as the previous version so that the project files and metadata remain the same. [Installing on Windows \(Version 2.0\)](#)
  - macOS: [Installing on macOS \(Version 2.0\)](#)
3. If you are using Watson Machine Learning Server, remove any existing connections to Watson Machine Learning Server 1.1:
  - a. Expand the side IBM Watson Studio pane, and click Add-ons and services.
  - b. Under Watson Machine Learning, click the three vertical dots next to the machine learning service instance, and click Remove.
4. Connect to Watson Machine Learning Server 2.0:
  - a. Expand the side IBM Watson Studio pane, and click Add-ons and services.
  - b. Click Add machine learning service.
  - c. Click Machine Learning Server.
  - d. Enter the connection details, and then click Connect.
5. Update the project space associations to new spaces on the Watson Machine Learning Server 2.0 instance. For instructions, see [Deployment spaces](#).

**Parent topic:** [Installation and setup \(Version 2.0\)](#)

## Connecting to the Watson Machine Learning Server (Version 2.0)

---

Watson Machine Learning Server is available as a separate offering for Watson Studio Desktop 1.1 or Watson Studio Desktop 2.0. With Watson Machine Learning Server, you can scale workloads for SPSS Modeler flows beyond the capabilities of a desktop compute environment and create deployments for SPSS models as well as open source frameworks.

### Prerequisite

Watson Machine Learning Server installed on your private network. The version number of Watson Studio Desktop and Watson Machine Learning Server must be the same. Installation and setup instructions for Watson Machine Learning Server are available at [Installing IBM Watson Machine Learning Server](#).

Before you can connect to Watson Machine Learning Server, a server administrator must follow the steps to:

- [Install the server](#)
- [Test the connection](#)
- [Set up user accounts](#)

After installation, the administrator must provide you with this connection information for the Watson Machine Learning Server:

- Server hostname
- Port number
- Username and password
- (Optional) A self-signed certificate to support an SSL connection with Watson Machine Learning Server.

### Creating the connection

1. Expand the side IBM Watson Studio pane, and click **Add-ons and services**.
2. Click **Add machine learning service**.
3. Click **Machine Learning Server**.

4. Enter the connection details.
5. Click **Connect**.

## Revalidating server credentials

If you re-install Watson Studio Desktop, you will have to revalidate the credentials you use to access Watson Machine Learning Server for existing connections. Open the server details, enter the username and password you use to access the server and save the credentials. After the credentials are saved, the connection will be available for use.

## Next step

[Deploying models to Watson Machine Learning Server](#)

# Installation and setup (Version 1.1)

---

Let's get started with the installation of Watson Studio Desktop 1.1.

- [System requirements \(Version 1.1\)](#)  
Ensure that your computer meets the minimum hardware and software requirements for the installation and deployment of Watson Studio Desktop 1.1.
- [Installing Watson Studio Desktop 1.1](#)  
Install IBM Watson Studio Desktop 1.1 on Windows or macOS.
- [Installing the environments for notebooks \(Version 1.1\)](#)  
To start using notebooks you first have to install the environments that you need to create and run your notebooks in Watson Studio Desktop.
- [Locating files \(Version 1.1\)](#)  
By default, Watson Studio Desktop 1.1 stores files in the following locations.
- [Uninstalling Watson Studio Desktop 1.1](#)  
If you decide to uninstall IBM Watson Studio Desktop, your files are not deleted, but you cannot access them anymore.
- [Connecting to IBM Watson Machine Learning Server](#)

## System requirements (Version 1.1)

---

Ensure that your computer meets the minimum hardware and software requirements for the installation and deployment of Watson Studio Desktop 1.1.

- [Microsoft Windows](#)
- [macOS](#)

## Microsoft Windows

---

Table 1. Minimum system requirements for running on Microsoft Windows

Requirement	Details
Operating system Microsoft Windows 64 bits	<ul style="list-style-type: none"><li>• Windows 10 Professional</li><li>• Windows 10 Enterprise</li></ul>
Computer and processor	<ul style="list-style-type: none"><li>• 1.6 gigahertz (GHz) or faster</li><li>• 2-core or more</li></ul>
Memory	<ul style="list-style-type: none"><li>• 8 GB of RAM</li></ul>
Display	<ul style="list-style-type: none"><li>• 1280 x 720 or higher resolution</li></ul>

Requirement	Details
Hard disk	<ul style="list-style-type: none"> <li>11 GB of available disk space</li> </ul>
Free disk space required for installation	<ul style="list-style-type: none"> <li>13 GB for basic installation</li> </ul>
Final installation size	<ul style="list-style-type: none"> <li>Approximately 8 GB</li> </ul>

## macOS

Table 2. Minimum system requirements for running on macOS

Requirement	Details
Operating system macOS	<ul style="list-style-type: none"> <li>Catalina 10.15</li> <li>Mojave 10.14</li> </ul>
Supported file systems	<ul style="list-style-type: none"> <li>Apple File System (APFS)</li> <li>Mac OS Extended (HFS+)</li> </ul>
Computer and processor	<ul style="list-style-type: none"> <li>1.6 gigahertz (GHz) or faster</li> <li>2-core or more</li> </ul>
Memory	<ul style="list-style-type: none"> <li>8 GB of RAM</li> </ul>
Display	<ul style="list-style-type: none"> <li>1280 x 720 or higher resolution</li> </ul>
Hard disk	<ul style="list-style-type: none"> <li>11 GB of available disk space</li> </ul>
Free disk space required for installation	<ul style="list-style-type: none"> <li>13 GB for the basic installation</li> </ul>
Final installation size	<ul style="list-style-type: none"> <li>Approximately 8 GB</li> </ul>

## Next steps

- Installing Watson Studio Desktop 1.1
- Installing the environments for notebooks (Version 1.1)

**Parent topic:** [Installation and setup \(Version 1.1\)](#)

## Installing Watson Studio Desktop 1.1

Install IBM Watson Studio Desktop 1.1 on Windows or macOS.

- Ensure that your system meets the [System requirements \(Version 1.1\)](#).
  - Download IBM Watson Studio Desktop from Passport Advantage. See [this download document](#) for details, including the Passport Advantage part numbers (in a table at the end of the document).
  - Choose the steps for your operating system:
    - [Installing on Windows \(Version 1.1\)](#)
    - [Installing on macOS \(Version 1.1\)](#)
  - If you want to create or run notebooks, refer to [Installing the environments for notebooks \(Version 1.1\)](#).
- Installing on Windows (Version 1.1)**  
Install IBM Watson Studio Desktop 1.1 on Windows.
  - Installing on macOS (Version 1.1)**  
Install IBM Watson Studio Desktop 1.1 on macOS.

**Parent topic:** [Installation and setup \(Version 1.1\)](#)

# Installing on Windows (Version 1.1)

---

Install IBM Watson Studio Desktop 1.1 on Windows.

## Prerequisites

---

- Ensure that your system meets the [System requirements \(Version 1.1\)](#).
- Enable User Account Control (UAC) and Administrator mode.
- If you have a *previous* version of IBM Watson Studio Desktop installed or a beta version installed, [uninstall](#) it before you install IBM Watson Studio Desktop 1.1.

## Installation steps

---

The installation steps differ for an administrator versus a user who is not and administrator. Depending on how privileges are set up on your computer, you might need to install as a user (not the administrator) even though you're listed as an Administrator of your computer.

1. Open the directory where you saved the .ISO file.
2. Double-click the .ISO file to access the .exe file and the .7z file.
3. Right-click the .exe file and select Run as administrator.
4. At the Completing IBM Watson Studio Setup screen, do one of the following actions:
  - If you are NOT an administrator, deselect Run IBM Watson Studio.
  - If you are installing as the administrator, select Run IBM Watson Studio. (Default)
5. Click Finish.
6. If you are NOT installing as an administrator, right-click the IBM Watson Studio icon on the desktop, and select Run as administrator.
7. Read and accept the terms and conditions, and then follow the online instructions until the message Installation complete! is displayed.
8. If you installed IBM Watson Studio Desktop as an administrator and you intend to use it as a different user, close and restart IBM Watson Studio Desktop so that the projects and other metadata match your login.
9. Select to launch IBM Watson Studio Desktop and click Finish.
10. Eject the ISO disk image: From Windows Explorer, navigate to the IBM\_Watson\_Studio DVD Drive. Right-click the drive, and select Eject.

When you first open IBM Watson Studio Desktop and you see a pop-up window that asks you to accept incoming network connections, click Accept.

The program, IBM Watson Studio Desktop, is installed by default in this path: C:\Program Files\IBM Watson Studio\IBM Watson Studio Desktop.exe.

## Next steps

---

- [Installing the environments for notebooks \(Version 1.1\)](#)
- [Create a project](#)
- **[Network access or other permissions for Windows \(Version 1.1\)](#)**  
Depending on your local setup, you might have to provide network access or other permissions to complete the installation on Windows. Watson Studio requires the privilege to set up a local `http` server and establish network connections.

**Parent topic:** [Installing Watson Studio Desktop 1.1](#)

# Network access or other permissions for Windows (Version 1.1)

---

Depending on your local setup, you might have to provide network access or other permissions to complete the installation on Windows. Watson Studio requires the privilege to set up a local `http` server and establish network connections.

## General installation - allowlist

---

To ensure a successful installation, the firewall (for example, Windows Defender Firewall) of your local machine must allow connections for the following applications:

IBM Watson Studio Desktop.exe

By default, located in the following Windows path:

C:\Program Files\IBM Watson Studio Desktop\IBM Watson Studio Desktop.exe

nginx.exe

By default, located in the following Windows path:

C:\Program Files\IBM Watson Studio Desktop\resources\application\core\nginx\nginx.exe

Rscript.exe

By default, located in the following Windows path:

C:\Program Files\IBM Watson Studio Desktop\resources\application\runtimes\r\R-3.5.1\bin\Rscript.exe

or

C:\Program Files\IBM Watson Studio Desktop\resources\application\runtimes\r\R-3.5.1\bin\x64\Rscript.exe

java.exe

By default, located in the following Windows path:

C:\Program Files\IBM Watson Studio Desktop\resources\application\runtimes\modeler\server\jre\bin

node.exe

By default, located in the following Windows path:

C:\Program Files\IBM Watson Studio Desktop\resources\application\dependencies\node\node.exe

## Notebook environments - allowlist

---

To ensure a successful environments installation, the firewall of your local machine must allow connection privileges for the following applications in Windows:

jupyter.exe

By default, located in the following Windows path:

C:\ProgramData\WatsonStudioDesktop\miniconda3\envs\desktop\Scripts\jupyter.exe

python.exe

By default, located in the following Windows path:

C:\ProgramData\WatsonStudioDesktop\miniconda3\envs\desktop\python.exe

jupyter-notebook.exe

By default, located in the following Windows path:

C:\ProgramData\WatsonStudioDesktop\miniconda3\envs\desktop\Scripts\jupyter-notebook.exe

**Parent topic:** [Installing on Windows \(Version 1.1\)](#)

## Installing on macOS (Version 1.1)

---

Install IBM Watson Studio Desktop 1.1 on macOS.

## Prerequisites

---

- Ensure that your system meets the [System requirements \(Version 1.1\)](#).
- If you have a *previous* version of IBM Watson Studio Desktop installed or a beta version installed, [uninstall](#) it before you install IBM Watson Studio Desktop 1.1.

## Installation steps

---

1. Double-click the .dmg file to open the pop-up window.
2. In the pop-up window, drag the IBM Watson Studio Desktop.app icon to the Applications folder and close the window.
3. Double-click IBM Watson Studio Desktop.app in the Applications folder, or use Spotlight to open the installation.
4. Click Finish to close the setup and run IBM Watson Studio Desktop. It may take a few minutes until a new dialog box is displayed for you to continue the installation.
5. Read and accept the terms and conditions, then follow the online instructions until the message Installation complete! is displayed.
6. Select to Launch IBM Watson Studio Desktop and click Finish.

The program, IBM Watson Studio Desktop, is installed by default in this path: /Applications/IBM Watson Studio Desktop.app.

Note: When you first open IBM Watson Studio Desktop and you see a pop-up window that asks you to accept incoming network connections, click Accept.

## Next steps

---

- [Installing the environments for notebooks \(Version 1.1\)](#)
- [Create a project](#)

**Parent topic:** [Installing Watson Studio Desktop 1.1](#)

## Installing the environments for notebooks (Version 1.1)

---

To start using notebooks you first have to install the environments that you need to create and run your notebooks in Watson Studio Desktop.

The first time you add a notebook to a project, you are prompted to install the notebook environments. Follow the instructions to finish your installation.

If you're using a Windows computer, make sure you run the installation as an administrator. If you're installing for a user other than the administrator, close the application, right-click the IBM Watson Studio icon on your desktop, and choose Run as administrator. After installation, restart the application so that the Notebooks metadata matches your login.

Note: The installation can take from 15 to 35 minutes or longer. Stay on this page until the installation is complete. When the installation completes, you can close the installation window and start creating notebooks. If you don't stay on the current page, the installation of environments is canceled.

## Uninstalling the notebook environments

---

To uninstall the notebook environment files, remove the WatsonStudioDesktop directory:

Windows

\ProgramData\WatsonStudioDesktop

macOS

/Users/*username*/WatsonStudioDesktop

**Parent topic:** [Installation and setup \(Version 1.1\)](#)



## Locating files (Version 1.1)

---

By default, Watson Studio Desktop 1.1 stores files in the following locations.

### Root installation folder

---

By default, files are stored in the following paths:

Windows

C:\Program Files\IBM Watson Studio Desktop

macOS

/Applications/IBM Watson Studio Desktop.app

### Default folder location

---

By default, files are stored in the following paths:

Windows

C:\Users\username\AppData\Roaming\IBM Watson Studio

macOS

/Users/username/Library/Application Support/IBM Watson Studio

### Project assets

---

By default, project assets are stored in the following paths:

Windows

C:\Users\username\AppData\Roaming\IBM Watson Studio\projects\

macOS

/Users/username/Library/Application Support/IBM Watson Studio/projects/

### Notebook files in projects

---

By default, notebooks that are created in your projects are stored in the following paths:

Windows

C:\Users\username\AppData\Roaming\IBM Watson Studio\projects\project-name\assets\notebook

macOS

/Users/username/Library/Application Support/IBM Watson Studio/projects/project-name/assets/notebook

### Notebook environments

---

By default, Miniconda3-4.5.12 is installed with Python 3. The default Python virtual environment is installed in the following paths:

Windows

C:\ProgramData\WatsonStudioDesktop\miniconda3\

macOS

/Users/username/WatsonStudioDesktop/miniconda3/

To use the Conda installation (Python runtime) you must be able to access:

<https://repo.anaconda.com/pkg/pro/noarch/repojson.json.bz2>

### Log files

---

By default, log files are stored in the following paths:

Windows

C:\Users\username\AppData\Roaming\IBM Watson Studio\logs\

macOS

/Users/username/Library/Application Support/IBM Watson Studio/logs/

**Parent topic:** [Installation and setup \(Version 1.1\)](#)

## Uninstalling Watson Studio Desktop 1.1

---

If you decide to uninstall IBM Watson Studio Desktop, your files are not deleted, but you cannot access them anymore.

The name of the IBM Watson Studio Desktop 1.1 application is *IBM Watson Studio Desktop*.

### Uninstall IBM Watson Studio Desktop on Windows

---

1. Sign in as an administrator.
2. Go to the Control Panel > Programs and Features and uninstall the IBM Watson Studio Desktop #.# program or the IBM Watson Studio Desktop program.

Files that are installed under C:\Users\username\AppData\Roaming\IBM Watson Studio are not removed. These files include the project files and other metadata.

### Uninstall IBM Watson Studio Desktop on macOS

---

Select Finder > Applications and right-click the IBM Watson Studio Desktop #.#.app or the IBM Watson Studio Desktop.app and select Move to Trash.

Files that are installed under /Users/username/Library/Application Support/IBM Watson Studio are not removed. These files include the project files and other metadata.

**Parent topic:** [Installation and setup \(Version 1.1\)](#)

## Connecting to IBM Watson Machine Learning Server

---

Watson Machine Learning Server is available as a separate offering for Watson Studio Desktop 1.1 or Watson Studio Desktop 2.0. With Watson Machine Learning Server, you can scale workloads for SPSS Modeler flows beyond the capabilities of a desktop compute environment and create deployments for SPSS models as well as open source frameworks.

### Prerequisite

Watson Machine Learning Server installed on your private network. The version number of Watson Studio Desktop and Watson Machine Learning Server must be the same. Installation and setup instructions for Watson Machine Learning Server are available at [Installing IBM Watson Machine Learning Server](#).

Before you can connect to Watson Machine Learning Server, a server administrator must follow the steps to:

- [Install the server](#)
- [Test the connection](#)
- [Set up user accounts](#)

After installation, the administrator must provide you with this connection information for the Watson Machine Learning Server:

- Server hostname
- Port number

- Username and password
- (Optional) A self-signed certificate to support an SSL connection with Watson Machine Learning Server.

## Creating the connection

1. Expand the side IBM Watson Studio pane, and click **Add-ons and services**.
2. Click **Add machine learning service**.
3. Click **Machine Learning Server**.
4. Enter the connection details.
5. Click **Connect**.

## Next step

[Deploying models to Watson Machine Learning Server](#)

# Projects

---

To get started with Watson Studio Desktop, you create a project, which is a container for organizing the assets that you will use for a data analysis task.

Projects are your home base for each task. Your project assets can include:

- Data assets (or links to data assets)
- Notebooks
- SPSS Modeler flows
- Data Refinery flows to refine the data in a data asset
- Connections (Information to create a connection to a data source)
- Deployment spaces

You can create a project, add assets to it, and manage it.

## Create a project

---

To create a project:

1. Choose Projects > View all projects from the left navigation and then click New project on the My projects page.
2. Choose to create an empty project or to import an existing project that was previously exported from Watson Studio Desktop.
3. Click Create. You can start adding assets if your project is empty, or begin working with the assets you imported.

Before you begin working with the imported assets, you should check for missing credentials, for example in notebooks and data connections, to enable successful relinking between the assets.

## Add assets to a project

---

You can add data assets and modeler flows to a project. To add an asset, click Add to project and choose the type of asset you want to add.

**Add to project** +

For more information about project assets, see

- [Add data to a project](#)
- [Notebooks](#)
- [Creating SPSS Modeler flows](#)
- [Refining data](#)
- [Adding connections to projects](#)

## Manage projects

---

You can change the project name and description from the Settings tab of the project. You can also delete a project from the My projects page.

- **Example projects**  
Watson Studio Desktop provides example projects that contain preloaded example assets to help you learn about the application. The examples are based on real-world scenarios and include many SPSS Modeler flows and their associated data sets. The flows contain computations and visualizations that you can adapt for your own needs.
- **Deployment spaces**
- **Exporting a project**

## Example projects

---

Watson Studio Desktop provides example projects that contain preloaded example assets to help you learn about the application. The examples are based on real-world scenarios and include many SPSS Modeler flows and their associated data sets. The flows contain computations and visualizations that you can adapt for your own needs.

Note: The flows included are intended to be for demonstration purposes only. They'll be replaced periodically when the product is updated. If you modify any of the examples and want to preserve your work, you should copy them as your own new assets to avoid them being replaced.

### Open the example project

---

1. Launch Watson Studio Desktop.
2. On the welcome screen, you see a list of projects. Select Example Project.
3. You'll see a list of many data assets and flows. Click View all to expand the lists. You can click a data asset to open it and read its description. The data assets are used in the example flows. Open a flow to read its description, run it, view its properties and its output, and experiment with it.

We recommend starting with the flow 01 - Introduction to Modeling.

Tip: See the [SPSS Modeler tutorials](#) associated with the examples.

### Open the example Text Analytics project

---

1. Launch Watson Studio Desktop.
  2. On the welcome screen, you'll see a list of projects. Select Example Project for Text Analytics.
  3. You'll see a list of data assets and a flow. Click the HotelSatisfaction flow to open it. Note this flow is large and we don't recommend running the entire flow. Only run individual nodes and branches.
- Tip: See [Hotel satisfaction example for Text Analytics](#) for details about this example.

**Parent topic:** [Projects](#)

## Deployment spaces (Version 2.0)

---

You can use deployment spaces to deploy models and manage your deployments.

Deployment spaces allow you to create deployments for saved models and view and manage all of the activity and assets for the deployments, including data connections and connected data assets. You can:

- [View deployed assets](#)
- [Create a deployment space](#)
- [Promote assets to a space](#)

- [Import a PMML model to a space](#)
- [Create deployments](#)
- [Export a deployment space](#)

## Viewing spaces

---

You configure and manage the deployment of a set of related assets in a space. A space contains an overview of deployment status, the deployable assets, deployments, associated input and output data, and the associated environments.

- To view all deployment spaces that you can access, click **Deployment Spaces** on the Watson Studio navigation menu.

## Creating a deployment space

---

A deployment space can only be associated with one project. If you have not already associated a deployment space with a project, when you attempt to promote an asset from a project, you are prompted to create a new space or choose an existing space to be associated with your project.

You can create a space from the **Overview** or **Settings** page of a project, or by following these steps:

1. Click **Deployment Spaces** on the Watson Studio navigation menu.
2. Click **Create new deployment space**
3. Choose whether to create a new space or import an existing one.
  - Choose **Empty Space** to create a new space.
  - Choose **Import space** to import a space that was created on IBM Watson Studio Desktop and saved as a .zip file. You can add the file from your file system. **Tip:** If you get an error importing a space file, try clearing your browser cookies then try again.
4. Enter the details for the space, then click **Create**.

View details about the space, including the space ID, from the **Settings** tab.

## Promoting assets to a deployment space

---

The following assets can be promoted to a deployment space:

- Saved models
- Data assets for use in deployments
- Connections defined in your project
- Functions
- Scripts

Promote or add assets to a deployment space in the following ways:

- From the space, choose **Add to space** and choose an asset type, such as data or machine learning model. Follow the prompt to upload or add the asset.
- From the project **Assets** page, choose **Promote** from the action menu for the asset to promote it to a deployment space.

For details on adding data to a space, see [Adding data sources to a space](#).

## Importing a model into the space

---

If you have a trained model saved in Predictive Model Markup Language (PMML) format in an .xml file, you can import that model directly into a deployment space and create a deployment for the model.

1. From the **Assets** tab of your deployment space, click **Add to space** and choose Watson Machine Learning model.

2. In the **Import model** dialog that displays, enter a name and optional description for the model.
3. Drop or upload the PMML file in the **Model content** box, then click **Import**.
4. Create a deployment for the model.

## Notes and restrictions

- Online is the only supported deployment type for PMML models.
- PMML models cannot be used in an SPSS stream flow.
- The PMML file must not contain a prolog. Depending on the library you are using when you save your model, a prolog might be added to the top of the file by default. For example, if your file contains a prolog string such as `spark-mllib-1r-model-pmml.xml`, remove the string before you import the PMML file to the deployment space.

## Creating a new deployment

---

When you promote a model to a space, components required for a successful deployment, such as a training library, model definition, or pipeline definition are automatically promoted as well. After promoting a model and data assets to a deployment space, you can create a deployment in the space.

1. Click the name of the saved model in the deployment space.
2. Click the Deployments tab.
3. Click **New Deployment** to create a deployment.
4. Choose the deployment type and fill out the specifics for the deployment. For details, see [deploying from a space](#).

## Exporting a deployment space

---

You can export a deployment space so that you can share the space with others or reuse the assets in another space.

To export a space:

1. From the space, click the export space icon.
2. Click **New export file**, specify a file name and optional description.
3. Select the assets you want to export with the space.
4. Click **Create** to create the export file.
5. Click **Download** to save the file.


You can reuse this space by choosing **Create a space from a file** when you create a new space.

## Exporting projects

---

In Watson Studio Desktop you can share a project with others by exporting the project. Exporting a project packs the project assets that you select into a single file that can be shared like any other file. You can export a project as a ZIP file to your desktop.

To export a project:

1. Open the project you want to export.
2. Check whether the assets that you include in your export, for example notebooks or connections, don't contain credentials or other sensitive information that you don't want to share. You should remove or hide this information before you begin the export.
3. Click  from the project toolbar and **Export to desktop**.
4. Select the assets to add. If you don't want to manually select all of the assets of a type, you can filter by asset type.

You can export the following asset types: data assets (including linked assets), connections, Data Refinery flows, and notebooks.

The data assets used in selected Data Refinery flows are included in the export file. When you select a linked asset, the linked asset is converted to a normal data asset and the data file is added to the project export file. Be mindful when exporting a linked data asset of the size of the data file. If the file is very large, it will increase the size of the ZIP file significantly and impact performance.

5. Optional: Change the export project file name and where to export the file to. By default, the project export file is saved to `C:\Users\<user Id>` on Windows and `/Users/<userId>` on macOS.

6. Click **Export**.

Do not leave the page while the export is running.

Ensure that your browser settings download the ZIP file to the desktop as a .zip file and not as a folder. Compressing this folder to enable project import leads to an error. Also note that you cannot manually add other assets to an exported project ZIP file on your desktop.

Currently, you can only import projects to Watson Studio Desktop that were exported from Watson Studio Desktop. For example, you cannot import a project to Watson Studio Desktop that was exported from Watson Studio Cloud.

## Adding data to a project

---

After you create a project, the next step is to add data to the project.

You can add these types of data assets to projects:


- [Local files](#)
- [Link to local files](#)
- [Database connections](#)

### Add local files

---

You can load files from your local system. Load CSV, delimited text files, Excel, or SAV files for SPSS Modeler or Data Refinery. Load the data files that you want to work with in your notebooks. The files are stored in the project's directory.

To add data files to a project:

1. From your project's **Assets** page, click **Add to project > Data** or click the **Find and add data** icon .
2. In the **Load** pane that opens, browse for the files or drag them onto the pane. You must stay on the page until the load is complete. You can cancel an ongoing load process if you want to stop loading a file.

The files are listed as data assets on the **Assets** page of your project.

Click the data asset name to preview the data. You can remove the data asset by choosing the **Remove** option from the action menu next to the asset name. Choose the **Refine** option to refine the data with Data Refinery.

### Link to local files

---

You can link to files on your local system. Unlike loading a file, the file is only referenced via a directory path. Linking to a file saves disk space and you can link to the same file from multiple projects.

Link to delimited text, CSV, SAV, or Excel files for SPSS Modeler or Data Refinery. Link to the data files that you want to work with in your notebooks.

To link data files to a project:

1. From your project's Assets page, click **Add to project > Link to file**.
2. Select one or more files on your local system.

The files are listed as data assets on the **Assets** page of your project. Click the data asset name to preview the data. You can remove the link by choosing the **Remove** option from the action menu next to the asset name. Choose the **Refine** option to refine the data with Data Refinery.

## Next steps

---

- [Refine your data](#) or go directly to [building an SPSS Modeler flow](#)
- [Analyze the data with notebooks](#)

## Adding connections to projects

---

You need to create a connection asset for a data source before you can load data from it. A connection asset contains the information necessary to create a connection to a data source.

You create connections to these types of data sources:

- IBM Cloud services
- Other cloud services
- On-premises databases

See [Connection types](#) for a full list of data sources.

To create a new connection in a project:

**Prerequisite:** Make sure you are connected to the internet before you add or edit a connection.

1. Click **Add to project > Connection**.
2. Choose a data source.
3. If necessary, enter the connection information required for your data source. Typically, you need to provide information like the host, port number, username, and password.  
Credentials are encrypted with an AES 256 algorithm and are stored with the connection asset in the project folder in which the asset resides.
4. Click **Create**. The connection appears on the **Assets** page. You can edit the connection by clicking the connection name on the **Assets** page.

**Note:** Connections to partitioned data are not supported.

## Next steps

---

Add data from this connection:

- [Navigate to Data Refinery](#), and select the connection. Drill down to a schema, and table or view.
- In SPSS Modeler, use the [Data Asset import node](#) to pull in data or the [Data Asset export node](#) to write data to remote data sources.

## Connection types

---

From a project, you must [create a connection](#) to a data source before you can read data from it or load data to it.

### Watson Machine Learning Server

If you are connected to Watson Machine Learning Server 2.0, you can promote the connection asset from a project to



a deployment space. The Watson Machine Learning Server needs to have access (for example, access behind a firewall) to the connection.

### Restriction for Data Refinery

Data Refinery does not support target connections.

### SPSS Modeler SQL optimization

For the list of databases supported for SQL pushback in SPSS Modeler flows, see [SQL optimization](#).

- [IBM services](#)
- [Third-party services](#)

## IBM services

---

- Analytics Engine HDFS (formerly known as BigInsights HDFS)
- Cloud Object Storage
- Cloud Object Storage (infrastructure)
- Cloudant  
When you create your Cloudant service on IBM Cloud, you must choose "Use both legacy credentials and IAM" for Available authentication methods.
- [Cognos Analytics](#) (*supports source connections only*)
- Compose for MySQL
- Databases for PostgreSQL (formerly known as Compose for PostgreSQL)
- Db2
- Db2 Big SQL
- Db2 for i
- Db2 for z/OS
- Db2 Hosted
- Db2 on Cloud
- Db2 Warehouse
- Informix
- Netezza (PureData System for Analytics)
- [Planning Analytics](#) (formerly known as IBM TM1)

## Third-party services

---

- Amazon Redshift
- Amazon S3
- Apache HDFS (formerly known as Hortonworks HDFS)
- Apache Hive (*supports source connections only*)
- Cloudera Impala (*supports source connections only*)
- Dropbox  
To obtain the application token that's needed to configure a Dropbox connection, follow the instructions at the [Dropbox OAuth guide](#).
- FTP (Remote file system transfer)
- Google BigQuery  
For **Connection Details**, enter either **Credentials** (the contents of the Google service account key JSON file) or the **Credentials path** (the path of the Google service account file).  
The connection to Google BigQuery requires the following BigQuery permissions:  
`bigquery.job.create`  
`bigquery.tables.get`  
`bigquery.tables.getData`

The predefined BigQuery Cloud IAM role `bigquery.admin` includes these permissions. Otherwise, a combination of two roles is needed. One role from each column in the following table.

First role	Second role
bigquery.dataEditor	bigquery.jobUser
bigquery.dataOwner	bigquery.user
bigquery.dataViewer	

For information about Google BigQuery's permissions and roles, see [Predefined roles and permissions](#).

- Google Cloud Storage
- Looker (*supports source connections only*)  
Before you configure the connection, you'll need to set up API3 credentials for your Looker instance. See instructions at [Looker API Authentication](#).
- Microsoft Azure Data Lake Store  
Before you configure the connection, you must create an Azure Active Directory (Azure AD) web application, get an application ID, authentication key, and a tenant ID. Then you must assign the Azure AD application to the Azure Data Lake Store account file or folder. Follow Steps 1, 2, and 3 at [Service-to-service authentication with Data Lake Store using Azure Active Directory](#).
- Microsoft Azure SQL Database
- Microsoft SQL Server
- MinIO
- MySQL
- OData (*supports source connections only*)
- Oracle
- Pivotal Greenplum
- PostgreSQL
- Salesforce.com (*supports source connections only*)
- SAP OData (*supports source connections only*)
- Snowflake
- Sybase (*supports source connections only*)
- Sybase IQ (*supports source connections only*)
- Tableau (*supports source connections only*)
- Teradata  
*Teradata JDBC Driver 15.10 Copyright (C) 2015 - 2017 by Teradata. All rights reserved. IBM provides embedded usage of the Teradata JDBC Driver under license from Teradata solely for use as part of the IBM Watson service offering.*

## IBM Cognos Analytics connection

---

You can create a connection asset for IBM Cognos Analytics in a project. If you are connected to Watson Machine Learning Server 2.0, you can promote the connection asset from a project to a deployment space.

Cognos Analytics is an AI-fueled business intelligence platform that supports the entire analytics cycle, from discovery to operationalization.

### Supported tools

---

These tools support Cognos Analytics connections:

- Data Refinery
- SPSS Modeler

### Restrictions

---

- For Data Refinery, you can use this connection only as a source. You cannot use this connection as a target connection.
- For SPSS Modeler, you can use this connection only to import data. You cannot export data to this connection.

## Supported versions

---

IBM Cognos Analytics 11

## Cognos Analytics setup

---

Instructions for setting up Cognos Analytics: [Getting started in Cognos Analytics](#).

## Supported encryption

---

(Optional) SSL certificate

## Credentials

---

- Username and password.
- Anonymous access is supported if it is enabled on the server.

## Supported content types

---

- Report (except Reports that require prompts)
- Query

## IBM Planning Analytics connection

---

You can create a connection asset for IBM Planning Analytics in a project. If you are connected to Watson Machine Learning Server 2.0, you can promote the connection asset from a project to a deployment space.

Planning Analytics (formerly known as TM1) is an enterprise performance management database that stores data in in-memory multidimensional OLAP cubes.

## Supported tools

---

These tools support Planning Analytics connections:

- Data Refinery. You must choose a *view* as the source.
- Notebooks. You must write your own code to access the data. See [Adding data from a Planning Analytics connection](#).
- SPSS Modeler. You can import only from a *view* and export only to a *cube*. For information about exporting to a Planning Analytics cube in SPSS Modeler, see [Data Asset Export node](#).

## Restrictions

---

- For Data Refinery, you can use this connection only as a source. You cannot use this connection as a target connection.
- For SPSS Modeler, in the Data Asset Export node, you must select **Replace the data set** when exporting to Planning Analytics.

## Supported versions

---

IBM Planning Analytics, version 2.0.5 or later

## Planning Analytics setup

---

Enable TM1 REST APIs on the TM1 Server. See TMI REST API [Installation and configuration](#).

## Supported encryption

---

(Optional) SSL certificate or SSL certificate file

## Credentials

---

These authentication methods are supported:

- Basic
- CAM Credentials
- CAM Passport
- Windows Integrated Authentication Token

For authentication setup information, see [Authenticating and managing sessions](#).

## Learn more

---

[Planning Analytics product documentation](#)

## OData connection

---

You can create a connection asset for OData in a project. If you are connected to Watson Machine Learning Server 2.0, you can promote the connection asset from a project to a deployment space.

The OData (Open Data) protocol is a REST-based data access protocol. The OData connection reads data from a data source that uses the OData protocol.

## Supported tools

---

These tools support OData connections:

- Data Refinery
- SPSS Modeler

## Restrictions

---

- For Data Refinery, you can use this connection only as a source. You cannot use this connection as a target connection.
- For SPSS Modeler, you can use this connection only to import data. You cannot export data to this connection.

## Supported versions

---

The OData connection is supported on OData protocol version 2 or version 4.

## OData setup

---

To set up the OData service, see [How to Use Web API OData to Build an OData V4 Service without Entity Framework](#).

## Supported encryption

---

(Optional) SSL certificate or SSL certificate file

## Credentials

---

- API Key

- Basic
- None

## Learn more

---

[www.odata.org](http://www.odata.org)

## SAP OData connection

---

You can create a connection asset for SAP OData in a project. If you are connected to Watson Machine Learning Server 2.0, you can promote the connection asset from a project to a deployment space.

Use the SAP OData connection to extract data from a SAP system through its exposed OData services.

## Supported tools

---

These tools support SAP OData connections:

- Data Refinery
- SPSS Modeler

## Restrictions

---

- For Data Refinery, you can use this connection only as a source. You cannot use this connection as a target connection.
- For SPSS Modeler, you can use this connection only to import data. You cannot export data to this connection.

## Supported SAP OData products

---

The SAP OData connection is supported on SAP products that support the OData protocol version 2 or version 4. Example products are S4/HANA (on premises or cloud), ERP, and CRM.

## SAP OData setup

---

See [Prerequisites for using the SAP ODATA Connector](#) for the SAP Gateway setup instructions. (Ignore the steps for DataStage.)

## Supported encryption

---

(Optional) SSL certificate or SSL certificate file

## Credentials

---

- API Key
- Basic
- None

## Refining data

---

Use Data Refinery to cleanse and shape tabular data with a graphical flow editor. You can also use interactive templates to code operations, functions, and logical operators. When you *cleanse data*, you fix or remove data that is incorrect, incomplete, improperly formatted, or duplicated. When you *shape data*, you customize it by filtering, sorting, combining or removing columns, and performing operations.

You create a *Data Refinery flow* as a set of ordered operations on data. Data Refinery includes a graphical interface to profile your data to validate it and over 20 customizable charts that give you perspective and insights into your data.

### Data format

CSV, delimited text, SAV, or Microsoft Excel. For Excel files, only the first sheet is read.

Tables in relational data sources

### Data size

Any. Data Refinery operates on a sample subset of rows in the data set. However, when you run a Data Refinery flow, the entire data set is processed.

- [Refine your data](#)
- [Data Refinery steps](#)

## Refine your data

1. Access Data Refinery from within a project. Click **Add to project**, and then choose **Data Refinery flow**. Then [select the data](#) you want to work with.

Alternatively, from the **Assets** tab of a project page, you can perform one of the following actions:

- Select **Refine** from the menu of a delimited-text (comma, tab, pipe, semicolon) data asset or a Microsoft Excel data asset
- Click a delimited-text data asset or a Microsoft Excel data asset to preview it first and then click the **Refine** link
- If you already have a Data Refinery flow, click **New Data Refinery flow** from the Data Refinery flows section and then [select the data](#) that you want to work with.

**Tip:** If your data doesn't display in tabular form, go to the **Data** tab. Scroll down to the SOURCE FILE information at the bottom of the page. Click the "Specify data format" icon. For more information, see [specify the format of your data source](#).

2. On the **Data** tab, apply operations to cleanse, shape, and enrich your data. You can [enter R code](#) in the command line and let autocomplete assist you in getting the correct syntax. Or you can use the Operations menu to [browse operation categories](#) or [search for a specific operation](#), then let the UI guide you.

The screenshot shows the Data Refinery 'Data' tab interface. At the top, there's a header with 'Operation' and a subtitle 'Code an operation to cleanse and shape your data'. Below this is a search bar labeled 'Search operations'. On the left, a 'FREQUENTLY USED' sidebar lists operations: Calculate, Convert column type, Filter, Math, Remove, Rename, and Sort ascending. The main area displays a data table with the following content:

Animal	city	ID	Phone
cat	Minneapolis	225	444
bird	San Francisco	25253	664
dog		NA	866
horse	Tucson	3	NA

On the right, a 'Steps' panel shows '1 Steps' with a 'Data Source' step (mydata.csv) and a 'Convert column type' step (AUTOMATIC). A note below the steps states: 'Automatically converted one or more columns to inferred data types.'

3. Click the **Profile** tab to [validate your data](#) throughout the data refinement process.



- Click the **Visualizations** tab to [visualize](#) the data in charts. Uncover patterns, trends, and correlations within your data.



- Refine the sample data set to suit your needs.
- Click **Save and Run flow**.
- Optional: In the Information pane **Details** tab, click the **Edit** button to change the name, description, and output details of the Data Refinery flow.
- Go back to the Project page to see the new assets:
  - The Data Refinery flow under the Data Refinery flows section.
  - The output of the Data Refinery flow under the Data assets section.

**Tip:** If you want to continue refining your data later, open the Data Refinery flow from the project's **Assets** tab > **Data Refinery flows** section and pick up from where you left off.

## Data Refinery steps

As you apply operations to a data set, Data Refinery keeps track of them and builds a Data Refinery flow. For each operation that you apply, Data Refinery adds a step in the Steps tab.

To see what your data looked like at any point in time, click a previous step to put Data Refinery into snapshot view. For example, if you click the data source step, you'll see what your data looked like before you started refining it. Click any operation step to see what your data looked like after that operation was applied. To leave snapshot mode, click **SNAPSHOT VIEW** or toggle it off by clicking the same step that you selected to get into snapshot view.

If you want to change the source of the Data Refinery flow, click the Edit icon next to **Data Source** (before the first step). For best results, the new data set should have a schema that is compatible to the original data set (for example, column names, number of columns, and data types). If the new data set has a different schema, operations that won't work with the schema will show errors. You can edit or delete the operations, or change the source to one that has a more compatible schema.

You can undo and redo operations from the toolbar. You can also insert, edit, and delete operations from the Steps tab.

To insert an operation between two steps:

1. Click the step before the position where you want to insert the new operation. Data Refinery shows you a snapshot view of the data set after that operation was applied.
2. Select and apply the new operation. Data Refinery inserts a new step between the existing steps, and it reruns all of the operations that follow the new step.

To edit an operation:

1. Click the Edit icon on the step for the operation you want to edit. Data Refinery goes into edit mode and either displays the operation to be edited on the command line or in the Operation pane.
2. Edit the operation or select a different operation to take its place.
3. Apply the edited operation. Data Refinery updates the relevant step to reflect your changes and it reruns all of the operations that follow the edited one.

## Learn more

---

- [Manage Data Refinery flows](#)

## Adding data to Data Refinery

---

You can add data to Data Refinery in one of several ways:

- Select **Refine** from the menu of a data asset in the project
- Preview a data asset in the project and then choose to refine it
- Navigate to Data Refinery first and then add data to it

## Navigate to Data Refinery

---

Access Data Refinery from within a project. Click **Add to project > Data Refinery flow**.

If you already have a Data Refinery flow, you can go to the project's **Assets** tab and click **New Data Refinery flow** in the Data Refinery flows section.

## Add data

---

To add data after you navigate to Data Refinery:

1. Select the data you want to work with from **Data assets** or from **Connections**.

From **Data assets**:



- Select a data file (the selection includes data files that have already been shaped with Data Refinery)

From **Connections:**

Make sure you are connected to the internet.

- Select a connection and file
- Select a connection, folder, and file
- Select a connection, schema, and table or view

Data Refinery supports CSV, delimited-text files, SAV files, and Microsoft Excel files (first sheet only).

2. Click **Add** to load the data into Data Refinery.

**Tip:** If your data doesn't display in tabular form, [specify the format of your data source](#). Go to the **Data** tab. Scroll down to the SOURCE FILE information at the bottom of the page. Click the "Specify data format" icon.

## Next steps


---

- [Refine your data](#)
- [Validate your data](#)
- [Use visualizations to gain insights into your data](#)

## Specifying the format of your data in Data Refinery

---

You can specify format options for CSV or TXT files.

When your data is read into Data Refinery, it should look like a well-formatted spreadsheet. If it doesn't display in tabular form or it doesn't look as you'd expect, go to the **Data** tab. Scroll down to the SOURCE FILE information at the bottom of the page. Click the "Specify data format" icon. 

**Tip:** Applying the format options might be an iterative process. Inspect your data after you apply an option. For example, changing the field delimiter to a semicolon might not work if the required delimiter is a pipe symbol.

Also, when you open a file in Data Refinery, the **Convert column type** operation is automatically applied as the first step if any of the column data types has been inferred as a non-string data type. If you see an error in the steps, click a previous step that is not in error to put Data Refinery into snapshot view, and then change the format options.

Use the following options to ensure that Data Refinery can correctly read your data:

### Indicate whether the first row of your data contains column headers

If your data doesn't contain column headers, Data Refinery will add them so that you can use them in cleansing and shaping operations.

### Encoding

The character encoding for the data source. Currently only UTF-8 is supported.

### Field delimiter

Identify the character that separates each field or column value from the next value.

### Quote character

Identify the character that encloses the field values. For example, CSV files typically enclose strings in double quotation marks.

**Default** means no changes. The inferred quote character is used.

### Escape character

Identify the character that's used to escape other characters, for example, backslashes ( \ ) are commonly used as escape characters. Escaping is a string technique that identifies characters (such as double quotation marks) as being part of a string value.

**Default** means no changes. The inferred escape character is used.

Click **Apply** to apply the format specification to your data and return to Data Refinery.

To change the data type or to specify the decimal symbol or the thousands grouping symbol of Integer or Decimal data types, use the **Convert column type** GUI operation. See [GUI operations in Data Refinery](#).

## Validating your data in Data Refinery

---

At any time after you've added data to Data Refinery, you can validate your data. Typically, you'll want to do this at multiple points in the refinement process.

To validate your data:

1. From Data Refinery, click the **Profile** tab.
2. Review the metrics for each column.
3. Take appropriate actions, as described in the following sections, depending on what you learn.

### Frequency

---

Frequency is the number of times that a value, or a value in a specified range, occurs. Each frequency distribution (bar) shows the count of unique values in a column.

Review the frequency distribution to find anomalies in your data. If you want to cleanse your data of those anomalies, simply remove the values.

For Integer and Date/Time columns, you can customize the number of bins (groupings) that you want to see. In the default multi-column view, the maximum is 20. If you expand the frequency chart row, the maximum is 50.

### Statistics

---

Statistics are a collection of quantitative data. The statistics for each column show the minimum, maximum, mean, and number of unique values in that column.

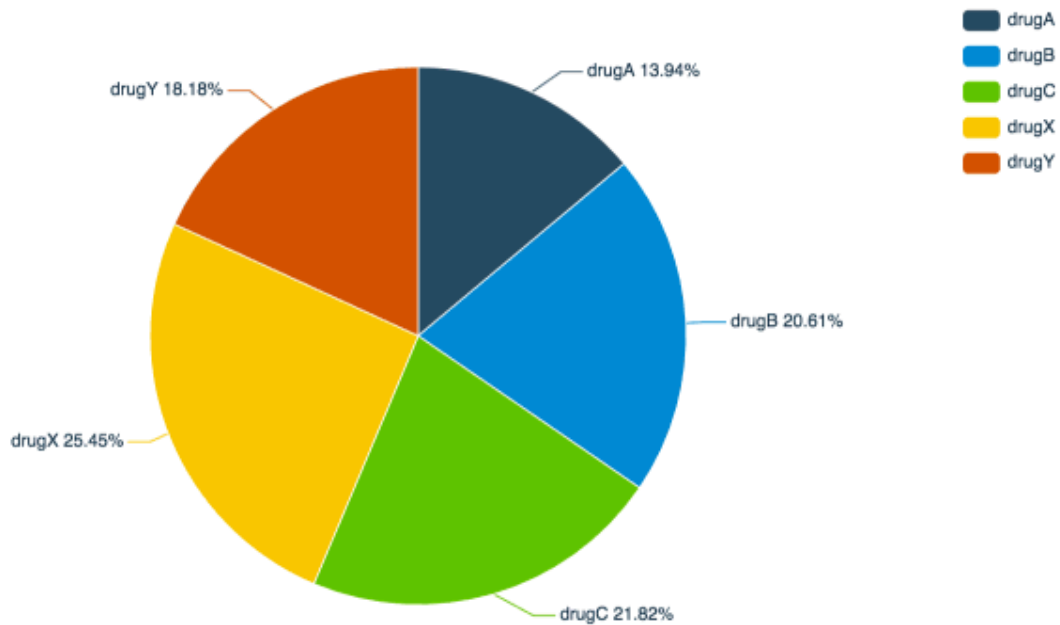
Depending on a column's data type, the statistics for each column will vary slightly. For example, statistics for a column of data type integer have minimum, maximum, and mean values while statistics for a column of data type string have minimum length, maximum length, and mean length values.

## Visualizing your data in Data Refinery

---

Visualizing information in graphical ways gives you insights into your data. By exploring data from different perspectives with visualizations, you can identify patterns, connections, and relationships within that data as well as quickly understand large amounts of information.

You can also visualize your data with these same charts in an SPSS Modeler flow. Use the Charts node, which is available under the Graphs section on the node palette. Double-click the Charts node to open the properties pane. Then click **Launch Chart Builder** to open the chart builder and create one or more chart definitions to associate with the node.



The same data asset might show different results in the charts in Data Refinery and the SPSS Modeler Chart builder because each tool loads a different sample size:


- Data Refinery: 10 MB or 100,000 records, whichever comes first.
- SPSS Modeler Chart builder: 5,000 records

To visualize your data:

**Prerequisite:** Decimal and Integer data types must use the dot (.) for the decimal symbol and optionally the comma (,) for the thousands grouping symbol. If your data uses a different format, use the Data Refinery **Convert column type** GUI operation to change the symbols. See [GUI operations](#).

1. From Data Refinery, click the **Visualizations** tab.
2. Start with a chart or select columns:
  - Click any of the available charts. Then, add columns in the **DETAILS** pane that opens on the left side of the page.
  - Select the columns that you want to work with. Suggested charts are indicated with a dot next to the chart name. Click a chart to visualize your data.

Watch this short video for an example of using the Visualization charts in Data Refinery.

Figure 1.  Visualization charts in Data Refinery

## Data Visualization in Watson Studio Desktop



<https://youtu.be/pUACjgVF218>

**Important:** Available chart types are ordered from most relevant to least relevant, based on the selected columns. If there are no columns in the data set with a data type that is supported for a chart type, that chart will not be available. If a column's data type is not supported for a chart, that column is not available for selection for that chart. Dots next to the charts' names suggest the best charts for your data.

## Charts

---

Data Refinery includes the following charts:

- 3D charts display data in a 3-D coordinate system by drawing each column as a cuboid to create a 3D effect.
- Bar charts are handy for displaying and comparing categories of data side by side. The bars can be in any order. You can also arrange them from high to low or from low to high.
- Box plot charts compare distributions between many groups or data sets. They display the variation in groups of data: the spread and skew of that data and the outliers.
- Bubble charts display each category in the groups as a bubble.
- Candlestick charts are a type of financial chart that displays price movements of a security, derivative, or currency.
- Circle packing charts display hierarchical data as a set of nested areas.
- Customized charts give you the ability to render charts based on JSON input.
- Dual Y-axes charts use two Y-axis variables to show relationships between data.
- Error bars indicate the error or uncertainty in a value. They give a general idea of how precise a value is or conversely, how far a value might be from the true value.
- Evaluation charts are combination charts that measure the quality of a binary classifier. You need three columns for input: actual (target) value, predict value, and confidence (0 or 1). Move the slider in the Cutoff chart to dynamically update the other charts. The ROC and other charts are standard measurements of the classifier.
- Heat map charts display data as color to convey activity levels or density. Typically low values are displayed as cooler colors and high values are displayed as warmer colors.

- Histogram charts show the frequency distribution of data.
- Line charts show trends in data over time by calculating a summary statistic for one column for each value of another column and then drawing a line that connects the values.
- Map charts show geographic point data, so you can compare values and show categories across geographical regions.
- Math curve charts display a group of curves based on equations that you enter. You do not use a data set with this chart. Instead, you use it to compare the results with the data set in another chart, like the scatter plot chart.
- Multi-charts display up to four combinations of Bar, Line, Pie, and Scatter plot charts. You can show the same kind of chart more than once with different data. For example, two pie charts with data from different columns.
- Multi-series charts display data from multiple data sets or multiple columns as a series of points that are connected by straight lines or bars.
- Parallel coordinate charts display and compare rows of data (called profiles) to find similarities. Each row is a line and the value in each column of the row is represented by a point on that line.
- Pie charts show proportion. Each value in a series is displayed as a proportional slice of the pie. The pie represents the total sum of the values.
- Population pyramid charts show the frequency distribution of a variable across categories. They are typically used to show changes in demographic data.
- Quantile-quantile (Q-Q) plot charts compare the expected distribution values with the observed values by plotting their quantiles.
- Radar charts integrate three or more quantitative variables that are represented on axes (radii) into a single radial figure. Data is plotted on each axis and joined to adjacent axes by connecting lines. Radar charts are useful to show correlations and compare categorized data.
- Relationship charts show how columns of data relate to one another and what the strength of that relationship is by using varying types of lines.
- Scatter matrix charts map columns against each other and display their scatter plots and correlation. Use to compare multiple columns and how strong their correlation is with one another.
- Scatter plot charts show correlation (how much one variable is affected by another) by displaying and comparing the values in two columns.
- Sunburst charts are similar to layered pie charts, in which different proportions of different categories are shown at once on multiple levels.
- Theme river charts use a specialized flow graph that shows changes over time.
- Time plot charts illustrate data points at successive intervals of time.
- t-SNE charts help you visualize high-dimensional data sets. They're useful for embedding high-dimensional data into a space of two or three dimensions, which can then be visualized in a scatter plot.
- Tree charts display hierarchical data, categorically splitting into different branches. Use to sort different data sets under different categories. The Tree chart consists of a root node, line connections called branches that represent the relationships and connections between the members, and leaf nodes that do not have child nodes.
- Treemap charts display hierarchical data as a set of nested areas. Use to compare sizes between groups and single elements that are nested in the groups.

- Word cloud charts display how frequently words appear in text by making the size of each word proportional to its frequency.

## Actions

---

You can take any of the following actions:

- Download visualization:
  - Download chart details: Download a JSON file that contains the details for the current chart.
  - Download chart image: Download a PNG file that contains an image of the current chart.
- Specify the field format that is displayed in the chart:
  - Numeric values: Currency, percentage, or exponential
  - Date values: Date format
- Set global preferences that apply to all charts
- Display data label and Display data value: This feature is not available at this time.
- Start over: Clears the visualization and the **DETAILS** pane, and returns you to the starting page for visualizations

## Chart actions

---

Available chart actions depend on the chart. Chart actions include:

- Zoom
- Restore: View the chart at normal scale
- Select data: Highlight data in the Data tab that you select in the chart
- Clear selection: Remove highlighting from the data in the Data tab

## Learn more

---

[Data Visualization - How to Pick the Right Chart Type?](#)

## Managing Data Refinery flows


---

A Data Refinery flow is an ordered set of steps to cleanse, shape, and enhance data. As you [refine your data](#) by applying operations to a data set, you dynamically build a customized Data Refinery flow that you can modify in real time and save for future use.

- [Save a Data Refinery flow](#)
- [Run a Data Refinery flow](#)
- [Reopen a Data Refinery flow to continue working](#)
- [Change the Data Refinery output file](#)
- [Change the source of a Data Refinery flow](#)
- [Rename a Data Refinery flow](#)
- [Remove a Data Refinery flow](#)

## Save a Data Refinery flow

---


Save a Data Refinery flow by clicking the Save Data Refinery flow icon  in the Data Refinery toolbar. Data Refinery flows are saved to the project that you're working in. Save a Data Refinery flow so that you can continue refining a data set later.

The default output of the Data Refinery flow is saved as a data asset file in CSV format: `source_file_name_shaped.csv`. For example, if the source file is `airline-data.csv`, the default name and output for the Data Refinery flow is `airline-data.csv_flow`.

## Run a Data Refinery flow

---

Data Refinery supports large data sets, which can be time-consuming and unwieldy to refine. So that you can work quickly and efficiently, Data Refinery operates on a sample subset of rows in each data set. When you the run the Data Refinery flow, the entire data set is processed.

To run a Data Refinery flow, click the Run icon  in the Data Refinery toolbar.


## Reopen a Data Refinery flow to continue working

---

To reopen a Data Refinery flow and continue refining your data, go to the project's **Assets** tab. Scroll down to the **Data Refinery flows** section and click the Data Refinery flow name.

## Change the Data Refinery flow output file

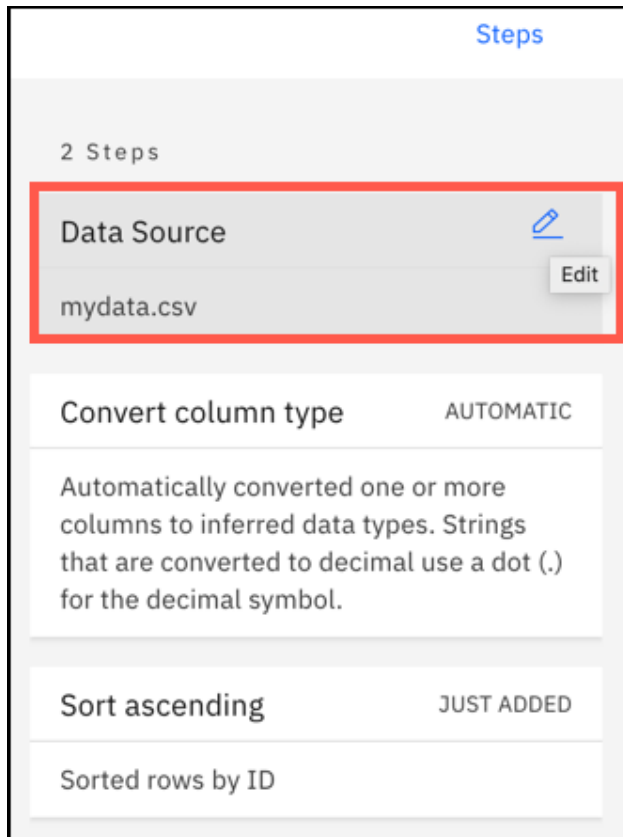
---

1. In Data Refinery, open the Information pane  and click the **Details** tab.
2. Click the **Edit** button.
3. In the DATA REFINERY FLOW OUTPUT pane, click the Edit icon to change any of the following properties:
  - Data set name and description
  - Column header information

## Change the source of a Data Refinery flow

---

Change the source of a saved Data Refinery flow. Run the same Data Refinery flow but with a different source data set. In the **Steps** pane in Data Refinery, click the Edit icon next to **Data Source** to choose a different source data set.



For best results, the new data set should have a schema that is compatible to the original data set (for example, column names, number of columns, and data types). If the new data set has a different schema, operations that won't work with the schema will show errors. You can edit or delete the operations, or change the source to one that has a more compatible schema.

## Rename a Data Refinery flow

---

1. In Data Refinery, open the info pane ⓘ and click the **Details** tab.
2. Click the Edit icon next to the Data Refinery name.

## Remove a Data Refinery flow

---

To remove a Data Refinery flow, go to the project. Select the **Assets** tab. Scroll down to the **Data Refinery flows** section and select **Remove** from the ACTIONS menu.

## GUI operations in Data Refinery

---

Data Refinery supports the following categories of GUI operations.

- [FREQUENTLY USED](#)
- [CLEANSE](#)
- [ORGANIZE](#)

Select a GUI operation from the **+ Operation** button.

A subset of the operations is available from each column's actions menu (three vertical dots in the column header). You can rename a column by clicking the Edit icon in the column header.

## FREQUENTLY USED

---



## Calculate

Perform a calculation with another column or with a specified value. The operators are:

- Addition
- Division
- Exponentiation
- Is between two numbers
- Is equal to
- Is greater than
- Is greater than or equal to
- Is less than
- Is less than or equal to
- Is not equal to
- Modulus
- Multiplication
- Subtraction

## Convert column type

When you open a file in Data Refinery, the **Convert column type** operation is automatically applied as the first step if it detects any non-string data types in the data. Data types are automatically converted to inferred data types. Click Edit to change the automatic conversion for selected columns. As with any other operation, you can undo the step. The **Convert column type** operation is reapplied every time that you open the file in Data Refinery. Automatic conversion is applied as needed for file-based data sources only. (It does not apply to a data source from a database connection.)

If the data is converted to an Integer or to a Decimal data type, you can specify the decimal symbol and the thousands grouping symbol for all applicable columns. Strings that are converted to the Decimal data type use a dot for the decimal symbol and a comma for the thousands grouping symbol. Alternatively, you can select comma for the decimal symbol and dot or a custom symbol for the thousands grouping symbol. The decimal symbol and the thousands grouping symbol cannot be the same.

The source data is read from left to right until a terminator or an unrecognized character is encountered. For example, if you are converting string data 12,834 to Decimal and you do not specify what to do with the comma (,), the data will be truncated to 12. Similarly, if the source data has multiple dots (.), and you select dot for the decimal symbol, the first dot is used as the decimal separator and the digits following the second dot are truncated. A source string of 1.834.230,000 is converted to a 1.834 value.

The **Convert column type** operation automatically converts these date and timestamp formats:

- Date: ymd, ydm
- Timestamp: ymdHMS, ymdHM, ydmHMS, ydmHM

You can manually apply the **Convert column type** operation to change the data type of a column at any point in the Data Refinery flow. You can create a new column to hold the result of this operation or you can overwrite the existing column.

Tip: A column's data type determines the operations that you can perform. Changing the data type can affect which operations are relevant for that column.

## Filter

Filter rows by the selected columns. Keep rows with the selected column values; filter out all other rows.

The operators for numeric, string, and Boolean (logical), and date and timestamp columns are:

Operator	Numeric	String	Boolean	Date and timestamp
Contains		✓		
Does not contain		✓		
Does not end with		✓		

Operator	Numeric	String	Boolean	Date and timestamp
Does not start with		✓		
Ends with		✓		
Is between two numbers	✓			
Is empty		✓	✓	✓
Is equal to	✓	✓		✓
Is false			✓	
Is greater than	✓			✓
Is greater than or equal to	✓			✓
Is in	✓	✓		
Is less than	✓			✓
Is less than or equal to	✓			✓
Is not empty		✓	✓	✓
Is not equal to	✓	✓		✓
Is not in	✓	✓		
Is not null		✓		
Is null	✓	✓		
Is true			✓	
Starts with		✓		

## Math

You can apply math operations only to numeric columns. You can create a new column to hold the result of an operation or you can overwrite the existing column.

### Math > Absolute value

Get the absolute value of a number.

Example: The absolute value of both 4 and -4 is 4.

### Math > Arc cosine

Get the arc cosine of an angle.

### Math > Ceiling

Get the nearest integer of greater value, also known as the ceiling of the number.

Examples: The ceiling of 2.31 is 3. The ceiling of -2.31 is -2.

### Math > Exponent

Get a number raised to the power of the column value.

### Math > Floor

Get the nearest integer of lesser value, also known as the floor of the number.

Example: The floor of 2.31 is 2. The floor of -2.31 is -3.

### Math > Round

Get the whole number nearest to the column value. If the column value is a whole number, return it.

### Math > Square root

Get the square root of the column value.

### Remove

Remove the selected column.

### Rename

Rename the selected column.

**Sort ascending**

Sort rows by the selected column in ascending order.

**Sort descending**

Sort rows by the selected column in descending order.

**Substitute**

Obscure sensitive information from view by substituting a random string of characters for the actual data in the selected column.

## Text

You can apply text operations only to string columns. You can create a new column to hold the result of an operation or you can overwrite the existing column.

**Text > Collapse spaces**

Collapse multiple, consecutive spaces in the text to a single space.

**Text > Concatenate string**

Link together any string to the text. You can prepend the string to the text, append the string to the text, or both.

**Text > Lower case**

Convert the text to lower case.

**Text > Number of characters**

Return the number of characters in the text.

**Text > Pad characters**

Pad the text with the specified string. Specify whether to pad the text on the left, right, or both the left and right.

**Text > Substring**

Create substrings from the text that start at the specified position and have the specified length.

**Text > Title case**

Convert the text to title case.

**Text > Trim quotes**

Remove single or double quotation marks from the text.

**Text > Trim spaces**

Remove leading, trailing, and extra spaces from the text.

**Text > Upper case**

Convert the text to upper case.

## CLEANSE

---

**Convert column value to missing**

Convert values in the selected column to missing values if they match values in the specified column or they match a specified value.

**Extract date or time value**

Extract a selected portion of a date or time value from a column with a date or timestamp data type.

**Remove duplicates**

Remove rows with duplicate column values.

**Remove empty rows**

Remove rows that have a blank or missing value for the selected column.

### Replace missing values

Replace missing values in the column with a specified value or with the value from a specified column in the same row.

### Replace substring

Replace the specified substring with the specified text.

## ORGANIZE

---

### Aggregate

Apply summary calculations to the values of one or more columns. Each aggregation creates a new column. Optionally select **Group by columns** to group the new column by another column that defines a characteristic of the group, for example, a department or an ID. You can group by multiple columns. You can combine multiple aggregations in a single operation.

The available aggregate operations depend on the data type.

Numeric data:

- Count unique values
- Minimum
- Maximum
- Sum
- Standard deviation
- Mean

String data:

- Combine row values
- Count unique values

### Concatenate

Concatenate the values of two or more columns.

### Conditional replace

Replace the values in a column based on conditions.

### Join

Combine data from two data sets based on a comparison of the values in specified key columns. Specify the type of join to perform, select the columns (join keys) in both data sets that you want to compare, and select the columns that you want in the resulting data set.

The join key columns in both data sets need to be compatible data types. If the **Join** operation is the first step that you add, check whether the **Convert column type** operation automatically converted the data type of the join key columns in the first data set when you opened the file in Data Refinery. Also, depending where the **Join** operation is in the Data Refinery flow, you can use the **Convert column type** operation to ensure that the join key column data types match. Click **Steps** to see the snapshot view of the steps.

The join types include:

Join type	Description
Left join	Returns all rows in the original data set and return only matching rows in the joining data set. Returns one row in the original data set for each matching row in the joining data set.
Right join	Returns all rows in the joining data set and return only matching rows in the original data set. Returns one row in the joining data set for each matching row in the original data set.
Inner join	Returns only the rows in each data set that match rows in the other data set. Returns one row in the original data set for each matching row in the joining data set.

Join type	Description
Full join	Returns all rows in both data sets. Blends rows in the original data set with matching rows in the joining data set.
Semi join	Returns only the rows in the original data set that match rows in the joining data set. Returns one row in the original data set for all matching rows in the joining data set.
Anti join	Returns only the rows in the original data set that do not match rows in the joining data set.

### Split column

Split the column by non-alphanumeric characters, position, pattern, or text.

### Union

Combine the rows from two data sets that share the same schema and filter out the duplicates. If you select **Allow a different number of columns and allow duplicate values**, the operation is a `UNION ALL` command.

Tip: If you receive an error about incompatible schemas, check if the automatic **Convert column type** operation changed the data types of the first data set. Delete the **Convert column type** step and try again.

## Interactive code templates in Data Refinery

Data Refinery provides interactive templates for you to code operations, functions, and logical operators. Access the templates from the command-line text box at the top of the page. The templates include interactive assistance to help you with the syntax options.

### Important

Support is for the operations and functions in the user interface. If you insert other operations or functions from an open source library, the Data Refinery flow might fail. See the command-line help and be sure to use the list of operations or functions from the templates. Use the examples in the templates to further customize the syntax as needed.

- [Operations](#)
- [Functions](#)
- [Logical operators](#)

## Operations

### arrange

```
arrange(`<column>`)
```

Sort rows, in ascending order, by the specified columns.

```
arrange(desc(`<column>`))
```

Sort rows, in descending order, by the specified column.

```
arrange(`<column>`, `<column>`)
```

Sort rows, in ascending order, by each specified, successive column, keeping the order from the prior sort intact.

### count

```
count()
```

Total the data by group.

```
count(`<column>`)
```

Group the data by the specified column and return the number of rows with unique values (for string values) or return the total for each group (for numeric values).

```
count(`<column>`, wt=`<column>`)
```

Group the data by the specified column and return the number of rows with unique values (for string values) or return the total for each group (for numeric values) in the specified weight column.

```
count(`<column>`, wt=<func>(`<column>`))
```

Group the data by the specified column and return the result of the function applied to the specified weight column.

```
count(`<column>`, wt=<func>(`<column>`), sort = <logical>)
```

Group the data by the specified column and return the result of the function applied to the specified weight column, sorted or not.

## distinct

```
distinct()
```

Keep distinct, unique rows based on all columns or on specified columns.

## filter

```
filter(`<column>` == <logical>)
```

Keep rows that meet the specified filter conditions based on logical value TRUE or FALSE.

```
filter(`<column>` <logicalOperator> provide_value)
```

Keep rows that meet the specified condition and filter out all other rows.

For the Boolean column type, provide\_value should be uppercase TRUE or FALSE.

```
filter(<func>(`<column>`) <logicalOperator> provide_value)
```

Keep rows that meet the specified condition and filter out all other rows. The condition can apply a function to a column on the left side of the operator.

```
filter(`<column>` <logicalOperator> <func>(column))
```

Keep rows that meet the specified condition and filter out all other rows. The condition can apply a function to a column on the right side of the operator.

```
filter(<logicalfunc>(column))
```

Keep rows that meet the specified condition and filter out all other rows. The condition can apply a logical function to a column.

```
filter(`<column>` <logicalOperator> provide_value <andor> `<column>` <logicalOperator> provide_value)
```

Keep rows that meet the specified conditions and filter out all other rows.

## group\_by

```
group_by(`<column>`)
```

Group the data based on the specified column.

```
group_by(desc(`<column>`))
```

Group the data, in descending order, based on the specified column.

## mutate

```
mutate(provide_new_column = `<column>`)
```

Add a new column and keep existing columns.

```
mutate(provide_new_column = <func>(column))
```

Add a new column by using the specified expression, which applies a function to a column. Keep existing columns.

```
mutate(provide_new_column = case_when(`<column>` <operator> provide_value_or_column_to_compare ~  
provide_value_or_column_to_replace, `<column>` <operator> provide_value_or_column_to_compare ~  
provide_value_or_column_to_replace, TRUE ~ provide_default_value_or_column))
```

Add a new column by using the specified conditional expression.

```
mutate(provide_new_column = `<column>` <operator> `<column>`)
```

Add a new column by using the specified expression, which performs a calculation with existing columns. Keep existing columns.

```
mutate(provide_new_column = coalesce(`<column>`, `<column>`))
```

Add a new column by using the specified expression, which replaces missing values in the new column with values from another, specified column. As an alternative to specifying another column, you can specify a value, a function on a column, or a function on a value. Keep existing columns.

```
mutate(provide_new_column = if_else(`<column>` <logicalOperator> provide_value, provide_value_for_true, provide_value_for_false))
```

Add a new column by using the specified conditional expression. Keep existing columns.

```
mutate(provide_new_column = `<column>`, provide_new_column = `<column>`)
```

Add multiple new columns and keep existing columns.

```
mutate(provide_new_column = n())
```

Count the values in the groups. Ensure grouping is done already using `group_by`. Keep existing columns.

## **mutate\_all**

```
mutate_all(funs(<func>))
```

Apply the specified function to all of the columns and overwrite the existing values in those columns. Specify whether to remove missing values.

```
mutate_all(funs(. <operator> provide_value))
```

Apply the specified operator to all of the columns and overwrite the existing values in those columns.

```
mutate_all(funs("provide_value" = . <operator> provide_value))
```

Apply the specified operator to all of the columns and create new columns to hold the results. Give the new columns names that end with the specified value.

## **mutate\_at**

```
mutate_at(vars(`<column>`), funs(<func>))
```

Apply functions to the specified columns.

## **mutate\_if**

```
mutate_if(<predicateFunc>, <func>)
```

Apply functions to the columns that meet the specified condition.

```
mutate_if(<predicateFunc>, funs(. <operator> provide_value))
```

Apply the specified operator to the columns that meet the specified condition.

```
mutate_if(<predicateFunc>, funs(<func>))
```

Apply functions to the columns that meet the specified condition. Specify whether to remove missing values.

## **rename**

```
rename(provide_new_column = `<column>`)
```

Rename the specified column.

## **sample\_frac**

```
sample_frac(provide_number_between_0_and_1, weight=`<column>`,replace=<logical>)
```

Generate a random sample based on a percentage of the data. `weight` is optional and is the ratio of probability the row will be chosen. Provide a numeric column. `replace` is optional and its Default is `FALSE`.

## sample\_n

`sample_n(provide_number_of_rows,weight= `<column>`,replace=<logical>)`

Generate a random sample of data based on a number of rows. weight is optional and is the ratio of probability the row will be chosen. Provide a numeric column. replace is optional and its default is FALSE.

## select

`select(`<column>`)`

Keep the specified column.

`select(-`<column>`)`

Remove the specified column.

`select(starts_with("provide_text_value"))`

Keep columns with names that start with the specified value.

`select(ends_with("provide_text_value"))`

Keep columns with names that end with the specified value.

`select(contains("provide_text_value"))`

Keep columns with names that contain the specified value.

`select(matches("provide_text_value"))`

Keep columns with names that match the specified value. The specified value can be text or a regular expression.

`select(`<column>`:`<column>`)`

Keep the columns in the specified range. Specify the range as from one column to another column.

`select(`<column>`, everything())`

Keep all of the columns, but make the specified column the first column.

`select(`<column>`,`<column>`)`

Keep the specified columns.

## select\_if

`select_if(<predicateFunc>)` Keep columns that meet the specified condition. Supported functions include:

- contains
- ends\_with
- matches
- num\_range
- starts\_with

## summarize

`summarize(provide_new_column = <func>(`<column>`))`

Apply aggregate functions to the specified columns to reduce multiple column values to a single value. Be sure to group the column data first by using the `group_by` operation.

## summarize\_all

`summarize_all(<func>)`

Apply an aggregate function to all of the columns to reduce multiple column values to a single value. Specify whether to remove missing values. Be sure to group the column data first by using the `group_by` operation.

`summarize_all(funs(<func>))`

Apply multiple aggregate functions to all of the columns to reduce multiple column values to a single value. Create



new columns to hold the results. Specify whether to remove missing values. Be sure to group the column data first by using the `group_by` operation.

## summarize\_if

`summarize_if(<predicate_conditions>,...)`

Apply aggregate functions to columns that meet the specified conditions to reduce multiple column values to a single value. Specify whether to remove missing values. Be sure to group the column data first by using the `group_by` operation. Supported functions include:

- `count`
- `max`
- `mean`
- `min`
- `standard deviation`
- `sum`

## tally

`tally()`

Counts the number of rows (for string columns) or totals the data (for numeric values) by group. Be sure to group the column data first by using the `group_by` operation.

`tally(wt= `<column>`)`

Counts the number of rows (for string columns) or totals the data (for numeric columns) by group for the weighted column.

`tally( wt=<func>(`<column>`), sort = <logical>)`

Applies a function to the specified weighted column and returns the result, by group, sorted or not.

## top\_n

`top_n(provide_value)`

Select the top or bottom N rows (by value) in each group. Specify a positive integer to select the top N rows; specify a negative integer to select the bottom N rows.

`top_n(provide_value, `<column>`)`

Select the top or bottom N rows (by value) in each group, based on the specified column. Specify a positive integer to select the top N rows; specify a negative integer to select the bottom N rows.

## transmute

`transmute(<new_or_existing_column> = `<column>`)`

Add a new column or overwrite an existing one by using the specified expression. Keep only columns that are mentioned in the expression.

`transmute(<new_or_existing_column> = <func(column)>)`

Add a new column or overwrite an existing one by applying a function to the specified column. Keep only columns that are mentioned in the expression.

`transmute(<new_or_existing_column> = `<column>` <operator> `<column>`)`

Add a new column or overwrite an existing one by applying an operator to the specified column. Keep only columns that are mentioned in the expression.

`transmute(<new_or_existing_column> = `<column>`, <new_or_existing_column> = `<column>`)`

Add multiple new columns. Keep only columns that are mentioned in the expression.

`transmute(<new_or_existing_column> = if_else( provide_value, provide_value_for_true, provide_value_for_false))`

Add a new column or overwrite an existing one by using the specified conditional expressions. Keep only columns that

are mentioned in the expressions.

## ungroup

ungroup()  
Ungroup the data.

## Functions

---

### Aggregate

- mean
- min
- n
- sd
- sum

### Logical

- is.na

### Numerical

- abs
- coalesce
- cut
- exp
- floor

### Text

- c
- coalesce
- paste
- tolower
- toupper

### Type

- as.character
- as.double
- as.integer
- as.logical

## Logical operators

---

- <
- <=
- >=
- >
- between
- !=
- ==
- %in%

## Data Refinery tutorial: Shape raw data

---

This tutorial demonstrates a few of the many data shaping capabilities of Data Refinery.

- [About this tutorial](#)
- [Prerequisites](#)
- [Bring the data into Data Refinery](#)
- [Review the data with Profile and Visualizations](#)
- [Data Refinery operations](#)
- [Refine the data](#)
- [Run the Data Refinery flow](#)
- [Create another data asset from the Data Refinery flow](#)
- [View the data assets and your Data Refinery flow in your project](#)

## About this tutorial

---

This tutorial is for users who are new to Data Refinery. You'll start with a source CSV file that contains data about different airlines. You'll create a Data Refinery flow that refines the data to one airline with columns for the total arrival and delay times, grouped by year, month, and day, with a sorted column for the average (mean) of all the delay times.

It will take you approximately 30 minutes to complete this tutorial.


## Prerequisites

---

1. Make sure you have access to the internet to download a CSV file.
2. Create a project. Go to the Projects page and click **New project**. You can name the project anything you like.

## Bring the data into Data Refinery

---

1. Download the [airline-data.csv file \(1.5 MB\)](#)   
Right-click the browser window, and then choose **Save Page As** or **Save As** depending on your browser. (For Safari: Choose Format: Page Source.) Make sure the downloaded file name is `airline-data.csv`.
2. Add the `airline-data.csv` file to your project:
  1. From your project's **Assets** page, click **Add to project > Data**.
  2. In the **Load** pane that opens, browse to the `airline-data.csv` file. Stay on the page until the load completes.

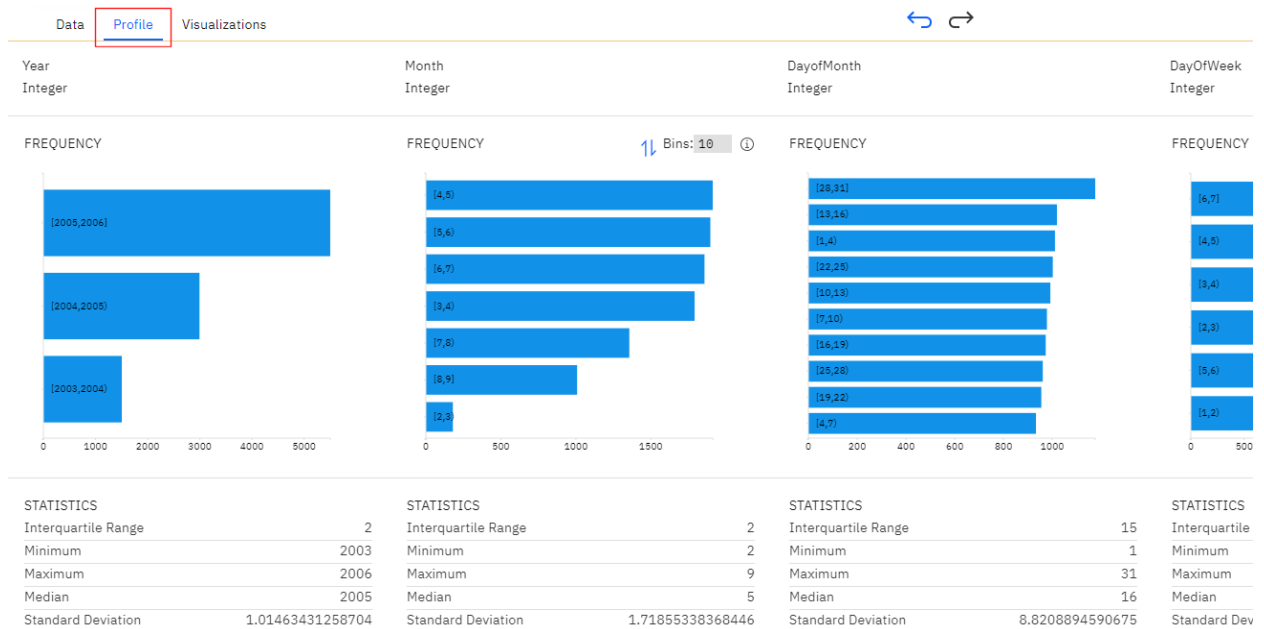
The `airline-data.csv` file is added to your project as a data asset.

3. Click the `airline-data.csv` data asset to preview its contents.
4. Click **Refine** to open a sample of the file in Data Refinery.

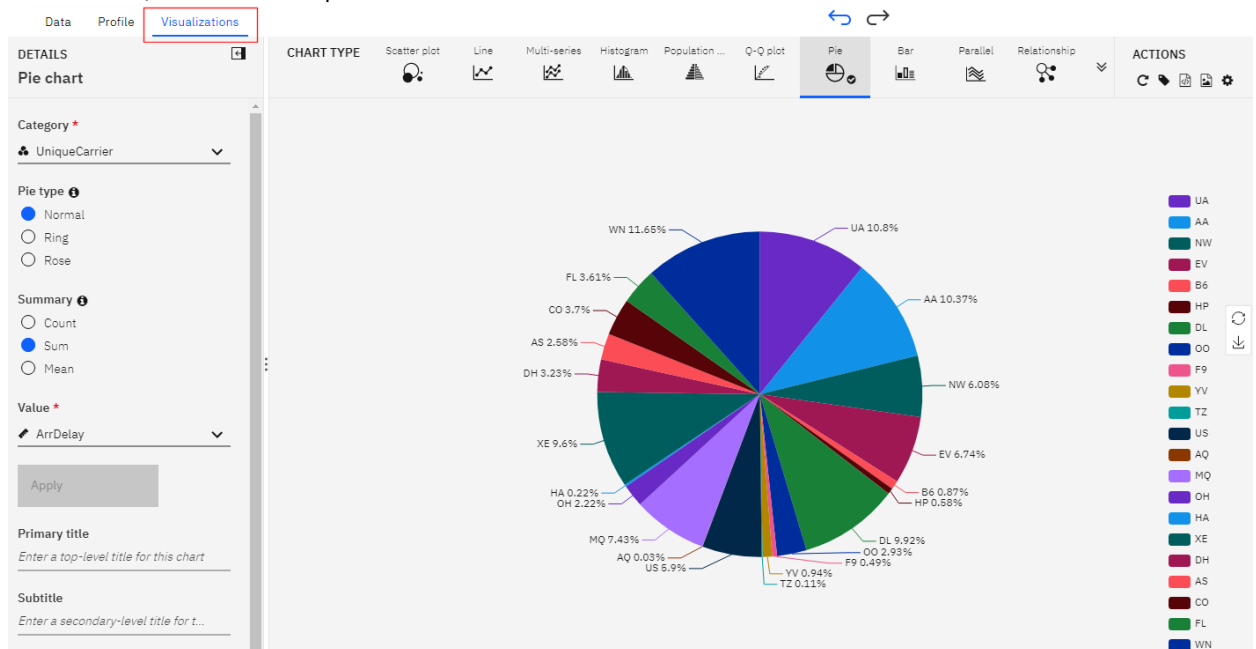
## Review the data with Profile and Visualizations

---

1. Click the **Profile** tab to review the [frequency distribution](#) of the data so that you can find the outliers. The statistics show the minimum, maximum, mean, and number of unique values in each column.



- Click the **Visualizations** tab. Select columns to visualize, then click **Visualize data**. Suggested charts have a blue dot next to their icons. Use the different perspectives available in the charts to identify patterns, connections, and relationships within the data.



**Tip:** Use the Profile and Visualizations pages to view changes in the data as you refine it.

## Data Refinery operations


Data Refinery uses two kinds of operations to refine data, *GUI operations* and *coding operations*. You will use both kinds of operations in this tutorial.

[My Projects](#) / [Airline analysis](#) / [airline-data.csv](#) / Refine data


Operation




Code an operation to cleanse and shape your data

- **GUI operations** can consist of multiple steps. Choose a GUI operation from the **Operation +** button. A subset of the GUI operations is also available from each column's actions menu .
- **Coding operations** are interactive templates for coding operations, functions, and logical operators. Most of the operations have interactive help. Click the operation name in the command-line text box to see the coding operations and their syntax options.

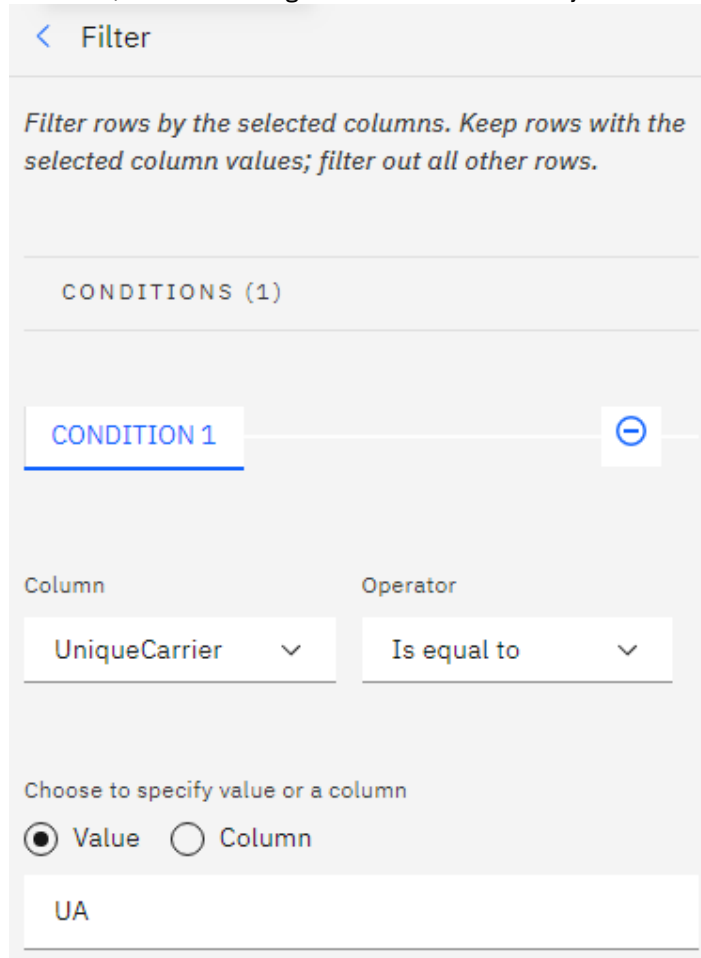
## Refine the data

Refining data is a series of steps to build a *Data Refinery flow*. As you go through this tutorial, view the **Steps** pane to follow your progress. You can select a step to delete or edit it. If you make a mistake, you can also click the Undo icon .

1. Go back to the **Data** tab.
2. Select the *Year* column. Click the actions menu  and choose **Sort descending**.
3. Focus on the delays for a specific airline:

This tutorial uses United Airlines (UA), but you can choose any airline.


1. Click **Operation +**, and then choose the GUI operation **Filter**.
2. Choose the *UniqueCarrier* column.
3. For Operator, choose *Is equal to*.
4. For Value, enter the string for the airline for which you want to see delay information. For example, UA.



< Filter

*Filter rows by the selected columns. Keep rows with the selected column values; filter out all other rows.*

CONDITIONS (1)

CONDITION 1 

Column	Operator
UniqueCarrier	Is equal to

Choose to specify value or a column

☒ Value ☐ Column

UA

5. Click **Apply**.
4. Create a new column that adds the arrival and departure delay times together.  
The **Convert column type** operation was automatically applied as the first step to convert the String data types in all the columns whose values are numbers to Integer data types.
  1. Click **Operation +**, and then choose the GUI operation **Calculate**.
  2. Choose the *ArrDelay* column, and click **Next**.
  3. For Operator, choose **Addition**.
  4. Choose to specify "Column," then choose the *DepDelay* column.
  5. Select **Create new column for results**.
  6. For **New column name**, enter *TotalDelay*.

< Calculate

[Change column selection](#)

Selected column: ArrDelay

*Perform a calculation with another column or with a specified value.*

Addition

Choose to specify value or a column

☐ Value ☒ Column

DepDelay

☒ Create new column for results ⓘ

TotalDelay

7. Click **Apply**.
- The new column, *TotalDelay*, is added to the end of the list of columns.
5. Move the new *TotalDelay* column to the beginning of the data set:
    1. In the command-line text box, choose the **select** operation.
    2. Click the word **select**, and then choose: `select(`<column>`, everything())`
    3. Click ``<column>``, and then choose the *TotalDelay* column.

When you finish, the command should look like this:

```
select(`TotalDelay`, everything())
```

4. Click **Apply**.

The *TotalDelay* column is now the first column.

6. Reduce the data to four columns: *Year*, *Month*, *DayofMonth*, and *TotalDelay*. Use the **group\_by** coding operation to divide the columns into groups of year, month, and day.

1. In the command-line text box, choose the **group\_by** operation.

2. Click `<column>`, and then choose the *Year* column.

3. Before the closing parenthesis, type: `,Month,DayofMonth`

When you finish, the command should look like this:

```
group_by(`Year`,Month,DayofMonth)
```

4. Click **Apply**.

5. Use the **select** coding operation for the *TotalDelay* column. In the command-line text box, select the **select** operation.

Click `<column>`, and choose the *TotalDelay* column.

The command should look like this:

```
select(`TotalDelay`)
```

6. Click **Apply**.

The shaped data now consists of the *Year*, *Month*, *DayofMonth*, and *TotalDelay* columns.

The first four rows of the data.

[My Projects](#) / [Airline analysis](#) / [airline-data.csv](#) / [Refine data](#)

Operation + Code an operation to cleanse and shape your data				
Data Profile Visualizations				
	Year Integer	Month Integer	DayofMonth Integer	TotalDelay Integer
1	2006	3	26	-14
2	2006	3	4	-7
3	2006	3	8	-7
4	2006	3	31	-6

7. Show the mean of the values of the *TotalDelay* column. Rename the *TotalDelay* column to *delay*:

1. Click **Operation +**, and then choose the GUI operation **Aggregate**.

2. Select the *TotalDelay* column, and click **Next**.

3. For **AGGREGATION 1**, select **Mean**.

4. For **Name of the aggregated column**, enter `delay`.

< Aggregate

Change column selection

Selected column: TotalDelay

☐ Group by columns

AGGREGATIONS (1)

AGGREGATION 1

Mean

delay

5. Click **Apply**.

The new column `delay` is the average of all the delay times.

The first four rows of the data.

[My Projects](#) / [Airline analysis](#) / [airline-data.csv](#) / Refine data

Operation + Code an operation to cleanse and shape your data

Data Profile Visualizations

	Year Integer	Month Integer	DayofMonth Integer	delay Decimal
1	2003	7	1	-14
2	2003	7	2	-18.5
3	2003	7	8	146.5
4	2003	7	11	-5

## Run the Data Refinery flow

When you run the Data Refinery flow, the steps are run on the entire data set. The output of the Data Refinery flow is added to the data assets in the project.

1. Click the Run Data Refinery flow icon .

The **DATA REFINERY FLOW DETAILS** and **DATA REFINERY FLOW OUTPUT** information panes are displayed.




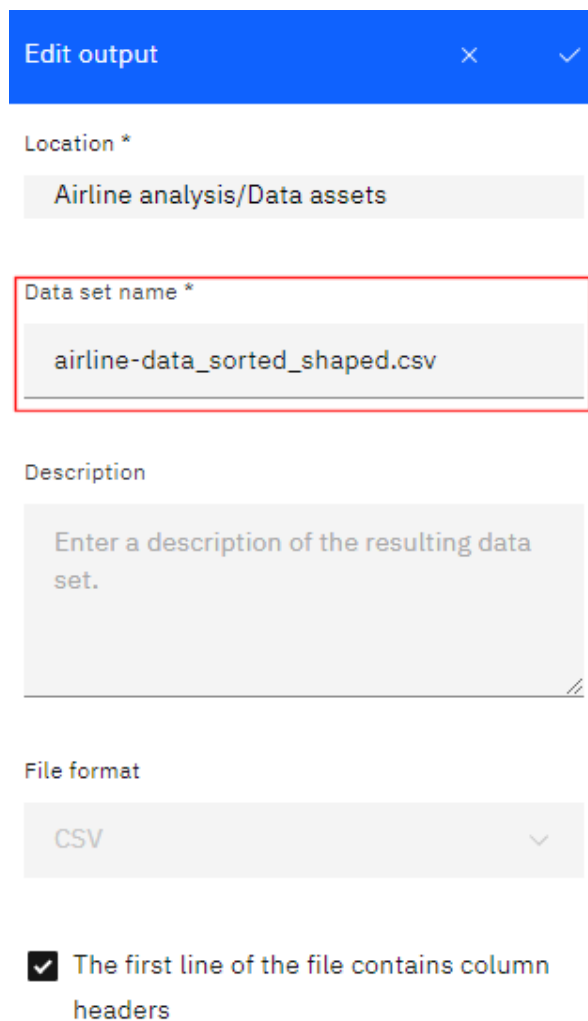
2. Click **Save and Run flow**.

The output of the Data Refinery flow, `airline-data_shaped.csv`, is added to the **Data assets** on the **Assets** page. The default name for the Data Refinery flow is `airline-data_flow`. It is added to the **Data Refinery flows** section.

## Create another data asset from the Data Refinery flow

---

1. On the Project page, click the Data Refinery flow, `airline-data_flow`, to open it in Data Refinery.
2. Sort the `delay` column in descending order. Select the `delay` column, click the column actions menu , and then select **Sort descending**.
3. In the **Details** pane, click **Edit**.
4. In the **DATA REFINERY FLOW OUTPUT** pane, click **Edit Output**. Change the **DATA SET NAME** to: `airline-data_sorted_shaped.csv`



**Edit output** × ✓

Location \*

Airline analysis/Data assets

Data set name \*

airline-data\_sorted\_shaped.csv

Description

Enter a description of the resulting data set.

File format

CSV

☒ The first line of the file contains column headers

5. Click the checkmark to save the change.
6. Click **Save and Run flow**.

## View the data assets and your Data Refinery flow in your project

---

1. When the Data Refinery flow completes, go to the project page.
2. Click the **Assets** tab.

3. Scroll down to **Data assets**. You'll see the original data set that you uploaded and the output of the two Data Refinery flows.

```
airline-data_sorted_shaped.csv  
airline-data_shaped.csv  
airline-data.csv
```

If you click the `airline-data_shaped.csv` data asset, you'll see the mean delay unsorted. Click `airline-data_sorted_shaped.csv` data asset to see the mean delay sorted in descending order.

The **Data Refinery flows** section shows the Data Refinery flow:

```
airline-data_flow
```

## Creating SPSS Modeler flows

---

With SPSS Modeler flows in Watson Studio, you can quickly develop predictive models using business expertise and deploy them into business operations to improve decision making. Designed around the long-established SPSS Modeler client software and the industry-standard CRISP-DM model it uses, the flows interface in supports the entire data mining process, from data to better business results.

Watson Studio offers a variety of modeling methods taken from machine learning, artificial intelligence, and statistics. The methods available on the node palette allow you to derive new information from your data and to develop predictive models. Each method has certain strengths and is best suited for particular types of problems.

Using the Flow Editor, you prepare or shape data, train or deploy a model, or transform data and export it back to a database table or a file. To create an SPSS model, add the Modeler flow asset type to your project, then select SPSS as the flow type.

An example project is installed with the product that includes example data and flows. See [Example projects](#).

Watch these short videos for a few modeling examples:

<https://www.ustream.tv/embed/recorded/127732173>

## Time Series Modeling in Watson Studio Desktop

<https://www.youtube.com/embed/EfnfKb8D-X8>



## Visual Classification Modeling in Watson Studio Desktop

<https://www.youtube.com/embed/DPBQASDQlrU>



### Data format

- Relational: Tables in relational data sources

- Tabular: Excel files (.xls, .xlsx), CSV files (.csv), or SPSS Statistics files (.sav). For Excel files, only the first sheet is read.

- Textual: In the supported relational tables or files

### Data size

- Any

### How can I prepare data?

- Use automatic data preparation functions

- Write SQL statements to manipulate data

- Cleanse, shape, sample, sort, and derive data

### How can I analyze data?

- Visualize data with many chart options

- Identify the natural language of a text field

### How can I build models?

- Build predictive models

- Choose from over 40 modeling algorithms, and many other nodes

- Use automatic modeling functions

- Model time series or geospatial data

- Classify textual data

- Identify relationships between the concepts in textual data

### Getting started

- To create an SPSS Modeler flow, click Add to project > Modeler flow. See the following information for more details.

Note: Watson Studio doesn't include SPSS functionality in Peru, Ecuador, Colombia, or Venezuela.

## Getting started with creating a flow

---


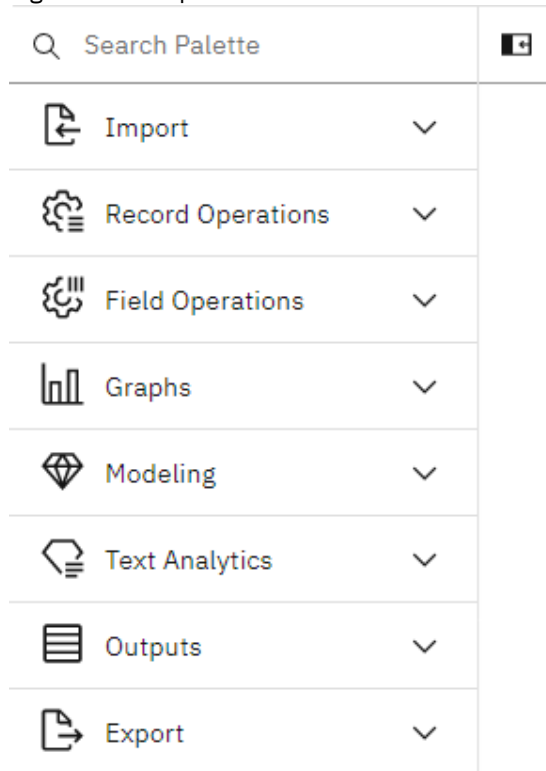
1. Open a project.
2. If your data isn't already part of the project, go the Assets tab, click Add to project, and add data to your project's data assets. These data file types are currently supported via the Data Asset import node: .csv, .txt, .json, .xls, .xlsx, and .sav.
3. While still on the Assets tab, click Add to project again and add a Modeler flow.
4. Type a name and description for your flow and click Create to create the new flow. Or you can use the From file tab to create a new flow based on an existing file you saved locally, or use the From example tab to open one of the available example flows.
5. Click the Palette icon (  ) to open the node palette, then drag nodes to the canvas as desired. Or you can import an existing SPSS Modeler stream file (.str). Many nodes are available. For a detailed description of each node, see [Nodes palette](#).

Figure 1. Node palette



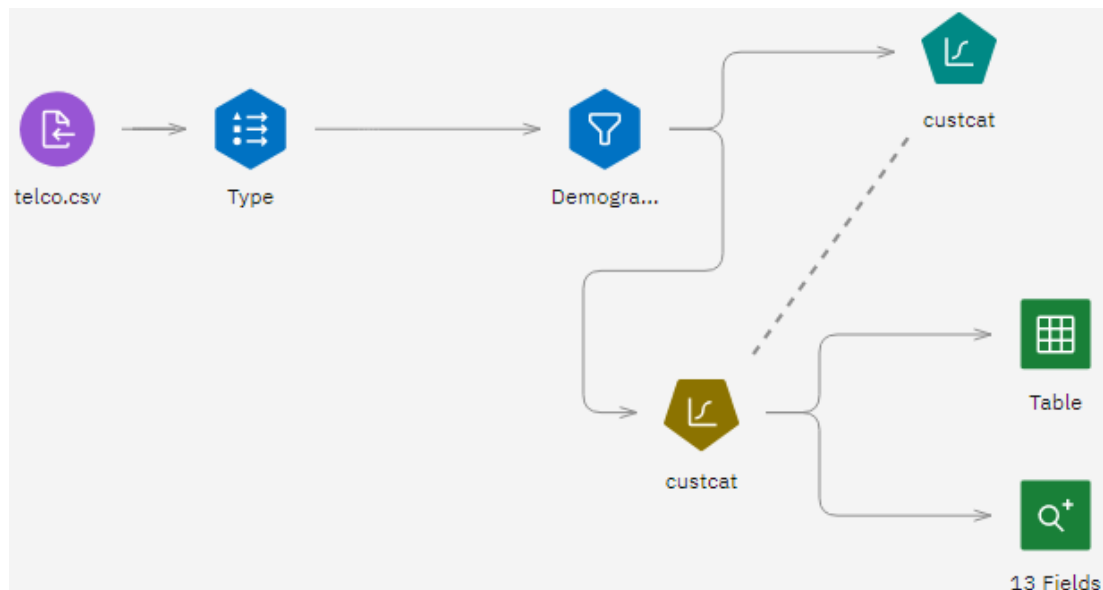
6. Double-click a node to set its properties.
7. To connect nodes, click the small empty circle icon on a node and drag on top of the node you want to connect it to.

Figure 2. Connecting nodes



8. Continue to add and connect nodes as desired to create your flow.

Figure 3. Creating a flow



9. Run your flow locally or, if you need more processing power, run it on the IBM Watson Machine Learning Server instead. You can also save and deploy your models to the server, if desired. See [Saving and running models on Watson Machine Learning Server](#).

## Options for building a flow

- You can import a stream (.str) that was created in SPSS Modeler Subscription or SPSS Modeler client. If the imported flow contains one or more import or export nodes, you'll be prompted to convert the nodes. See [Importing an SPSS Modeler stream](#).
- You can add data to your project to use as source or target nodes in a flow. See [Add data to a project](#).
- You can take a quick look at a portion of a flow's data by right-clicking a node and selecting Preview. Or, more thoroughly examine your data by using a Charts node to launch the chart builder and use advanced visualizations to explore your data from different perspectives and identify patterns, connections, and relationships within your data.
- You can group related nodes together into a supernode, which is represented by a star icon. Ctrl-click the desired nodes, right-click, and select Create supernode.
- You can run any terminal node without running the entire flow. Right-click the node and select Run.
- To view the results of an Outputs node (such as a Table node), run the node and then click the View outputs and versions icon (🔍). In the side panel, on the Outputs tab, click the object, such as a table, to open it.
- To save a version of a flow, click the View outputs and versions icon (🔍). In the side panel, on the Versions tab, save the version.
- After running a flow, you'll notice a new model nugget is generated. Model nuggets are gold in color. You can right-click the nugget and select View Model to explore the output.
- You can download a flow to your local machine as an SPSS Modeler stream file (.str) by clicking the Download stream icon (📄).
- In some cases, a task may fail to complete and you'll be prompted to continue running or restart your session. You can also force your session to restart at any time by clicking the Restart session button (🔄) in the right-hand Information panel.
- The Control Language for Expression Manipulation (CLEM) is a powerful language for analyzing and manipulating the data streams through your flows. Data miners use CLEM extensively in flow operations to perform tasks as simple as deriving profit from cost and revenue data or as complex as transforming web log data into a set of fields and records with usable information. The following CLEM functions are available for working with data in Watson Studio. You can enter these functions as code in the Expression field in the settings panel for various nodes, such as Derive and Set To Flag.

Table 1. CLEM functions for use with your data

Function type	Description
---------------	-------------

Function type	Description
Information	Used to gain insight into field values. For example, the function <code>is_string</code> returns <code>true</code> for all records whose type is a string.
Conversion	Used to construct new fields or convert storage type. For example, the function <code>to_timestamp</code> converts the selected field to a timestamp.
Comparison	Used to compare field values to each other or to a specified string. For example, <code>&lt;=</code> is used to compare whether the values of two fields are lesser or equal.
Logical	Used to perform logical operations, such as <code>if</code> , <code>then</code> , <code>else</code> operations.
Numeric	Used to perform numeric calculations, such as the natural log of field values.
Trigonometric	Used to perform trigonometric calculations, such as the arccosine of a specified angle.
Probability	Returns probabilities that are based on various distributions, such as probability that a value from Student's t distribution is less than a specific value.
Spatial	Used to perform spatial calculations on geospatial data.
Bitwise	Used to manipulate integers as bit patterns.
Random	Used to randomly select items or generate numbers.
String	Used to perform various operations on strings, such as <code>stripchar</code> , which allows you to remove a specified character.
SoundEx	Used to find strings when the precise spelling is not known; based on phonetic assumptions about how certain letters are pronounced.
Date and time	Used to perform various operations on date, time, and timestamp fields.
Sequence	Used to gain insight into the record sequence of a data set or perform operations that are based on that sequence.
Global	Used to access global values that are created by a Set Globals node. For example, <code>@MEAN</code> is used to refer to the mean average of all values for a field across the entire data set.
Blanks and null	Used to access, flag, and frequently fill user-specified blanks or system-missing values. For example, <code>@BLANK (FIELD)</code> is used to raise a true flag for records where blanks are present.
Special fields	Used to denote the specific fields under examination. For example, <code>@FIELD</code> is used when deriving multiple fields.

- **Nodes palette**

The following sections describe all the nodes available on the palette in Watson Studio. Drag-and-drop or double-click a node in the list to add it to your flow canvas. You can then double-click any node icon in your flow to set its properties. Hover over a property to see information about it, or click the information icon to see Help.

- **Working with your data**

To see a quick sample of a flow's data, right-click a node and select Preview. To more thoroughly examine your data, use a Charts node to launch the chart builder.

- **Saving and running models on Watson Machine Learning Server**

With Watson Studio Desktop, you can run your models on a Watson Machine Learning Server for better performance.

- **Flow scripting**

Scripts can be used to customize operations within a particular flow, and they are saved with that flow. For example, you might use a script to specify a particular run order for the terminal nodes within a flow. You use the flow properties panel to edit the script that is saved with the current flow.

- **SQL optimization**

You can push many data preparation and mining operations directly in your database to improve performance.

- **Disabling or caching nodes in a flow**

You can disable a node so it's ignored when the flow runs. And you can set up a cache on a node.

- **Importing an SPSS Modeler stream**  
You can import a stream (.str) that was created in SPSS Modeler Subscription or SPSS Modeler client.
- **Extension nodes**  
SPSS Modeler supports the languages R and Apache Spark (via Python).
- **Setting properties for flows**  
You can specify a number of properties to apply to the current flow.
- **Expression Builder**  
You can type CLEM expressions manually or use the Expression Builder, which displays a complete list of CLEM functions and operators as well as data fields from the current flow, allowing you to quickly build expressions without memorizing the exact names of fields or functions.

## Nodes palette

---

The following sections describe all the nodes available on the palette in Watson Studio. Drag-and-drop or double-click a node in the list to add it to your flow canvas. You can then double-click any node icon in your flow to set its properties. Hover over a property to see information about it, or click the information icon to see Help.

When first creating a flow, you select which runtime to use. By default, the flow will use the IBM SPSS Modeler runtime. If you want to use native Spark algorithms instead of SPSS algorithms, select the Spark runtime. Properties for some nodes will vary depending on which runtime option you choose.

- **Import**  
Use Import nodes to import data stored in various formats, or to generate your own synthetic data.
- **Record Operations**  
Record Operations nodes are useful for making changes to data at the record level. These operations are important during the *data understanding* and *data preparation* phases of data mining because they allow you to tailor the data to your particular business need.
- **Field Operations**  
After an initial data exploration, you will probably need to select, clean, or construct data in preparation for analysis. The Field Operations palette contains many nodes useful for this transformation and preparation.
- **Graphs**  
Several phases of the data mining process use graphs and charts to explore data brought in to Watson Studio.
- **Modeling**  
Watson Studio offers a variety of modeling methods taken from machine learning, artificial intelligence, and statistics.
- **Text Analytics**  
The Text Analytics nodes offer powerful text analytic capabilities, which use advanced linguistic technologies and Natural Language Processing (NLP) to rapidly process a large variety of unstructured text data and, from this text, extract and organize the key concepts. Text Analytics can also group these concepts into categories.
- **Outputs**  
Output nodes provide the means to obtain information about your data and models. They also provide a mechanism for exporting data in various formats to interface with your other software tools.
- **Export**  
Export nodes provide a mechanism for exporting data in various formats to interface with your other software tools.

**Parent topic:** [Creating SPSS Modeler flows](#)

## Import

---

Use Import nodes to import data stored in various formats, or to generate your own synthetic data.

- **Data Asset node**  
You can use the Data Asset import node to pull in data from remote data sources using connections or from your local computer. First, you must create the connection. See [Adding connections to projects](#).



- **User Input node**  
The User Input node provides an easy way for you to create synthetic data--either from scratch or by altering existing data. This is useful, for example, when you want to create a test dataset for modeling.
- **Sim Gen node**  
The Simulation Generate node provides an easy way to generate simulated data, either without historical data using user specified statistical distributions, or automatically using the distributions obtained from running a Simulation Fitting node on existing historical data. Generating simulated data is useful when you want to evaluate the outcome of a predictive model in the presence of uncertainty in the model inputs.
- **Extension Import node**  
With the Extension Import node, you can run R scripts or Python for Spark scripts to import data.

**Parent topic:** [Nodes palette](#)

## Data Asset node

You can use the Data Asset import node to pull in data from remote data sources using connections or from your local computer. First, you must create the connection. See [Adding connections to projects](#).

Note for connections to a Planning Analytics database, you must choose a view (not a cube).

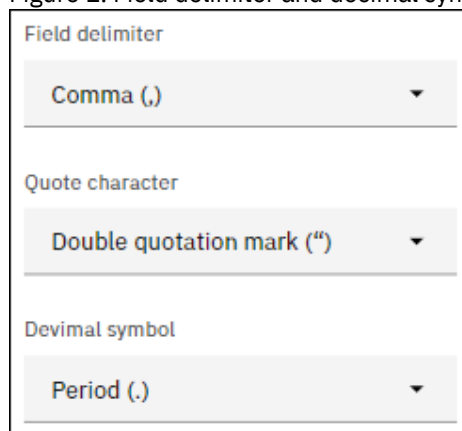
You can also pull in data from a local data file (.csv, .txt, .json, .xls, .xlsx, and .sav are supported). Only the first sheet is imported from spreadsheets. In the node's properties, under DATA, select one or more data files to upload. You can also simply drag-and-drop the data file from your local file system onto your canvas.

Note: You can import a stream (.str) into Watson Studio that was created in SPSS Modeler Subscription or SPSS Modeler client. If the imported stream contains one or more import or export nodes, you'll be prompted to convert the nodes. See [Importing an SPSS Modeler stream](#).

## Setting the field delimiter, quote character, and decimal symbol

Different countries use different symbols to separate the integer part from the fractional part of a number and to separate fields in data. For example, you might use a comma instead of a period to separate the integer part from the fractional part of numbers. And, rather than using commas to separate fields in your data, you might use colons or tabs. With a Data Asset import or export node, you can specify these symbols for field delimiter and decimal. Double-click the node to open its properties and select the Field delimiter and the Decimal symbol as desired. Available delimiters are Comma, Tab, Colon, or Other. Select Other if you need to specify your own custom delimiter.

Figure 1. Field delimiter and decimal symbol options



The image shows a screenshot of the 'Data Asset' node properties in Watson Studio. It contains three dropdown menus:

- Field delimiter:** The dropdown menu is open, showing 'Comma (,)' as the selected option.
- Quote character:** The dropdown menu is open, showing 'Double quotation mark (")' as the selected option.
- Devimal symbol:** The dropdown menu is open, showing 'Period (.)' as the selected option.

## Importing data from an SPSS Statistics file

If you import data from an SPSS Statistics file (.sav), the following options are available:

Variable names. Select a method of handling variable names and labels upon import from an SPSS Statistics .sav file. Metadata that you choose to include here persists throughout your work in SPSS Modeler and may be exported again for use in IBM SPSS Statistics.

- Read names and labels. Select to read both variable names and labels into SPSS Modeler. This is the default option, and variable names are displayed in the Type node. Labels may be displayed in charts, model browsers, and other types of output. By default, the display of labels in output is disabled.
- Read labels as names. Select to read the descriptive variable labels from the SPSS Statistics .sav file rather than the short field names, and use these labels as variable names in SPSS Modeler.

Values. Select a method of handling values and labels upon import from an SPSS Statistics .sav file. Metadata that you choose to include here persists throughout your work in SPSS Modeler and may be exported again for use in SPSS Statistics.

- Read data and labels. Select to read both actual values and value labels into SPSS Modeler. This is the default option, and values themselves are displayed in the Type node. Value labels may be displayed in the Expression Builder, charts, model browsers, and other types of output.
- Read labels as data. Select if you want to use the value labels from the .sav file rather than the numerical or symbolic codes used to represent the values. For example, selecting this option for data with a gender field whose values of 1 and 2 actually represent *male* and *female*, respectively, will convert the field to a string and import `male` and `female` as the actual values.

It's important to consider missing values in your SPSS Statistics data before selecting this option. For example, if a numeric field uses labels only for missing values (0 = *No Answer*, 1 = *Unknown*), then selecting the Read labels as data option will import only the value labels No Answer and Unknown and will convert the field to a string. In such cases, you should import the values themselves and set missing values in a Type node.

Use field format information to determine storage. If you deselect this option, field values that are formatted in the .sav file as integers (i.e., fields specified as `Fn.0` in the Variable View in IBM SPSS Statistics) are imported using integer storage. All other field values except strings are imported as real numbers.

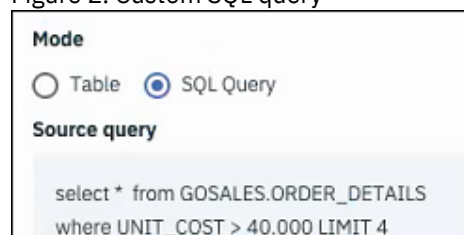
If you select this option (default), all field values except strings are imported as real numbers, whether formatted in the .sav file as integers or not.

Read timestamp as date. By default, all timestamp values are shown as dates. Deselect this option to override this behavior.

## Using SQL to pull in data

In the Data Asset import node properties, under Mode, you can select SQL Query if you want to use custom SQL to import data from a database. Use a SQL `SELECT` statement to pull in rows or columns of data from a database. Note that the Source path field doesn't apply if you're using the SQL Query mode.

Figure 2. Custom SQL query



The screenshot shows a dialog box with the title "Mode". There are two radio buttons: "Table" and "SQL Query". The "SQL Query" radio button is selected. Below the radio buttons is a text area labeled "Source query" containing the SQL statement: `select * from GOSALES.ORDER_DETAILS where UNIT_COST > 40.000 LIMIT 4`.

The following example pulls in certain rows of data from a database table:

```
select * from GOSALES.ORDER_DETAILS
where UNIT_COST > 40,000 LIMIT 4
```

And this example pulls in certain columns of data from a database table:

```
select QUANTITY, UNIT_COST, UNIT_PRICE from GOSALES.ORDER_DETAILS
```

Note that the SQL syntax you use can vary depending on database platform. For example, if pulling in data from an Informix database, Informix requires field names to be surrounded by double quotes. For example:

```
select "Age", "Sex" from testuser.canvas_drug
```

This SQL feature should only be used to pull in data. Use caution so as not to manipulate the data in your database.

The following databases currently support this custom SQL feature:

- Amazon Redshift
- Apache Hive
- Cloudera Impala
- Compose for PostgreSQL
- Db2 on Cloud
- Db2 Warehouse
- Google BigQuery
- Informix
- Microsoft SQL Server
- MySQL
- Netezza
- Oracle
- Pivotal Greenplum
- Salesforce.com
- Snowflake
- Sybase
- Sybase IQ
- Teradata

## Oracle

---

When using the Data Asset node to pull in data from an Oracle database, you may encounter errors. To work around this issue, perform the following steps:

1. Add or modify the following line in `sqlnet.ora` of your Oracle database.

```
SQLNET.ALLOWED_LOGON_VERSION_SERVER=8
```

2. Restart the database.
3. Create a new user and set a password for it, or change the password of an existing user.
4. Use the following query to verify that the user has 10G in its `PASSWORD_VERSIONS`. If it doesn't, add it.

```
select USERNAME,ACCOUNT_STATUS,PASSWORD_VERSIONS from dba_users;
```

5. In `sqlnet.ora`, change the value from 8 to 11 in the following line.

```
SQLNET.ALLOWED_LOGON_VERSION_SERVER=11
```

6. Verify that the user can still connect to the database.

**Parent topic:** [Import](#)

## User Input node

---

The User Input node provides an easy way for you to create synthetic data--either from scratch or by altering existing data. This is useful, for example, when you want to create a test dataset for modeling.

### Creating data from scratch

---

The User Input node is available in the nodes palette under Import and can be added directly to the flow canvas.

1. Open the nodes palette and click the Import category.
2. Drag and drop or double-click to add the User Input node to the flow canvas.
3. Double-click to open the node's settings and specify fields and values.

Notes:

- User Input nodes selected from the nodes palette will be completely blank, with no fields and no data information. This allows you to create synthetic data entirely from scratch.
- In the node properties, under Define Fields, Add Values, to enter an array you must enter each value on a separate line.

**Parent topic:** [Import](#)

## Sim Gen node

The Simulation Generate node provides an easy way to generate simulated data, either without historical data using user specified statistical distributions, or automatically using the distributions obtained from running a Simulation Fitting node on existing historical data. Generating simulated data is useful when you want to evaluate the outcome of a predictive model in the presence of uncertainty in the model inputs.

### Creating data without historical data

The Simulation Generate node is available from the nodes palette and can be added directly to your flow.

1. Click the Import tab of the nodes palette.
2. Drag and drop or double-click the Sim Gen node to add it to your flow.
3. Double-click the node to open its properties and specify fields, storage types, statistical distributions, and distribution parameters.

Note: After you add a Simulation Generate node to your flow it will be completely blank, with no fields and no distribution information. This enables you to entirely create simulated data without historical data.

### Generating simulated data using existing historical data

You can also create a Simulation Generate node by running a Simulation Fitting terminal node:

1. Right-click a Sim Fit node in your flow and choose Run.
2. The Simulation Generate node appears on the flow canvas with an update link to the Simulation Fitting node.
3. When generated, the Simulation Generate node inherits all the fields, storage types, and statistical distribution information from the Simulation Fitting node.

### Defining an update link to a Sim Fit node

You can create a link between a Simulation Generate node and a Simulation Fitting node. This is useful if you want to update one or more fields with the information of the best fitting distribution, determined by fitting to historical data.

1. Right-click the Simulation Generate node and choose Define Update Link.
2. Click another node. If this node is a Simulation Fitting node, a link is made. If this node is not a Simulation Fitting node, no link is made.

When you use a Simulation Fitting node to update a linked Simulation Generate node, the result depends on whether the same fields are present in both nodes, and if the fields are unlocked in the Simulation Generate node. The results of updating a Simulation Fitting node are shown in the following table.

Table 1. Results of updating a Simulation Fitting node

	Field in Simulation	Fitting node
--	---------------------	--------------

	Field in Simulation	Fitting node
Field in Simulation Generate node	Present	Missing
Present and unlocked.	Field is overwritten.	Field is deleted.
Missing.	Field is added.	No change.
Present and locked.	The distribution of the field is not overwritten. The information in the Fit Details dialog box and the correlations are updated.	The field is not overwritten. The correlations are set to zero.
Do not clear Min/Max when refitting option is selected.	The field is overwritten, apart from	the values in the Min, Max column.
Do not recalculate correlations when refitting option is selected.	If the field is unlocked, it is overwritten.	The correlations are not overwritten.

## Removing an update link to a Sim Fit node

You can remove a link between a Simulation Generate node and a Simulation Fitting node by right-clicking the Simulation Generate node and selecting Remove Update Link.

**Parent topic:** [Import](#)

## Extension Import node

With the Extension Import node, you can run R scripts or Python for Spark scripts to import data.

After adding the node to your canvas, double-click the node to open its properties.

### Syntax tab

Select your type of syntax - R or Python for Spark. Then enter or paste your custom script for importing data. When your syntax is ready, you can run the node.

### Console Output tab

The Console Output tab contains any output that's received when the R script or Python for Spark script runs (for example, if using an R script, it shows output received from the R console when the R script in the R Syntax field on the Syntax tab is executed). This output might include R or Python error messages or warnings that are produced when the R or Python script is executed. The output can be used, primarily, to debug the script. The Console Output tab also contains the script from the R Syntax or Python Syntax field.

Every time the Extension Import script runs, the content of the Console Output tab is overwritten with the output received from the R console or Python for Spark. You can't edit the output.

## Filtering or renaming fields

You can rename or exclude fields at any point in a flow. For example, as a medical researcher, you may not be concerned about the potassium level (field-level data) of patients (record-level data); therefore, you can filter out the K (potassium) field.

- Using a Filter node, you can rename or filter fields at any point in the flow
- You can use a Filter node to map fields from one import node to another

## Viewing and setting information about types

From the Type node, you can specify field metadata and properties that are invaluable to modeling and other work. These properties include:

- Specifying a usage type, such as range, set, ordered set, or flag, for each field in your data
- Setting options for handling missing values and system nulls
- Setting the role of a field for modeling purposes
- Specifying values for a field and options used to automatically read values from your data
- Specifying value labels

**Parent topic:** [Import](#)

## Record Operations

---

Record Operations nodes are useful for making changes to data at the record level. These operations are important during the *data understanding* and *data preparation* phases of data mining because they allow you to tailor the data to your particular business need.

For example, based on the results of a data audit conducted using the Data Audit node (Outputs palette), you might decide that you would like to merge customer purchase records for the past three months. Using a Merge node, you can merge records based on the values of a key field, such as `Customer ID`. Or you might discover that a database containing information about web site hits is unmanageable with over one million records. Using a Sample node, you can select a subset of data for use in modeling.

- **Select node**  
You can use Select nodes to select or discard a subset of records from the data stream based on a specific condition, such as `BP (blood pressure) = "HIGH"`.
- **Sample node**  
You can use Sample nodes to select a subset of records for analysis, or to specify a proportion of records to discard. A variety of sample types are supported, including stratified, clustered, and nonrandom (structured) samples.
- **Sort node**  
You can use Sort nodes to sort records into ascending or descending order based on the values of one or more fields. For example, Sort nodes are frequently used to view and select records with the most common data values. Typically, you would first aggregate the data using the Aggregate node and then use the Sort node to sort the aggregated data into descending order of record counts. Displaying these results in a table will allow you to explore the data and to make decisions, such as selecting the records of the top 10 best customers.
- **Balance node**  
You can use Balance nodes to correct imbalances in datasets so they conform to specified test criteria.
- **Distinct node**  
Duplicate records in a data set must be removed before data mining can begin. For example, in a marketing database, individuals may appear multiple times with different address or company information. You can use the Distinct node to find or remove duplicate records in your data, or to create a single, composite record from a group of duplicate records.
- **Aggregate node**  
Aggregation is a data preparation task frequently used to reduce the size of a dataset. Before proceeding with aggregation, you should take time to clean the data, concentrating especially on missing values. Once you have aggregated, potentially useful information regarding missing values may be lost.
- **Merge node**  
The function of a Merge node is to take multiple input records and create a single output record containing all or some of the input fields. This is a useful operation when you want to merge data from different sources, such as internal customer data and purchased demographic data.
- **Append node**  
You can use Append nodes to concatenate sets of records. Unlike Merge nodes, which join records from different sources together, Append nodes read and pass downstream all of the records from one source until there are no more. Then the records from the next source are read using the same data structure (number of

records, number of fields, and so on) as the first, or primary, input. When the primary source has more fields than another input source, the system null string (\$null\$) will be used for any incomplete values.

- **Streaming Time Series node**

You use the Streaming Time Series node to build and score time series models in one step. A separate time series model is built for each target field, however model nuggets are not added to the generated models palette and the model information cannot be browsed.

- **SMOTE node**

The Synthetic Minority Over-sampling Technique (SMOTE) node provides an over-sampling algorithm to deal with imbalanced data sets. It provides an advanced method for balancing data. The SMOTE node in Watson Studio is implemented in Python and requires the imbalanced-learn Python library.

- **RFM Aggregate node**

The Recency, Frequency, Monetary (RFM) Aggregate node allows you to take customers' historical transactional data, strip away any unused data, and combine all of their remaining transaction data into a single row (using their unique customer ID as a key) that lists when they last dealt with you (recency), how many transactions they have made (frequency), and the total value of those transactions (monetary).

- **Space-Time-Boxes node**

Space-Time-Boxes (STB) are an extension of Geohashed spatial locations. More specifically, an STB is an alphanumeric string that represents a regularly shaped region of space and time.

- **Extension Transform node**

With the Extension Transform node, you can take data from an SPSS Modeler flow and apply transformations to the data using R scripting or Python for Spark scripting.

- **CPLEX Optimization node**

With the CPLEX Optimization node, you can use complex mathematical (CPLEX) based optimization via an Optimization Programming Language (OPL) model file.

**Parent topic:** [Nodes palette](#)

## Select node

---

You can use Select nodes to select or discard a subset of records from the data stream based on a specific condition, such as `BP (blood pressure) = "HIGH"`.

Mode. Specifies whether records that meet the condition will be included or excluded from the data stream.

- Include. Select to include records that meet the selection condition.
- Discard. Select to exclude records that meet the selection condition.

Condition. Displays the selection condition that will be used to test each record, which you specify using a CLEM expression. Either enter an expression in the window or use the Expression Builder by clicking the calculator (Expression Builder) button to the right of the window.

If you choose to discard records based on a condition, such as the following:

```
(var1='value1' and var2='value2')
```

the Select node by default also discards records having null values for all selection fields. To avoid this, append the following condition to the original one:

```
and not (@NULL(var1) and @NULL(var2))
```

Select nodes are also used to choose a proportion of records. Typically, you would use a different node, the Sample node, for this operation. However, if the condition you want to specify is more complex than the parameters provided, you can create your own condition using the Select node. For example, you can create a condition such as:

```
BP = "HIGH" and random(10) <= 4
```

This will select approximately 40% of the records showing high blood pressure and pass those records downstream for further analysis.

## Sample node

---

You can use Sample nodes to select a subset of records for analysis, or to specify a proportion of records to discard. A variety of sample types are supported, including stratified, clustered, and nonrandom (structured) samples.

Sampling can be used for several reasons:

- To improve performance by estimating models on a subset of the data. Models estimated from a sample are often as accurate as those derived from the full dataset, and may be more so if the improved performance allows you to experiment with different methods you might not otherwise have attempted.
- To select groups of related records or transactions for analysis, such as selecting all the items in an online shopping cart (or market basket), or all the properties in a specific neighborhood.
- To identify units or cases for random inspection in the interest of quality assurance, fraud prevention, or security.

Note: If you simply want to partition your data into training and test samples for purposes of validation, a Partition node can be used instead. See [Partition node](#) for more information.

## Types of samples

---

**Clustered samples.** Sample groups or clusters rather than individual units. For example, suppose you have a data file with one record per student. If you cluster by school and the sample size is 50%, then 50% of schools will be chosen and all students from each selected school will be picked. Students in unselected schools will be rejected. On average, you would expect about 50% of students to be picked, but because schools vary in size, the percentage may not be exact. Similarly, you could cluster shopping cart items by transaction ID to make sure that all items from selected transactions are maintained.

**Stratified samples.** Select samples independently within non-overlapping subgroups of the population, or strata. For example, you can ensure that men and women are sampled in equal proportions, or that every region or socioeconomic group within an urban population is represented. You can also specify a different sample size for each stratum (for example, if you think that one group has been under-represented in the original data).

**Systematic or 1-in-n sampling.** When selection at random is difficult to obtain, units can be sampled systematically (at a fixed interval) or sequentially.

**Sampling weights.** Sampling weights are automatically computed while drawing a complex sample and roughly correspond to the "frequency" that each sampled unit represents in the original data. Therefore, the sum of the weights over the sample should estimate the size of the original data.

## Sampling frame

---

A sampling frame defines the potential source of cases to be included in a sample or study. In some cases, it may be feasible to identify every single member of a population and include any one of them in a sample--for example, when sampling items that come off a production line. More often, you will not be able to access every possible case. For example, you cannot be sure who will vote in an election until after the election has happened. In this case, you might use the electoral register as your sampling frame, even though some registered people won't vote, and some people may vote despite not having been listed at the time you checked the register. Anybody not in the sampling frame has no prospect of being sampled. Whether your sampling frame is close enough in nature to the population you are trying to evaluate is a question that must be addressed for each real-life case.

Parent topic: [Record Operations](#)

## Sort node

---



You can use Sort nodes to sort records into ascending or descending order based on the values of one or more fields. For example, Sort nodes are frequently used to view and select records with the most common data values. Typically, you would first aggregate the data using the Aggregate node and then use the Sort node to sort the aggregated data into descending order of record counts. Displaying these results in a table will allow you to explore the data and to make decisions, such as selecting the records of the top 10 best customers.

The following settings are available for the Sort node

Sort by. All fields selected to use as sort keys are displayed in a table. A key field works best for sorting when it is numeric.

- Add fields to this list using the Field Chooser button on the right.
- Select an order by clicking the Ascending or Descending arrow in the table's *Order* column.
- Delete fields using the red delete button.
- Sort directives using the up and down arrow buttons.

Default sort order. Select either Ascending or Descending to use as the default sort order when new fields are added above.

Note: The Sort node is not applied if there is a Distinct node down the model flow. For information about the Distinct node, see [Distinct node](#).

**Parent topic:** [Record Operations](#)

## Balance node

---

You can use Balance nodes to correct imbalances in datasets so they conform to specified test criteria.

For example, suppose that a dataset has only two values--*low* or *high*--and that 90% of the cases are *low* while only 10% of the cases are *high*. Many modeling techniques have trouble with such biased data because they will tend to learn only the *low* outcome and ignore the *high* one, since it is more rare. If the data is well balanced with approximately equal numbers of *low* and *high* outcomes, models will have a better chance of finding patterns that distinguish the two groups. In this case, a Balance node is useful for creating a balancing directive that reduces cases with a *low* outcome.

Balancing is carried out by duplicating and then discarding records based on the conditions you specify. Records for which no condition holds are always passed through. Because this process works by duplicating and/or discarding records, the original sequence of your data is lost in downstream operations. Be sure to derive any sequence-related values before adding a Balance node to the data stream.

**Parent topic:** [Record Operations](#)

## Distinct node

---

Duplicate records in a data set must be removed before data mining can begin. For example, in a marketing database, individuals may appear multiple times with different address or company information. You can use the Distinct node to find or remove duplicate records in your data, or to create a single, composite record from a group of duplicate records.

To use the Distinct node, you must first define a set of key fields that determine when two records are considered to be duplicates.

If you do not pick all your fields as key fields, then two "duplicate" records may not be truly identical because they can still differ in the values of the remaining fields. In this case, you can also define a sort order that is applied within each group of duplicate records. This sort order gives you fine control over which record is treated as the first within a group. Otherwise, all duplicates are considered to be interchangeable and any record might be selected. The incoming order of the records is not taken into account, so it does not help to use an upstream Sort node (see "Sorting records within the distinct node" below).

Mode. Specify whether to create a composite record, or to either include or exclude (discard) the first record.

- Create a composite record for each group. Provides a way for you to aggregate non-numeric fields. Selecting this option makes the Composite tab available where you specify how to create the composite records.
- Include only the first record in each group. Selects the first record from each group of duplicate records and discards the rest. The *first* record is determined by the sort order defined below, and not by the incoming order of the records.
- Discard only the first record in each group. Discards the first record from each group of duplicate records and selects the remainder instead. The *first* record is determined by the sort order defined below, and not by the incoming order of the records. This option is useful for *finding* duplicates in your data so that you can examine them later in the flow.

Key fields for grouping. Lists the field or fields used to determine whether records are identical. You can:

- Add fields to this list using the field picker button on the right.
- Delete fields from the list by using the red X (remove) button.

Within groups, sort records by. Lists the fields used to determine how records are sorted within each group of duplicates, and whether they are sorted in ascending or descending order. You can:

- Add fields to this list using the field picker button on the right.
- Delete fields from the list by using the red X (remove) button.
- Move fields using the up or down buttons, if you are sorting by more than one field.

You must specify a sort order if you have chosen to include or exclude the first record in each group, and it matters to you which record is treated as the first.

You may also want to specify a sort order if you have chosen to create a composite record, for certain options on the Composite tab.

Specify whether, by default, records are sorted in Ascending or Descending order of the sort key values.

## Sorting records within the Distinct node

---

If the order of records within a group of duplicates is important to you, then you must specify the order using the Within groups, sort records by option in the Distinct node. Do not rely on an upstream Sort node. Remember that the incoming order of the records is not taken into account -- only the order specified within the node.

If you do not specify any sort fields (or you specify insufficient sort fields), then the records within each group of duplicates will be unordered (or incompletely ordered) and the results may be unpredictable.

For example, assume we have a very large set of log records pertaining to a number of machines. The log contains data such as the following:

Table 1. Machine log data

Timestamp	Machine	Temperature
17:00:22	Machine A	31
13:11:30	Machine B	26
16:49:59	Machine A	30
18:06:30	Machine X	32
16:17:33	Machine A	29
19:59:04	Machine C	35
19:20:55	Machine Y	34
15:36:14	Machine X	28
12:30:41	Machine Y	25
14:45:49	Machine C	27
19:42:00	Machine B	34

Timestamp	Machine	Temperature
20:51:09	Machine Y	36
19:07:23	Machine X	33

To reduce the number of records down to the latest record for each machine, use `Machine` as the key field and use `Timestamp` as the sort field (in descending order). The input order does not affect the result because the sort selection specifies which of the many rows for a given Machine is to be returned, and the final data output would be as follows.

Table 2. Sorted machine log data

Timestamp	Machine	Temperature
17:00:22	Machine A	31
19:42:00	Machine B	34
19:59:04	Machine C	35
19:07:23	Machine X	33
20:51:09	Machine Y	36

Parent topic: [Record Operations](#)

## Aggregate node

Aggregation is a data preparation task frequently used to reduce the size of a dataset. Before proceeding with aggregation, you should take time to clean the data, concentrating especially on missing values. Once you have aggregated, potentially useful information regarding missing values may be lost.

You can use an Aggregate node to replace a sequence of input records with summary, aggregated output records. For example, you might have a set of input sales records such as those shown in the following table.

Table 1. Sales record input example

Age	Sex	Region	Branch	Sales
23	M	S	8	4
45	M	S	16	4
37	M	S	8	5
30	M	S	5	7
44	M	N	4	9
25	M	N	2	11
29	F	S	16	6
41	F	N	4	8
23	F	N	6	2
45	F	N	4	5
33	F	N	6	10

You can aggregate these records with `Sex` and `Region` as key fields. Then choose to aggregate `Age` with the mode Mean and `Sales` with the mode Sum. Select the Include record count in field aggregate node option and your aggregated output will be similar to the following table.

Table 2. Aggregated record example

Age (mean)	Sex	Region	Sales (sum)	Record Count
35.5	F	N	25	4
29	F	S	6	1
34.5	M	N	20	2

Age (mean)	Sex	Region	Sales (sum)	Record Count
33.75	M	S	20	4

From this you learn, for example, that the average age of the four female sales staff in the North region is 35.5, and the sum total of their sales was 25 units.

Note: Fields such as `Branch` are automatically discarded when no aggregate mode is specified.

**Parent topic:** [Record Operations](#)

## Merge node

---

The function of a Merge node is to take multiple input records and create a single output record containing all or some of the input fields. This is a useful operation when you want to merge data from different sources, such as internal customer data and purchased demographic data.

You can merge data in the following ways.

- Merge by Order concatenates corresponding records from all sources in the order of input until the smallest data source is exhausted. It is important if using this option that you have sorted your data using a Sort node.
- Merge using a Key field, such as `Customer ID`, to specify how to match records from one data source with records from the other(s). Several types of joins are possible, including inner join, full outer join, partial outer join, and anti-join.
- Merge by Condition means that you can specify a condition to be satisfied for the merge to take place. You can specify the condition directly in the node, or build the condition using the Expression Builder.
- Merge by Ranked Condition is a left sided outer join in which you specify a condition to be satisfied for the merge to take place and a ranking expression which sorts into order from low to high. Most often used to merge geospatial data, you can specify the condition directly in the node, or build the condition using the Expression Builder.

**Parent topic:** [Record Operations](#)

## Append node

---

You can use Append nodes to concatenate sets of records. Unlike Merge nodes, which join records from different sources together, Append nodes read and pass downstream all of the records from one source until there are no more. Then the records from the next source are read using the same data structure (number of records, number of fields, and so on) as the first, or primary, input. When the primary source has more fields than another input source, the system null string (`$null$`) will be used for any incomplete values.

Append nodes are useful for combining datasets with similar structures but different data. For example, you might have transaction data stored in different files for different time periods, such as a sales data file for March and a separate one for April. Assuming that they have the same structure (the same fields in the same order), the Append node will join them together into one large file, which you can then analyze.

Note: To append files, the field measurement levels must be similar. For example, a `Nominal` field cannot be appended with a field whose measurement level is `Continuous`.

**Parent topic:** [Record Operations](#)

## Streaming Time Series node

---

You use the Streaming Time Series node to build and score time series models in one step. A separate time series model is built for each target field, however model nuggets are not added to the generated models palette and the model information cannot be browsed.

Methods for modeling time series data require a uniform interval between each measurement, with any missing values indicated by empty rows. If your data does not already meet this requirement, you need to transform values as needed.

Other points of interest regarding Time Series nodes:

- Fields must be numeric.
- Date fields cannot be used as inputs.
- Partitions are ignored.

The Streaming Time Series node estimates exponential smoothing, univariate Autoregressive Integrated Moving Average (ARIMA), and multivariate ARIMA (or transfer function) models for time series and produces forecasts based on the time series data. Also available is an Expert Modeler, which attempts to automatically identify and estimate the best-fitting ARIMA or exponential smoothing model for one or more target fields.

**Parent topic:** [Record Operations](#)

## SMOTE node

---

The Synthetic Minority Over-sampling Technique (SMOTE) node provides an over-sampling algorithm to deal with imbalanced data sets. It provides an advanced method for balancing data. The SMOTE node in Watson Studio is implemented in Python and requires the imbalanced-learn® Python library.

For details about the imbalanced-learn library, see <http://contrib.scikit-learn.org/imbalanced-learn/about.html><sup>1</sup>.

The Modeling tab on the nodes palette contains the SMOTE node and other Python nodes.

<sup>1</sup>Lematre, Nogueira, Aridas. "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning." *Journal of Machine Learning Research*, vol. 18, no. 17, 2017, pp. 1-5.  
(<http://jmlr.org/papers/v18/16-365.html>)

**Parent topic:** [Record Operations](#)

## RFM Aggregate node

---

The Recency, Frequency, Monetary (RFM) Aggregate node allows you to take customers' historical transactional data, strip away any unused data, and combine all of their remaining transaction data into a single row (using their unique customer ID as a key) that lists when they last dealt with you (recency), how many transactions they have made (frequency), and the total value of those transactions (monetary).

Before proceeding with any aggregation, you should take time to clean the data, concentrating especially on any missing values.

Once you have identified and transformed the data using the RFM Aggregate node, you may use an RFM Analysis node to carry out further analysis.

Note that once the data file has been run through the RFM Aggregate node, it will not have any target values; therefore, before being able to use it as input for further predictive analysis with any modeling nodes such as C5.0 or CHAID, you will need to merge it with other customer data (for example, by matching the customer IDs).

The RFM Aggregate and RFM Analysis nodes are set up to use independent binning; that is, they rank and bin data on each measure of recency, frequency, and monetary value, without regard to their values or the other two measures.

**Parent topic:** [Record Operations](#)

## Space-Time-Boxes node

---

Space-Time-Boxes (STB) are an extension of Geohashed spatial locations. More specifically, an STB is an alphanumeric string that represents a regularly shaped region of space and time.

For example, the STB dr5ru7|2013-01-01 00:00:00|2013-01-01 00:15:00 is made up of the following three parts:

- The geohash dr5ru7
- The start timestamp 2013-01-01 00:00:00
- The end timestamp 2013-01-01 00:15:00

As an example, you could use space and time information to improve confidence that two entities are the same because they are virtually in the same place at the same time. Alternatively, you could improve the accuracy of relationship identification by showing that two entities are related due to their proximity in space and time.

In the node properties, you can choose the Individual Records or Hangouts mode as appropriate for your requirements. Both modes require the same basic details, as follows:

Latitude field. Select the field that identifies the latitude (in WGS84 coordinate system).

Longitude field. Select the field that identifies the longitude (in WGS84 coordinate system).

Timestamp field. Select the field that identifies the time or date.

## Individual Records

---

Use this mode to add an additional field to a record to identify its location at a given time.

## Hangouts

---

A hangout can be thought of as a location and/or time in which an entity is continually or repeatedly found. For example, you can use a hangout to identify a vehicle that makes regular transportation runs and identify any deviations from the norm.

The hangout detector monitors the movement of entities and flags conditions where an entity is observed to be "hanging out" in the area. The hangout detector automatically assigns each flagged hangout to one or more STBs, and uses in-memory entity and event tracking to detect hangouts with optimum efficiency.

Following is more detail about what qualifies as a hangout:

Let  $e_1, \dots, e_n$  denote all time ordered events that are received from a given entity ID during a time duration  $(t_1, t_n)$ . These events qualify as a hangout if:

- $n \geq \text{minimum number of events}$
- $t_n - t_1 \geq \text{minimum dwell time}$
- All events  $e_1, \dots, e_n$  occur in the same STB

Notes:

- The hangout detector's in-memory event data is not shared across processes. Therefore, a particular entity has affinity with a particular hangout detector node. That is, incoming motion data for an entity must always be consistently passed to the hangout detector node tracking that entity, which is ordinarily the same node throughout the run.
- The hangout detector's in-memory event data is volatile. Whenever the hangout detector is exited and restarted, any work-in-progress hangouts are lost. This means stopping and restarting the process might cause the system to miss reporting real hangouts. A potential remedy involves replaying some of the historical motion data (for example, going back 48 hours and replaying the motion records that are applicable to any node that was restarted).
- The hangout detector must be fed data in time-sequential order.

**Parent topic:** [Record Operations](#)

## Extension Transform node

---

With the Extension Transform node, you can take data from an SPSS Modeler flow and apply transformations to the data using R scripting or Python for Spark scripting.

When the data has been modified, it's returned to the flow for further processing, model building, and model scoring. The Extension Transform node makes it possible to transform data using algorithms that are written in R or Python for Spark, and enables you to develop data transformation methods that are tailored to a particular problem.

After adding the node to your canvas, double-click the node to open its properties.

### Syntax tab

---

Select your type of syntax - R or Python for Spark. Then enter or paste your custom script for transforming data. When your syntax is ready, you can run the node. The following options are available for R syntax:

- Convert flag fields. Specifies how flag fields are treated. There are two options: Strings to factor, Integers and Reals to double, and Logical values (True, False). If you select Logical values (True, False) the original values of the flag fields are lost. For example, if a field has values `Male` and `Female`, these are changed to `True` and `False`.
- Convert missing values to the R 'not available' value (NA). When selected, any missing values are converted to the R NA value. The value NA is used by R to identify missing values. Some R functions that you use might have an argument that can control how the function behaves when the data contains NA. For example, the function might allow you to choose to automatically exclude records that contain NA. If this option isn't selected, any missing values are passed to R unchanged, and might cause errors when your R script runs.
- Convert date/time fields to R classes with special control for time zones. When selected, variables with date or datetime formats are converted to R date/time objects. You must select one of the following options:
  - **R POSIXct.** Variables with date or datetime formats are converted to R `POSIXct` objects.
  - **R POSIXlt (list).** Variables with date or datetime formats are converted to R `POSIXlt` objects.

Note: The POSIX formats are advanced options. Use these options only if your R script specifies that datetime fields are treated in ways that require these formats. The POSIX formats don't apply to variables with time formats.

### Console Output tab

---

The Console Output tab contains any output that's received when the R script or Python for Spark script runs (for example, if using an R script, it shows output received from the R console when the R script in the R Syntax field on the Syntax tab is executed). This output might include R or Python error messages or warnings that are produced when the R script or Python script is executed. The output can be used, primarily, to debug the script. The Console Output tab also contains the script from the R Syntax or Python Syntax field.

Every time the Extension Transform script runs, the content of the Console Output tab is overwritten with the output received from the R console or Python for Spark. You can't edit the output.

**Parent topic:** [Record Operations](#)

## CPLEX Optimization node

---

With the CPLEX Optimization node, you can use complex mathematical (CPLEX) based optimization via an Optimization Programming Language (OPL) model file.

For more information about CPLEX optimization and OPL, see the [IBM ILOG CPLEX Optimization Studio documentation](#).

When outputting the data generated by the CPLEX Optimization node, you can output the original data from the data sources together as single indexes, or as multiple dimensional indexes of the result.

## OPL Model and Input Data

---

### OPL Model

Type or paste the Optimization Programming Language (OPL) model syntax in this field.

### Tuple Set Name in OPL Corresponds to Incoming Data

Enter the tuple set name in the OPL model that corresponds to the incoming data. Then, if needed, verify that all tuple fields are mapped to data input fields according to their order in the tuple definition.

### Input Mapping

Enter the tuple fields and data input fields for mapping. All tuple fields must be mapped to data input fields in the order they're declared in the tuple definition.

## Other Data

---

### OPL Data

Use the Other Data tab if you need to specify any other data for the optimization.

## Output

---

When the output is a decision variable, it must take the prior data sources (incoming data) as indexes, and the indexes must be predefined in the Input Mappings section on the OPL Model and Input Data tab. No other type of decision variables are currently supported. The decision variable can have a single index or multiple indexes. SPSS Modeler will output the CPLEX results with all or part of the original incoming data together, which is consistent with other SPSS Modeler nodes. The referred corresponding indexes must be specified in the Output Tuple field described below.

### Output Mode

Choose the output mode (Raw Output or Decision Variable) and specify other options as appropriate. The Raw Output option will output the objective function value directly, regardless of name.

### Objective Function Value Variable Name in OPL

This field is enabled if you selected the Decision Variable output mode. Enter the name of the objective function value variable from the OPL model.

### Objective Function Value Field Name for Output

Enter the field name to use in the output. The default is `_OBJECTIVE`.

### Output Tuple

Enter the name of the predefined tuple from the incoming data. This acts as the indexes of the decision variable and is expected to be output with the Variable Outputs. The Output Tuple should be consistent with the decision variable definition in the OPL. If there are multiple indexes, the tuple names must be joined by a comma (,).

### Variable Outputs

Add one or more variables to include in the output.

### Note:

- CPLEX isn't supported on macOS.
- When running a flow containing a CPLEX Optimization node, the CPLEX library has a limitation of 1000 variables and 1000 constraints. If desired, you can install the full edition of IBM ILOG CPLEX and complete the following step to avoid this limitation.
  - Edit the options.cfg file and add the OPL library path (by default, the file is installed at C:\Program Files\IBM Watson Studio\resources\application\runtimes\modeler\server\config) as follows:

```
cplex_opl_lib_path="<CPLEX_path>\opl\bin\x64_win64"
```



Where <CPLEX\_path> is your CPLEX installation directory, such as C:\Program Files\IBM\ILOG\CPLEX\_Studio127. This is the only supported setting in options.cfg. Don't edit any other settings in the file.

**Parent topic:** [Record Operations](#)

## Field Operations

---

After an initial data exploration, you will probably need to select, clean, or construct data in preparation for analysis. The Field Operations palette contains many nodes useful for this transformation and preparation.

For example, using a Derive node, you might create an attribute that is not currently represented in the data. Or you might use a Binning node to recode field values automatically for targeted analysis. You will probably find yourself using a Type node frequently; it allows you to assign a measurement level, values, and a modeling role for each field in the dataset. Its operations are useful for handling missing values and downstream modeling.

- **Auto Data Prep node**  
Preparing data for analysis is one of the most important steps in any project - and traditionally, one of the most time consuming. Automated Data Preparation (ADP) handles the task for you, analyzing your data and identifying fixes, screening out fields that are problematic or not likely to be useful, deriving new attributes when appropriate, and improving performance through intelligent screening techniques. You can use the algorithm in fully *automatic* fashion, allowing it to choose and apply fixes, or you can use it in *interactive* fashion, previewing the changes before they are made and accept or reject them as you want.
- **Type node**  
You can specify field properties in a Type node.
- **Filter node**  
You can rename or exclude fields at any point in a flow. For example, as a medical researcher, you may not be concerned about the potassium level (field-level data) of patients (record-level data); therefore, you can filter out the K (potassium) field. This can be done using a separate Filter node or using the Filter tab on an import or output node. The functionality is the same regardless of which node it's accessed from.
- **Derive node**  
One of the most powerful features in Watson Studio is the ability to modify data values and derive new fields from existing data. During lengthy data mining projects, it is common to perform several derivations, such as extracting a customer ID from a string of Web log data or creating a customer lifetime value based on transaction and demographic data. All of these transformations can be performed, using a variety of field operations nodes.
- **Filler node**  
Filler nodes are used to replace field values and change storage. You can choose to replace values based on a specified CLEM condition, such as @BLANK (FIELD) . Alternatively, you can choose to replace all blanks or null values with a specific value. Filler nodes are often used in conjunction with the Type node to replace missing values.
- **Reclassify node**  
The Reclassify node enables the transformation from one set of categorical values to another. Reclassification is useful for collapsing categories or regrouping data for analysis.
- **Binning node**  
The Binning node enables you to automatically create new nominal fields based on the values of one or more existing continuous (numeric range) fields. For example, you can transform a continuous income field into a new categorical field containing income groups of equal width, or as deviations from the mean. Alternatively, you can select a categorical "supervisor" field in order to preserve the strength of the original association between the two fields.
- **RFM Analysis node**  
You can use the Recency, Frequency, Monetary (RFM) Analysis node to determine quantitatively which customers are likely to be the best ones by examining how recently they last purchased from you (recency), how often they purchased (frequency), and how much they spent over all transactions (monetary).
- **Ensemble node**  
The Ensemble node combines two or more model nuggets to obtain more accurate predictions than can be

gained from any of the individual models. By combining predictions from multiple models, limitations in individual models may be avoided, resulting in a higher overall accuracy. Models combined in this manner typically perform at least as well as the best of the individual models and often better.

- **Partition node**  
Partition nodes are used to generate a partition field that splits the data into separate subsets or samples for the training, testing, and validation stages of model building. By using one sample to generate the model and a separate sample to test it, you can get a good indication of how well the model will generalize to larger datasets that are similar to the current data.
- **Set to Flag node**  
Use the Set to Flag node to derive flag fields based on the categorical values defined for one or more nominal fields.
- **Restructure node**  
The Restructure node can be used to generate multiple fields based on the values of a nominal or flag field. The newly generated fields can contain values from another field or numeric flags (0 and 1). The functionality of this node is similar to that of the Set to Flag node. However, it offers more flexibility. It allows you to create fields of any type (including numeric flags), using the values from another field. You can then perform aggregation or other manipulations with other nodes downstream. (The Set to Flag node lets you aggregate fields in one step, which may be convenient if you are creating flag fields.)
- **Transpose node**  
By default, columns are fields and rows are records or observations. If necessary, you can use a Transpose node to swap the data in rows and columns so that fields become records and records become fields.
- **Field Reorder node**  
With the Field Reorder node, you can define the natural order used to display fields downstream. This order affects the display of fields in a variety of places, such as tables, lists, and the Field Chooser.
- **History node**  
History nodes are most often used for sequential data, such as time series data.
- **Time Intervals node**  
Use the Time Intervals node to specify intervals and derive a new time field for estimating or forecasting. A full range of time intervals is supported, from seconds to years.
- **Anonymize node**  
With the Anonymize node, you can disguise field names, field values, or both when working with data that's to be included in a model downstream of the node. In this way, the generated model can be freely distributed (for example, to Technical Support) with no danger that unauthorized users will be able to view confidential data, such as employee records or patients' medical records.
- **Reproject node**  
With geospatial or map data, two of the most common ways to identify coordinates are projected coordinate and geographic coordinate systems. In Watson Studio, items such as the Expression Builder spatial functions and the Spatio-Temporal Prediction (STP) node use the projected coordinate system. Therefore, any data you import that is recorded with a geographic coordinate system must be reprojected.

**Parent topic:** [Nodes palette](#)

## Auto Data Prep node

---

Preparing data for analysis is one of the most important steps in any project - and traditionally, one of the most time consuming. Automated Data Preparation (ADP) handles the task for you, analyzing your data and identifying fixes, screening out fields that are problematic or not likely to be useful, deriving new attributes when appropriate, and improving performance through intelligent screening techniques. You can use the algorithm in fully *automatic* fashion, allowing it to choose and apply fixes, or you can use it in *interactive* fashion, previewing the changes before they are made and accept or reject them as you want.

Using ADP enables you to make your data ready for model building quickly and easily, without needing prior knowledge of the statistical concepts involved. Models will tend to build and score more quickly

Note: When ADP prepares a field for analysis, it creates a new field containing the adjustments or transformations, rather than replacing the existing values and properties of the old field. The old field is not used in further analysis; its

role is set to None.

Example. An insurance company with limited resources to investigate homeowner's insurance claims wants to build a model for flagging suspicious, potentially fraudulent claims. Before building the model, they will ready the data for modeling using automated data preparation. Since they want to be able to review the proposed transformations before the transformations are applied, they will use automated data preparation in interactive mode.

An automotive industry group keeps track of the sales for a variety of personal motor vehicles. In an effort to be able to identify over- and underperforming models, they want to establish a relationship between vehicle sales and vehicle characteristics. They will use automated data preparation to prepare the data for analysis, and build models using the data "before" and "after" preparation to see how the results differ.

What is your objective? Automated data preparation recommends data preparation steps that will affect the speed with which other algorithms can build models and improve the predictive power of those models. This can include transforming, constructing and selecting features. The target can also be transformed. You can specify the model-building priorities that the data preparation process should concentrate on.

- Balance speed and accuracy. This option prepares the data to give equal priority to both the speed with which data are processed by model-building algorithms and the accuracy of the predictions.
- Optimize for speed. This option prepares the data to give priority to the speed with which data are processed by model-building algorithms. When you are working with very large datasets, or are looking for a quick answer, select this option.
- Optimize for accuracy. This option prepares the data to give priority to the accuracy of predictions produced by model-building algorithms.
- Custom analysis. When you want to manually change the algorithm on the Settings tab, select this option. Note that this setting is automatically selected if you subsequently make changes to options on the Settings tab that are incompatible with one of the other objectives.

## Training the node

---

The ADP node is implemented as a process node and works in a similar way to the Type node; **training** the ADP node corresponds to instantiating the Type node. Once analysis has been performed, the specified transformations are applied to the data without further analysis as long as the upstream data model does not change. Like the Type and Filter nodes, if the ADP node is disconnected it remembers the data model and transformations so that if it is reconnected it does not need to be retrained; this enables you to train it on a subset of typical data and then copy or deploy it for use it on live data as often as required.

**Parent topic:** [Field Operations](#)

## Type node

---

You can specify field properties in a Type node.

The following main properties are available.

- Field. Specify value and field labels for data in Watson Studio. For example, field metadata imported from a data asset can be viewed or modified here. Similarly, you can create new labels for fields and their values.
- Measure. This is the measurement level, used to describe characteristics of the data in a given field. If all the details of a field are known, it's called *fully instantiated*.  
Note: The measurement level of a field is different from its storage type, which indicates whether the data is stored as strings, integers, real numbers, dates, times, timestamps, or lists.
- Role. Used to tell modeling nodes whether fields will be Input (predictor fields) or Target (predicted fields) for a machine-learning process. Both and None are also available roles, along with Partition, which indicates a field used to partition records into separate samples for training, testing, and validation. The value Split specifies that separate models will be built for each possible value of the field.
- Value mode. Use this column to specify options for reading data values from the dataset, or use the Specify option to specify measurement levels and values.

- **Values.** With this column, you can specify options for reading data values from the data set, or specify measurement levels and values separately. You can also choose to pass fields without reading their values. You can't amend the cell in this column if the corresponding Field entry contains a list.
- **Check.** With this column, you can set options to ensure that field values conform to the specified values or ranges. You can't amend the cell in this column if the corresponding Field entry contains a list.

Click the Edit (gear) icon next to each row to open additional options.

Tip: Icons in the Type node properties quickly indicate the data type of each field, such as string, date, double integer, or hashtag.

Figure 1. New Type node icons

Field
# customer_
abc campaign
# response
🕒 response_

- **Viewing and setting information about types**  
From the Type node, you can specify field metadata and properties that are invaluable to modeling and other work.
- **Measurement levels**  
The measure, also referred to as measurement level, describes the usage of data fields in SPSS Modeler.
- **Converting continuous data**  
Treating categorical data as continuous can have a serious impact on the quality of a model, especially if it's the target field (for example, producing a regression model rather than a binary model). To prevent this, you can convert integer ranges to categorical types such as `Ordinal` or `Flag`.
- **What is instantiation?**  
*Instantiation* is the process of reading or specifying information, such as storage type and values for a data field. To optimize system resources, instantiating is a user-directed process - you tell the software to read values by running data through a Type node.
- **Data values**  
Using the Value mode column in the Type node settings, you can read values automatically from the data, or you can specify measures and values.
- **Defining missing values**  
In the Type node settings, select the desired field in the table and then click the gear icon at the end of its row. Missing values settings are available at the bottom of the window that appears.
- **Checking type values**  
Turning on the Check option for each field examines all values in that field to determine whether they comply with the current type settings or the values that you've specified. This is useful for cleaning up datasets and reducing the size of a dataset within a single operation.
- **Setting the field role**  
A field's role controls how it's used in model building; for example, whether a field is an input or target (the thing being predicted).
- **Setting field format options**  
With the FORMAT settings in the Type and Table nodes you can specify formatting options for current or unused fields.

**Parent topic:** [Field Operations](#)

## Viewing and setting information about types

---

From the Type node, you can specify field metadata and properties that are invaluable to modeling and other work.

These properties include:

- Specifying a usage type, such as range, set, ordered set, or flag, for each field in your data
- Setting options for handling missing values and system nulls
- Setting the role of a field for modeling purposes
- Specifying values for a field and options used to automatically read values from your data
- Specifying value labels

**Parent topic:** [Type node](#)

## Measurement levels

---

The measure, also referred to as measurement level, describes the usage of data fields in SPSS Modeler.

You can specify the Measure in the node properties of an import node or a Type node. For example, you may want to set the measure for an integer field with values of 1 and 0 to Flag. This usually indicates that 1 = `True` and 0 = `False`.

Storage versus measurement. Note that the measurement level of a field is different from its storage type, which indicates whether data is stored as a string, integer, real number, date, time, or timestamp. While you can modify data types at any point in a flow by using a Type node, storage must be determined at the source when reading data in (although you can subsequently change it using a conversion function).

The following measurement levels are available:

- **Default.** Data whose storage type and values are unknown (for example, because they haven't yet been read) are displayed as Default.
- **Continuous.** Used to describe numeric values, such as a range of 0-00 or 0.75-1.25. A continuous value can be an integer, real number, or date/time.
- **Categorical.** Used for string values when an exact number of distinct values is unknown. This is an *uninstantiated* data type, meaning that all possible information about the storage and usage of the data is not yet known. Once data is read, the measurement level will be Flag, Nominal, or Typeless, depending on the maximum number of members for nominal fields specified.
- **Flag.** Used for data with two distinct values that indicate the presence or absence of a trait, such as `true` and `false`, `Yes` and `No`, or 0 and 1. The values used may vary, but one must always be designated as the "true" value, and the other as the "false" value. Data may be represented as text, integer, real number, date, time, or timestamp.
- **Nominal.** Used to describe data with multiple distinct values, each treated as a member of a set, such as `small/medium/large`. Nominal data can have any storage - numeric, string, or date/time. Note that setting the measurement level to Nominal doesn't automatically change the values to string storage.
- **Ordinal.** Used to describe data with multiple distinct values that have an inherent order. For example, salary categories or satisfaction rankings can be typed as ordinal data. The order is defined by the natural sort order of the data elements. For example, 1, 3, 5 is the default sort order for a set of integers, while `HIGH`, `LOW`, `NORMAL` (ascending alphabetically) is the order for a set of strings. The ordinal measurement level enables you to define a set of categorical data as ordinal data for the purposes of visualization, model building, and export to other applications (such as IBM SPSS Statistics) that recognize ordinal data as a distinct type. You can use an ordinal field anywhere that a nominal field can be used. Additionally, fields of any storage type (real, integer, string, date, time, and so on) can be defined as ordinal.
- **Typeless.** Used for data that doesn't conform to any of the above types, for fields with a single value, or for nominal data where the set has more members than the defined maximum. It's also useful for cases in which the measurement level would otherwise be a set with many members (such as an account number). When you

select Typeless for a field, the role is automatically set to None, with Record ID as the only alternative. The default maximum size for sets is 250 unique values.

- **Collection.** Used to identify non-geospatial data that is recorded in a list. A collection is effectively a list field of zero depth, where the elements in that list have one of the other measurement levels.
- **Geospatial.** Used with the List storage type to identify geospatial data. Lists can be either List of Integer or List of Real fields with a list depth that's between zero and two, inclusive.

You can manually specify measurement levels, or you can allow the software to read the data and determine the measurement level based on the values it reads. Alternatively, where you have several continuous data fields that should be treated as categorical data, you can choose an option to convert them. See [Converting continuous data](#).

## To use auto-typing

---

1. In a Type node, set the Value mode column to Read for the desired fields. This will make metadata available to all nodes downstream.
2. Click Read Values to read values from the data source immediately.

## To manually set the measurement level for a field

---

1. Select a field in the table.
  2. From the drop-down in the Measure column, select a measurement level for the field.
  3. Alternatively, you can use the check boxes to select multiple fields, and then use the top-level drop-down to set the measurement level for all the selected fields at once.
- **Geospatial measurement sublevels**  
The Geospatial measurement level, which is used with the List storage type, has six sublevels that are used to identify different types of geospatial data.

**Parent topic:** [Type node](#)

## Geospatial measurement sublevels

---

The Geospatial measurement level, which is used with the List storage type, has six sublevels that are used to identify different types of geospatial data.

- **Point.** Identifies a specific location (for example, the center of a city).
- **Polygon.** A series of points that identifies the single boundary of a region and its location (for example, a county).
- **LineString.** Also referred to as a Polyline or just a Line, a LineString is a series of points that identifies the route of a line. For example, a LineString might be a fixed item, such as a road, river, or railway; or the track of something that moves, such as an aircraft's flight path or a ship's voyage.
- **MultiPoint.** Used when each row in your data contains multiple points per region. For example, if each row represents a city street, the multiple points for each street can be used to identify every street lamp.
- **MultiPolygon.** Used when each row in your data contains several polygons. For example, if each row represents the outline of a country, the US can be recorded as several polygons to identify the different areas such as the mainland, Alaska, and Hawaii.
- **MultiLineString.** Used when each row in your data contains several lines. Because lines cannot branch, you can use a MultiLineString to identify a group of lines (for example, data such as the navigable waterways or the railway network in each country).

## Restrictions

---

Be aware of these restrictions when using geospatial data:

- The coordinate system can affect the format of the data. For example, a Projected coordinate system uses the coordinate values x, y, and (when required) z, whereas a Geographic coordinate system uses the coordinate

values longitude, latitude, and (when required) a value for either altitude or depth.

- A LineString can't cross over itself
- A Polygon is not self-closing; for each Polygon you must ensure that the first and last points are defined as the same.
- The direction of the data in a MultiPolygon is important; clockwise indicates a solid form and counterclockwise indicates a hole. For example, if you record an area of a country that contains lakes, the main land area border can be recorded in a clockwise direction and the shape of each lake in a counterclockwise direction.
- A Polygon can't intersect with itself. An example of this intersection would be if you tried to plot the boundary of the polygon as a continuous line in the form of a figure 8.
- MultiPolygons can't overlap each other
- For Geospatial fields, the only relevant storage types are Real and Integer (the default setting is Real)

**Parent topic:** [Measurement levels](#)

## Converting continuous data

---

Treating categorical data as continuous can have a serious impact on the quality of a model, especially if it's the target field (for example, producing a regression model rather than a binary model). To prevent this, you can convert integer ranges to categorical types such as `Ordinal` or `Flag`.

1. Double-click a Type node to open its properties. Expand the Type Operations section.
2. Specify a value for Set continuous integer field to ordinal if range less than or equal to.
3. Click Apply to convert the affected ranges.
4. If desired, you can also specify a value for Set categorical fields to None if they exceed this many values to automatically ignore large sets with many members.

## Results of the conversion

---

- Where a `Continuous` field with integer storage is changed to `Ordinal`, the upper value is expanded to include all of the integer values through the upper value. For example, if you set the value to 5, the set of values is 1, 2, 3, 4, 5.
- If the `Continuous` field changes to `Flag`, the lower and upper values become the false and true values of the flag field.

**Parent topic:** [Type node](#)

## What is instantiation?

---

*Instantiation* is the process of reading or specifying information, such as storage type and values for a data field. To optimize system resources, instantiating is a user-directed process - you tell the software to read values by running data through a Type node.

- Data with unknown types is also referred to as *uninstantiated*. Data whose storage type and values are unknown is displayed in the Measure column of the Type node settings as *Typeless*.
- When you have some information about a field's storage, such as string or numeric, the data is called *partially instantiated*. Categorical or Continuous are partially instantiated measurement levels. For example, Categorical specifies that the field is symbolic, but you don't know whether it's nominal, ordinal, or flag.
- When all of the details about a type are known, including the values, a *fully instantiated* measurement level - nominal, ordinal, flag, or continuous - is displayed in this column. Note that the *continuous* type is used for both partially instantiated and fully instantiated data fields. Continuous data can be either integers or real numbers.

When a data flow with a Type node runs, uninstantiated types immediately become partially instantiated, based on the initial data values. Once all of the data has passed through the node, all data becomes fully instantiated unless values were set to Pass. If the flow run is interrupted, the data will remain partially instantiated. Once the Types



settings are instantiated, the values of a field are static at that point in the flow. This means that any upstream changes will not affect the values of a particular field, even if you rerun the flow. To change or update the values based on new data or added manipulations, you need to edit them in the Types settings or set the value for a field to Read or Extend.

## When to instantiate

Instantiating in a Type node is useful when:

- The dataset is large, and the flow filters a subset prior to the Type node
- Data has been filtered in the flow
- Data has been merged or appended in the flow
- New data fields are derived during processing

**Parent topic:** [Type node](#)

## Data values

Using the Value mode column in the Type node settings, you can read values automatically from the data, or you can specify measures and values.

The options available in the Value mode drop-down provide instructions for auto-typing, as shown in the following table.

Table 1. Instructions for auto-typing

Option	Function
Read	Data is read when the node runs.
Extend	Data is read and appended to the current data (if any exists).
Pass	No data is read.
Current	Keep current data values.
Specify	You can click the gear icon on the far right of the row to specify values.

Running a Type node or clicking Read Values auto-types and reads values from your data source based on your selection. You can also specify these values manually by using the Specify option and clicking the gear icon at the far right of a row.

After you make changes for fields in the Type node, you can reset value information using the following buttons:

- Using the Read Values button, you can clear changes to field values made in this node (non-inherited values) and reread values from upstream operations. This option is useful for resetting changes that you may have made for specific fields upstream.
- Using the Clear All Values button, you can reset values for all fields read into the node. This option effectively sets the Value mode column to Read for all fields. This option is useful for resetting values for all fields and rereading values and measurement levels from upstream operations.

## Gray text in the Values column

In either a Type node or an import node, if the data in the Values column is shown in black text, it indicates that the values of that field have been read and are stored within that node. If no black text is present in this field, the values of that field haven't been read and are determined further upstream.

There are occasions when you can see the data as gray text. This occurs when the software can identify or infer the valid values of a field without actually reading and storing the data. This is likely to occur if you use a User Input node (because the data is defined within the node, the range of values for a field is always known, even if the values haven't been stored in the node; the values are shown in gray text until you click Read Values).



Important: If you don't instantiate the data in your flow, and your data values are shown in gray, any checking of type values that you set in the Check column is not applied.

- **Setting options for values**  
The Value mode column under the Type node settings displays a drop-down list of predefined values. Choosing the Specify option on this list and then clicking the gear icon opens a new screen where you can set options for reading, specifying, labeling, and handling values for the selected field.
- **Specifying values and labels for continuous data**  
The Continuous measurement level is for numeric fields.
- **Specifying values and labels for nominal and ordinal data**  
Nominal (set) and ordinal (ordered set) measurement levels indicate that the data values are used discretely as a member of the set. The storage types for a set can be string, integer, real number, or date/time.
- **Specifying values for a flag**  
Use flag fields to display data that has two distinct values. The storage types for flags can be string, integer, real number, or date/time.
- **Specifying values for collection data**  
Collection fields display non-geospatial data that's in a list.
- **Specifying values for geospatial data**  
Geospatial fields display geospatial data that's in a list. For the Geospatial measurement level, you can use various options to set the measurement level of the elements within the list.

**Parent topic:** [Type node](#)

## Setting options for values

---

The Value mode column under the Type node settings displays a drop-down list of predefined values. Choosing the Specify option on this list and then clicking the gear icon opens a new screen where you can set options for reading, specifying, labeling, and handling values for the selected field.

Many of the controls are common to all types of data. These common controls are discussed here.

**Measure.** Displays the currently selected measurement level. You can change this setting to reflect the way that you intend to use data. For instance, if a field called `day_of_week` contains numbers that represent individual days, you might want to change this to nominal data in order to create a distribution node that examines each category individually.

**Role.** Used to tell modeling nodes whether fields will be Input (predictor fields) or Target (predicted fields) for a machine-learning process. Other roles are also available such as Both , None, Partition, Split, Frequency, or Record ID.

**Value mode.** Select a mode to determine values for the selected field. Choices for reading values include the following:

- **Read.** Select to read values when the node runs.
- **Pass.** Select not to read data for the current field.
- **Specify.** Options here are used to specify values and labels for the selected field. Used with value checking, use this option to specify values that are based on your knowledge of the current field. This option activates unique controls for each type of field. You can't specify values or labels for a field whose measurement level is Typeless.
- **Extend.** Select to append the current data with the values that you enter here. For example, if `field_1` has a range from  $(0, 10)$  and you enter a range of values from  $(8, 16)$ , the range is extended by adding the 16 without removing the original minimum. The new range would be  $(0, 16)$ .
- **Current.** Select to keep the current data values.

**Value Labels (Add/Edit Labels).** In this section you can enter custom labels for each value of the selected field.

Max list length. Only available for data with a measurement level of either Geospatial or Collection. Set the maximum length of the list by specifying the number of elements the list can contain.

Max string length. Only available for typeless data. Use this field when you're generating SQL to create a table. Enter the value of the largest string in your data; this generates a column in the table that's big enough for the string. If the string length value is not available, a default string size is used that may not be appropriate for the data (for example, if the value is too small, errors can occur when writing data to the table; too large a value could adversely affect performance).

Check. Select a method of coercing values to conform to the specified continuous, flag, or nominal values. This option corresponds to the Check column in the main Type node settings, and a selection made here will override those in the main settings. Used with the options for specifying values and labels, value checking allows you to conform values in the data with expected values. For example, if you specify values as 1, 0 and then use the Discard. option here, you can discard all records with values other than 1 or 0.

Define missing values. Select to activate the following controls you can use to declare missing values or blanks in your data.

- Missing values. Use this field to define specific values (such as 99 or 0) as blanks. The value should be appropriate for the storage type of the field.
- Range. Used to specify a range of missing values (such as ages 1–17 or greater than 65). If a bound value is left blank, then the range is unbounded. For example, if you specify a lower bound of 100 with no upper bound, then all values greater than or equal to 100 are defined as missing. The bound values are inclusive. For example, a range with a lower bound of 5 and an upper bound of 10 includes 5 and 10 in the range definition. You can define a missing value range for any storage type, including date/time and string (in which case the alphabetic sort order is used to determine whether a value is within the range).
- Null/White space. You can also specify system nulls (displayed in the data as `$null$`) and white space (string values with no visible characters) as blanks. Note that the Type node also treats empty strings as white space for purposes of analysis, although they are stored differently internally and may be handled differently in certain cases.

Note: To code blanks as undefined or `$null$`, use the Filler node.

**Parent topic:** [Data values](#)

## Specifying values and labels for continuous data

---

The Continuous measurement level is for numeric fields.

There are three storage types for continuous data:

- Real
- Integer
- Date/Time

The same settings are used to edit all continuous fields. The storage type is displayed for reference only. Select the desired field in the Type node settings and then click the gear icon at the end of its row.

### Specifying values

---

The following controls are unique to continuous fields. Use these controls to specify a range of values:

Lower. Specify a lower limit for the value range.

Upper. Specify an upper limit for the value range.

### Specifying labels

---

You can specify labels for any value of a range field.

**Parent topic:** [Data values](#)

## Specifying values and labels for nominal and ordinal data

---

Nominal (set) and ordinal (ordered set) measurement levels indicate that the data values are used discretely as a member of the set. The storage types for a set can be string, integer, real number, or date/time.

The following controls are unique to nominal and ordinal fields. You can use them to specify values and labels. Select the desired field in the Type node settings and then click the gear icon at the end of its row.

Values and Labels. You can specify values based on your knowledge of the current field. You can enter expected values for the field and check the dataset's conformity to these values using the Check options. And you can specify labels for each value in the set. These labels appear in a variety of locations, such as graphs, tables, output, and model browsers.

**Parent topic:** [Data values](#)

## Specifying values for a flag

---

Use flag fields to display data that has two distinct values. The storage types for flags can be string, integer, real number, or date/time.

True. Specify a flag value for the field when the condition is met.

False. Specify a flag value for the field when the condition is not met.

Labels. Specify labels for each value in the flag field. These labels appear in a variety of locations, such as graphs, tables, output, and model browsers.

**Parent topic:** [Data values](#)

## Specifying values for collection data

---

Collection fields display non-geospatial data that's in a list.

The only item you can set for the Collection measurement level is the List measure. By default, this measure is set to Typeless, but you can select another value to set the measurement level of the elements within the list. You can choose one of the following options:

- Typeless
- Continuous
- Nominal
- Ordinal
- Flag

**Parent topic:** [Data values](#)

## Specifying values for geospatial data

---

Geospatial fields display geospatial data that's in a list. For the Geospatial measurement level, you can use various options to set the measurement level of the elements within the list.

Type. Select the measurement sublevel of the geospatial field. The available sublevels are determined by the depth of the list field. The defaults are: Point (zero depth), LineString (depth of one), and Polygon (depth of one).

For more information about sublevels, see [Geospatial measurement sublevels](#).

Coordinate system. This option is only available if you changed the measurement level to Geospatial from a non-geospatial level. To apply a coordinate system to your geospatial data, select this option. To use a different coordinate system, click Change.

**Parent topic:** [Data values](#)

## Defining missing values

---

In the Type node settings, select the desired field in the table and then click the gear icon at the end of its row. Missing values settings are available at the bottom of the window that appears.

Select Define missing values to define missing value handling for this field. Here you can define explicit values to be considered as missing values for this field, or this can also be accomplished by means of a downstream Filler node.

**Parent topic:** [Type node](#)

## Checking type values

---

Turning on the Check option for each field examines all values in that field to determine whether they comply with the current type settings or the values that you've specified. This is useful for cleaning up datasets and reducing the size of a dataset within a single operation.

The Check column in the Type node determines what happens when a value outside of the type limits is discovered. To change the check settings for a field, use the drop-down list for that field in the Check column. To set the check settings for all fields, select the check box beside the top-level Field column heading. Then use the top-level drop-down above the Check column.

The following check options are available:

None. Values will be passed through without checking. This is the default setting.

Nullify. Change values outside of the limits to the system null (`$null$`).

Coerce. Fields whose measurement levels are fully instantiated will be checked for values that fall outside the specified ranges. Unspecified values will be converted to a legal value for that measurement level using the following rules:

- For flags, any value other than the true and false value is converted to the false value
- For sets (nominal or ordinal), any unknown value is converted to the first member of the set's values
- Numbers greater than the upper limit of a range are replaced by the upper limit
- Numbers less than the lower limit of a range are replaced by the lower limit
- Null values in a range are given the midpoint value for that range

Discard. When illegal values are found, the entire record is discarded.

Warn. The number of illegal items is counted and reported in the flow properties dialog when all of the data has been read.

Abort. The first illegal value encountered terminates the running of the flow. The error is reported in the flow properties dialog.

**Parent topic:** [Type node](#)

## Setting the field role

---

A field's role controls how it's used in model building; for example, whether a field is an input or target (the thing being predicted).

Note: The Partition, Frequency, and Record ID roles can each be applied to a single field only.

The following roles are available:

**Input.** The field is used as an input to machine learning (a predictor field).

**Target.** The field is used as an output or target for machine learning (one of the fields that the model will try to predict).

**Both.** The field is used as both an input and an output by the Apriori node. All other modeling nodes will ignore the field.

**None.** The field is ignored by machine learning. Fields whose measurement level is set to Typeless are automatically set to None in the Role column.

**Partition.** Indicates a field used to partition the data into separate samples for training, testing, and (optional) validation purposes. The field must be an instantiated set type with two or three possible values (as defined in the advanced settings by clicking the gear icon). The first value represents the training sample, the second represents the testing sample, and the third (if present) represents the validation sample. Any additional values are ignored, and flag fields can't be used. Note that to use the partition in an analysis, partitioning must be enabled in the node settings of the appropriate model-building or analysis node. Records with null values for the partition field are excluded from the analysis when partitioning is enabled. If you defined multiple partition fields in the flow, you must specify a single partition field in the node settings for each applicable modeling node. If a suitable field doesn't already exist in your data, you can create one using a Partition node or Derive node. See [Partition node](#) for more information.

**Split.** (Nominal, ordinal, and flag fields only.) Specifies that a model is built for each possible value of the field.

**Frequency.** (Numeric fields only.) Setting this role enables the field value to be used as a frequency weighting factor for the record. This feature is supported by C&R Tree, CHAID, QUEST, and Linear nodes only; all other nodes ignore this role. Frequency weighting is enabled by means of the Use frequency weight option in the node settings of those modeling nodes that support the feature.

**Record ID.** The field is used as the unique record identifier. This feature is ignored by most nodes; however, it's supported by Linear models.

**Parent topic:** [Type node](#)

## Setting field format options

---

With the FORMAT settings in the Type and Table nodes you can specify formatting options for current or unused fields.

Under each formatting type, click Add Columns and add one or more fields. The field name and format setting will be displayed for each field you select. Then click the gear icon to specify formatting options.

The following formatting options are available on a per-field basis:

**Date format.** Select a date format to use for date storage fields or when strings are interpreted as dates by CLEM date functions.

**Time format.** Select a time format to use for time storage fields or when strings are interpreted as times by CLEM time functions.

Number format. You can choose from standard (####.###), scientific (#.###E+##), or currency display formats (\$###.##).

Decimal symbol. Select either a comma (,) or period (.) as the decimal separator.

Number grouping symbol. For number display formats, select the symbol used to group values (for example, the comma in 3,000.00). Options include none, period, comma, space, and locale-defined (in which case the default for the current locale is used).

Decimal places (standard, scientific, currency, export). For number display formats, specify the number of decimal places to use when displaying real numbers. This option is specified separately for each display format. Note that the Export decimal places setting only applies to flat file exports. The number of decimal places exported by the XML Export node is always 6.

Justify. Specifies how the values should be justified within the column. The default setting is Auto, which left-justifies symbolic values and right-justifies numeric values. You can override the default by selecting left, right, or center.

Column width. By default, column widths are automatically calculated based on the values of the field. You can specify a custom width, if needed.

**Parent topic:** [Type node](#)

## Filter node

---

You can rename or exclude fields at any point in a flow. For example, as a medical researcher, you may not be concerned about the potassium level (field-level data) of patients (record-level data); therefore, you can filter out the K (potassium) field. This can be done using a separate Filter node or using the Filter tab on an import or output node. The functionality is the same regardless of which node it's accessed from.

- From import nodes, you can rename or filter fields as the data is read in.
- Using a Filter node, you can rename or filter fields at any point in the flow.
- You can use the Filter tab in any of the above nodes to define or edit multiple response sets.
- Finally, you can use a Filter node to map fields from one import node to another.

**Parent topic:** [Field Operations](#)

## Derive node

---

One of the most powerful features in Watson Studio is the ability to modify data values and derive new fields from existing data. During lengthy data mining projects, it is common to perform several derivations, such as extracting a customer ID from a string of Web log data or creating a customer lifetime value based on transaction and demographic data. All of these transformations can be performed, using a variety of field operations nodes.

Several nodes provide the ability to derive new fields:

- The Derive node modifies data values or creates new fields from one or more existing fields. It creates fields of type formula, flag, nominal, state, count, and conditional.
- The Reclassify node transforms one set of categorical values to another. Reclassification is useful for collapsing categories or regrouping data for analysis.
- The Binning node automatically creates new nominal (set) fields based on the values of one or more existing continuous (numeric range) fields. For example, you can transform a continuous income field into a new categorical field containing groups of income as deviations from the mean. Once you have created bins for the new field, you can generate a Derive node based on the cut points.
- The Set to Flag node derives multiple flag fields based on the categorical values defined for one or more nominal fields.
- The Restructure node converts a nominal or flag field into a group of fields that can be populated with the values of yet another field. For example, given a field named `payment type`, with values of `credit`, `cash`, and

debit, three new fields would be created (credit, cash, debit), each of which might contain the value of the actual payment made.

## Using the Derive node

---

Using the Derive node, you can create six types of new fields from one or more existing fields:

- Formula. The new field is the result of an arbitrary CLEM expression.
- Flag. The new field is a flag, representing a specified condition.
- Nominal. The new field is nominal, meaning that its members are a group of specified values.
- State. The new field is one of two states. Switching between these states is triggered by a specified condition.
- Count. The new field is based on the number of times that a condition has been true.
- Conditional. The new field is the value of one of two expressions, depending on the value of a condition.

Each of these nodes contains a set of special options. These options are discussed in subsequent topics.

Note that use of the following may change row order:

- Executing in a database via SQL pushback
- Executing via remote Analytic Server
- Using functions that run in embedded Analytic Server
- Deriving a list
- Calling spatial functions

**Parent topic:** [Field Operations](#)

## Filler node

---

Filler nodes are used to replace field values and change storage. You can choose to replace values based on a specified CLEM condition, such as `@BLANK (FIELD)`. Alternatively, you can choose to replace all blanks or null values with a specific value. Filler nodes are often used in conjunction with the Type node to replace missing values.

Fill in fields. Select fields from the dataset whose values will be examined and replaced. The default behavior is to replace values depending on the Condition and Replace with expressions specified below. You can also select an alternative method of replacement using the Replace options below.

Note: When selecting multiple fields to replace with a user-defined value, it is important that the field types are similar (all numeric or all symbolic).

Replace. Select to replace the values of the selected field(s) using one of the following methods:

- Based on condition. This option activates the Condition field and Expression Builder for you to create an expression used as a condition for replacement with the value specified.
- Always. Replaces all values of the selected field. For example, you could use this option to convert the storage of income to a string using the following CLEM expression: `(to_string(income))`.
- Blank values. Replaces all user-specified blank values in the selected field. The standard condition `@BLANK (@FIELD)` is used to select blanks. *Note:* You can define blanks using the Types tab of the source node or with a Type node.
- Null values. Replaces all system null values in the selected field. The standard condition `@NULL (@FIELD)` is used to select nulls.
- Blank and null values. Replaces both blank values and system nulls in the selected field. This option is useful when you are unsure whether or not nulls have been defined as missing values.

Condition. This option is available when you have selected the Based on condition option. Use this text box to specify a CLEM expression for evaluating the selected fields. Click the calculator button to open the Expression Builder.

Replace with. Specify a CLEM expression to give a new value to the selected fields. You can also replace the value with a null value by typing `undef` in the text box. Click the calculator button to open the Expression Builder.

Note: When the field(s) selected are string, you should replace them with a string value. Using the default 0 or another numeric value as the replacement value for string fields will result in an error.

Note that use of the following may change row order:

- Executing in a database via SQL pushback
- Executing via remote IBM SPSS Analytic Server
- Using functions that run in embedded Analytic Server
- Deriving a list
- Calling any of the CLEM spatial functions

**Parent topic:** [Field Operations](#)

## Reclassify node

---

The Reclassify node enables the transformation from one set of categorical values to another. Reclassification is useful for collapsing categories or regrouping data for analysis.

For example, you could reclassify the values for `Product` into three groups, such as `Kitchenware`, `Bath` and `Linens`, and `Appliances`. Often, this operation is performed directly from a Distribution node by grouping values and generating a Reclassify node.

Reclassification can be performed for one or more symbolic fields. You can also choose to substitute the new values for the existing field or generate a new field.

### When to use a Reclassify node

---

Before using a Reclassify node, consider whether another Field Operations node is more appropriate for the task at hand:

- To transform numeric ranges into sets using an automatic method, such as ranks or percentiles, you should use a Binning node. See [Binning node](#) for more information.
- To classify numeric ranges into sets manually, you should use a Derive node. For example, if you want to collapse salary values into specific salary range categories, you should use a Derive node to define each category manually. See [Derive node](#) for more information.
- To create one or more flag fields based on the values of a categorical field, such as `Mortgage_type`, you should use a Set to Flag node.
- To convert a categorical field to numeric storage, you can use a Derive node. For example, you could convert `No` and `Yes` values to 0 and 1, respectively.

**Parent topic:** [Field Operations](#)

## Binning node

---

The Binning node enables you to automatically create new nominal fields based on the values of one or more existing continuous (numeric range) fields. For example, you can transform a continuous income field into a new categorical field containing income groups of equal width, or as deviations from the mean. Alternatively, you can select a categorical "supervisor" field in order to preserve the strength of the original association between the two fields.

Binning can be useful for a number of reasons, including:

- Algorithm requirements. Certain algorithms, such as Naive Bayes and Logistic Regression, require categorical inputs.
- Performance. Algorithms such as multinomial logistic may perform better if the number of distinct values of input fields is reduced. For example, use the median or mean value for each bin rather than using the original values.



- **Data Privacy.** Sensitive personal information, such as salaries, may be reported in ranges rather than actual salary figures in order to protect privacy.

A number of binning methods are available. Once you have created bins for the new field, you can generate a Derive node based on the cut points.

## When to use a Binning node

---

Before using a Binning node, consider whether another technique is more appropriate for the task at hand:

- To manually specify cut points for categories, such as specific predefined salary ranges, use a Derive node. See [Derive node](#) for more information.
- To create new categories for existing sets, use a Reclassify node. See [Reclassify node](#) for more information.

## Missing value handling

---

The Binning node handles missing values in the following ways:

- **User-specified blanks.** Missing values specified as blanks are included during the transformation. For example, if you designated -99 to indicate a blank value using the Type node, this value will be included in the binning process. To ignore blanks during binning, you should use a Filler node to replace the blank values with the system null value.
- **System-missing values (\$null\$).** Null values are ignored during the binning transformation and remain nulls after the transformation.

The Settings tab provides options for available techniques. The View tab displays cut points established for data previously run through the node.

**Parent topic:** [Field Operations](#)

## RFM Analysis node

---

You can use the Recency, Frequency, Monetary (RFM) Analysis node to determine quantitatively which customers are likely to be the best ones by examining how recently they last purchased from you (recency), how often they purchased (frequency), and how much they spent over all transactions (monetary).

The reasoning behind RFM analysis is that customers who purchase a product or service once are more likely to purchase again. The categorized customer data is separated into a number of bins, with the binning criteria adjusted as you require. In each of the bins, customers are assigned a score; these scores are then combined to provide an overall RFM score. This score is a representation of the customer's membership in the bins created for each of the RFM parameters. This binned data may be sufficient for your needs, for example, by identifying the most frequent, high-value customers; alternatively, it can be passed on in a flow for further modeling and analysis.

Note, however, that although the ability to analyze and rank RFM scores is a useful tool, you must be aware of certain factors when using it. There may be a temptation to target customers with the highest rankings; however, oversolicitation of these customers could lead to resentment and an actual fall in repeat business. It is also worth remembering that customers with low scores should not be neglected but instead may be cultivated to become better customers. Conversely, high scores alone do not necessarily reflect a good sales prospect, depending on the market. For example, a customer in bin 5 for recency, meaning that they have purchased very recently, may not actually be the best target customer for someone selling expensive, longer-life products such as cars or televisions.

Note: Depending on how your data is stored, you may need to precede the RFM Analysis node with an RFM Aggregate node to transform the data into a usable format. For example, input data must be in customer format, with one row per customer; if the customers' data is in transactional form, use an RFM Aggregate node upstream to derive the recency, frequency, and monetary fields.

The RFM Aggregate and RFM Analysis nodes are set up to use independent binning; that is, they rank and bin data on each measure of recency, frequency, and monetary value, without regard to their values or the other two measures.

**Parent topic:** [Field Operations](#)

## Ensemble node

---

The Ensemble node combines two or more model nuggets to obtain more accurate predictions than can be gained from any of the individual models. By combining predictions from multiple models, limitations in individual models may be avoided, resulting in a higher overall accuracy. Models combined in this manner typically perform at least as well as the best of the individual models and often better.

This combining of nodes happens automatically in the Auto Classifier and Auto Numeric automated modeling nodes.

After using an Ensemble node, you can use an Analysis node or Evaluation node to compare the accuracy of the combined results with each of the input models. To do this, make sure the Filter out fields generated by ensembled models option is not selected in the Ensemble node settings.

## Output fields

---

Each Ensemble node generates a field containing the combined scores. The name is based on the specified target field and prefixed with `$XF_`, `$XS_`, or `$XR_`, depending on the field measurement level: flag, nominal (set), or continuous (range), respectively. For example, if the target is a flag field named `response`, the output field would be `$XF_response`.

Confidence or propensity fields. For flag and nominal fields, additional confidence or propensity fields are created based on the ensemble method, as detailed in the following table.

Table 1. Ensemble method field creation

Ensemble method	Field name
Voting	\$XFC_<field>
Confidence-weighted voting	
Raw-propensity-weighted voting	
Adjusted-propensity-weighted voting	
Highest confidence wins	
Average raw propensity	\$XFRP_<field>
Average adjusted raw propensity	\$XFAP_<field>

**Parent topic:** [Field Operations](#)

## Partition node

---

Partition nodes are used to generate a partition field that splits the data into separate subsets or samples for the training, testing, and validation stages of model building. By using one sample to generate the model and a separate sample to test it, you can get a good indication of how well the model will generalize to larger datasets that are similar to the current data.

The Partition node generates a nominal field with the role set to Partition. Alternatively, if an appropriate field already exists in your data, it can be designated as a partition using a Type node. In this case, no separate Partition node is required. Any instantiated nominal field with two or three values can be used as a partition, but flag fields cannot be used.

Multiple partition fields can be defined in a flow, but if so, a single partition field must be selected in each modeling node that uses partitioning. (If only one partition is present, it is automatically used whenever partitioning is enabled.)

To create a partition field based on some other criterion such as a date range or location, you can also use a Derive node. See [Derive node](#) for more information.

Example. When building an RFM flow to identify recent customers who have positively responded to previous marketing campaigns, the marketing department of a sales company uses a Partition node to split the data into training and test partitions.

**Parent topic:** [Field Operations](#)

## Set to Flag node

Use the Set to Flag node to derive flag fields based on the categorical values defined for one or more nominal fields.

For example, your dataset might contain a nominal field, BP (blood pressure), with the values *High*, *Normal*, and *Low*. For easier data manipulation, you might create a flag field for high blood pressure, which indicates whether or not the patient has high blood pressure.

**Parent topic:** [Field Operations](#)

## Restructure node

The Restructure node can be used to generate multiple fields based on the values of a nominal or flag field. The newly generated fields can contain values from another field or numeric flags (0 and 1). The functionality of this node is similar to that of the Set to Flag node. However, it offers more flexibility. It allows you to create fields of any type (including numeric flags), using the values from another field. You can then perform aggregation or other manipulations with other nodes downstream. (The Set to Flag node lets you aggregate fields in one step, which may be convenient if you are creating flag fields.)

For example, the following dataset contains a nominal field, *Account*, with the values *Savings* and *Draft*. The opening balance and current balance are recorded for each account, and some customers have multiple accounts of each type. Let's say you want to know whether each customer has a particular account type, and if so, how much money is in each account type. You use the Restructure node to generate a field for each of the *Account* values, and you select *Current\_Balance* as the value. Each new field is populated with the current balance for the given record.

Table 1. Sample data before restructuring

CustID	Account	Open_Bal	Current_Bal
12701	Draft	1000	1005.32
12702	Savings	100	144.51
12703	Savings	300	321.20
12703	Savings	150	204.51
12703	Draft	1200	586.32

Table 2. Sample data after restructuring

CustID	Account	Open_Bal	Current_Bal	Account_Draft_Current_Bal	Account_Savings_Current_Bal
12701	Draft	1000	1005.32	1005.32	\$null\$
12702	Savings	100	144.51	\$null\$	144.51
12703	Savings	300	321.20	\$null\$	321.20
12703	Savings	150	204.51	\$null\$	204.51
12703	Draft	1200	586.32	586.32	\$null\$

## Using the Restructure node with the Aggregate node

In many cases, you may want to pair the Restructure node with an Aggregate node. In the previous example, one customer (with the ID 12703) has three accounts. You can use an Aggregate node to calculate the total balance for each account type. The key field is `CustID`, and the aggregate fields are the new restructured fields, `Account_Draft_Current_Bal` and `Account_Savings_Current_Bal`. The following table shows the results.

Table 3. Sample data after restructuring and aggregation

CustID	Record_Count	Account_Draft_Current_Bal_Sum	Account_Savings_Current_Bal_Sum
12701	1	1005.32	\$null\$
12702	1	\$null\$	144.51
12703	3	586.32	525.71

**Parent topic:** [Field Operations](#)

## Transpose node

---

By default, columns are fields and rows are records or observations. If necessary, you can use a Transpose node to swap the data in rows and columns so that fields become records and records become fields.

For example, if you have time series data where each series is a row rather than a column, you can transpose the data prior to analysis.

**Parent topic:** [Field Operations](#)

## Field Reorder node

---

With the Field Reorder node, you can define the natural order used to display fields downstream. This order affects the display of fields in a variety of places, such as tables, lists, and the Field Chooser.

This operation is useful, for example, when working with wide datasets to make fields of interest more visible.

**Parent topic:** [Field Operations](#)

## History node

---

History nodes are most often used for sequential data, such as time series data.

They are used to create new fields containing data from fields in previous records. When using a History node, you may want to use data that is presorted by a particular field. You can use a Sort node to do this.

**Parent topic:** [Field Operations](#)

## Time Intervals node

---

Use the Time Intervals node to specify intervals and derive a new time field for estimating or forecasting. A full range of time intervals is supported, from seconds to years.

Use the node to derive a new time field. The new field has the same storage type as the input time field you chose. The node generates the following items:

- The field specified in the node properties as the Time Field, along with the chosen prefix/suffix. By default the prefix is `$TI_`.
- The fields specified in the node properties as the Dimension fields.
- The fields specified in the node properties as the Fields to aggregate.

You can also generate a number of extra fields, depending on the selected interval or period (such as the minute or second within which a measurement falls).

**Parent topic:** [Field Operations](#)

## Anonymize node

---

With the Anonymize node, you can disguise field names, field values, or both when working with data that's to be included in a model downstream of the node. In this way, the generated model can be freely distributed (for example, to Technical Support) with no danger that unauthorized users will be able to view confidential data, such as employee records or patients' medical records.

Depending on where you place the Anonymize node in your flow, you may need to make changes to other nodes. For example, if you insert an Anonymize node upstream from a Select node, the selection criteria in the Select node will need to be changed if they are acting on values that have now become anonymized.

The method to be used for anonymizing depends on various factors. For field names and all field values except Continuous measurement levels, the data is replaced by a string of the form:

*prefix\_Sn*

where *prefix\_* is either a user-specified string or the default string *anon\_*, and *n* is an integer value that starts at 0 and is incremented for each unique value (for example, *anon\_S0*, *anon\_S1*, etc.).

Field values of type Continuous must be transformed because numeric ranges deal with integer or real values rather than strings. As such, they can be anonymized only by transforming the range into a different range, thus disguising the original data. Transformation of a value *x* in the range is performed in the following way:

$$A * (x + B)$$

where:

A is a scale factor, which must be greater than 0.

B is a translation offset to be added to the values.

## Example

---

In the case of a field *AGE* where the scale factor A is set to 7 and the translation offset B is set to 3, the values for *AGE* are transformed into:

$$7 * (AGE + 3)$$

**Parent topic:** [Field Operations](#)

## Reproject node

---

With geospatial or map data, two of the most common ways to identify coordinates are projected coordinate and geographic coordinate systems. In Watson Studio, items such as the Expression Builder spatial functions and the Spatio-Temporal Prediction (STP) node use the projected coordinate system. Therefore, any data you import that is recorded with a geographic coordinate system must be reprojected.

Where possible, geospatial fields (any fields with a geospatial measurement level) are automatically reprojected when used (not when they are imported). Where any fields cannot be reprojected automatically, you use the Reproject node to change their coordinate system. Reprojecting in this way means that you can correct situations where an error occurs due to use of an incorrect coordinate system.

Here are some example situations where you might have to reproject to change the coordinate system:

- **Append.** If you try to append two data sets with different coordinate systems for a geospatial field, Watson Studio may display an error message such as: `Coordinate systems of <Field1> and <Field2> are not compatible. Reproject one or both fields to the same coordinate system.`

`<Field1>` and `<Field2>` are the names of the geospatial fields that caused the error.

- **If/else expression.** If you use an expression that contains an if/else statement with geospatial fields or return types in both parts of the expression, but with different coordinate systems, Watson Studio may display an error message such as: `The conditional expression contains incompatible coordinate systems: <arg1> and <arg2>.`

`<arg1>` and `<arg2>` are the then or else arguments that return a geospatial type with different coordinate systems.

- **Constructing a list of geospatial fields.** To create a list field that consists of numerous geospatial fields, all geospatial field arguments that are supplied to the list expression must be in the same coordinate system. If they aren't, then an error message such as the following may be displayed: `Coordinate systems of <Field1> and <Field2> are not compatible. Reproject one or both fields to the same coordinate system.`

**Parent topic:** [Field Operations](#)

## Graphs

---

Several phases of the data mining process use graphs and charts to explore data brought in to Watson Studio.

For example, you can connect a Plot or Distribution node to a data source to gain insight into data types and distributions. You can then perform record and field manipulations to prepare the data for downstream modeling operations. Another common use of graphs is to check the distribution and relationships between newly derived fields.

- **Charts node**  
With the Charts node, you can launch the chart builder and create chart definitions to save with your flow. Then when you run the node, chart output is generated.
- **Plot node**  
Plot nodes show the relationship between numeric fields. You can create a plot using points (also known as a scatterplot), or you can use lines. You can create three types of line plots by specifying an X Mode in the node properties.
- **Multiplot node**  
A multiplot is a special type of plot that displays multiple  $y$  fields over a single  $x$  field. The  $y$  fields are plotted as colored lines and each is equivalent to a Plot node with Style set to Line and X Mode set to Sort. Multiplots are useful when you have time sequence data and want to explore the fluctuation of several variables over time.
- **Time Plot node**  
Time Plot nodes allow you to view one or more time series plotted over time. The series you plot must contain numeric values and are assumed to occur over a range of time in which the periods are uniform.
- **Distribution node**  
A distribution graph or table shows the occurrence of symbolic (non-numeric) values, such as mortgage type or gender, in a dataset. A typical use of the Distribution node is to show imbalances in the data that can be rectified by using a Balance node before creating a model. You can automatically generate a Balance node using the Generate menu in the distribution graph or table window.
- **Histogram node**  
Histogram nodes show the occurrence of values for numeric fields. They are often used to explore the data

before manipulations and model building. Similar to the Distribution node, Histogram nodes are frequently used to reveal imbalances in the data.

- **Collection node**

Collections are similar to histograms, but collections show the distribution of values for one numeric field relative to the values of another, rather than the occurrence of values for a single field. A collection is useful for illustrating a variable or field whose values change over time.

- **Web node**

Web nodes show the strength of relationships between values of two or more symbolic fields. The graph displays connections using varying types of lines to indicate connection strength. You can use a Web node, for example, to explore the relationship between the purchase of various items at an e-commerce site or a traditional retail outlet.

- **Evaluation node**

The Evaluation node offers an easy way to evaluate and compare predictive models to choose the best model for your application. Evaluation charts show how models perform in predicting particular outcomes. They work by sorting records based on the predicted value and confidence of the prediction, splitting the records into groups of equal size (*quantiles*), and then plotting the value of the business criterion for each quantile, from highest to lowest. Multiple models are shown as separate lines in the plot.

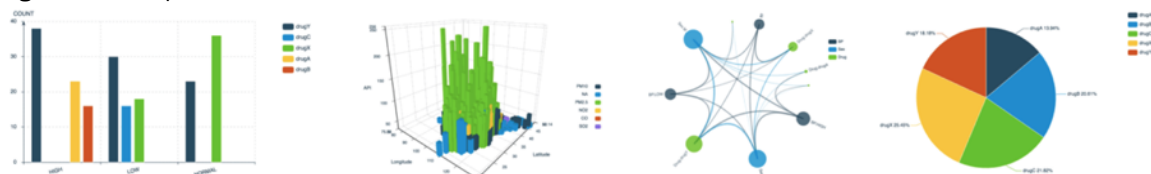
**Parent topic:** [Nodes palette](#)

## Charts node

With the Charts node, you can launch the chart builder and create chart definitions to save with your flow. Then when you run the node, chart output is generated.

The Charts node is available under the Graphs section on the node palette. After adding a Charts node to your flow, double-click it to open the properties pane. Then click Launch Chart Builder to open the chart builder and create one or more chart definitions to associate with the node. See [Visualizing your data](#) for details about creating charts.

Figure 1. Example charts



Notes:

- When you create a chart, it uses a sample of your data. After clicking Save and close to save the chart definition and return to your flow, the Charts node will then use all of your data when you run it.
- Chart definitions are listed in the node properties panel, with icons available for editing them or removing them.
- When you right-click a Charts node to run it, the defined chart (or charts) is built and added to the Outputs pane. Open the chart output to interact with it by hovering over it, zooming in or out, or downloading the chart as an image file (.png).
- When creating a chart, you can click Back to flow to close the chart builder and return to your flow. But you can't run the Charts node until you save a chart definition.

**Parent topic:** [Graphs](#)

## Plot node

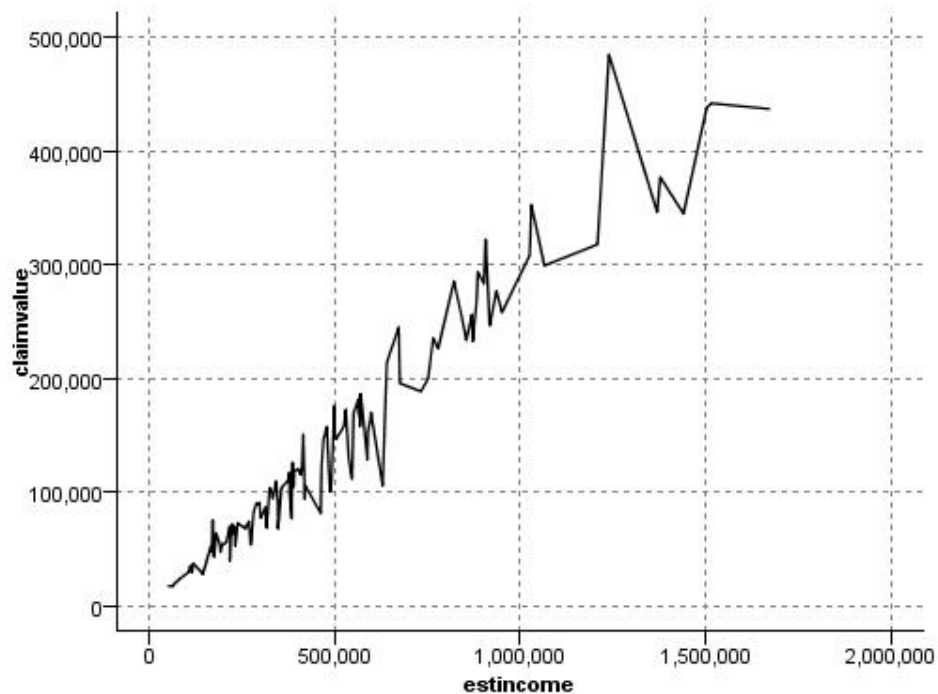
Plot nodes show the relationship between numeric fields. You can create a plot using points (also known as a scatterplot), or you can use lines. You can create three types of line plots by specifying an X Mode in the node properties.

## X Mode = Sort

---

Setting X Mode to Sort causes data to be sorted by values for the field plotted on the x axis. This produces a single line running from left to right on the graph. Using a nominal field as an overlay produces multiple lines of different hues running from left to right on the graph.

Figure 1. Line plot with X Mode set to Sort



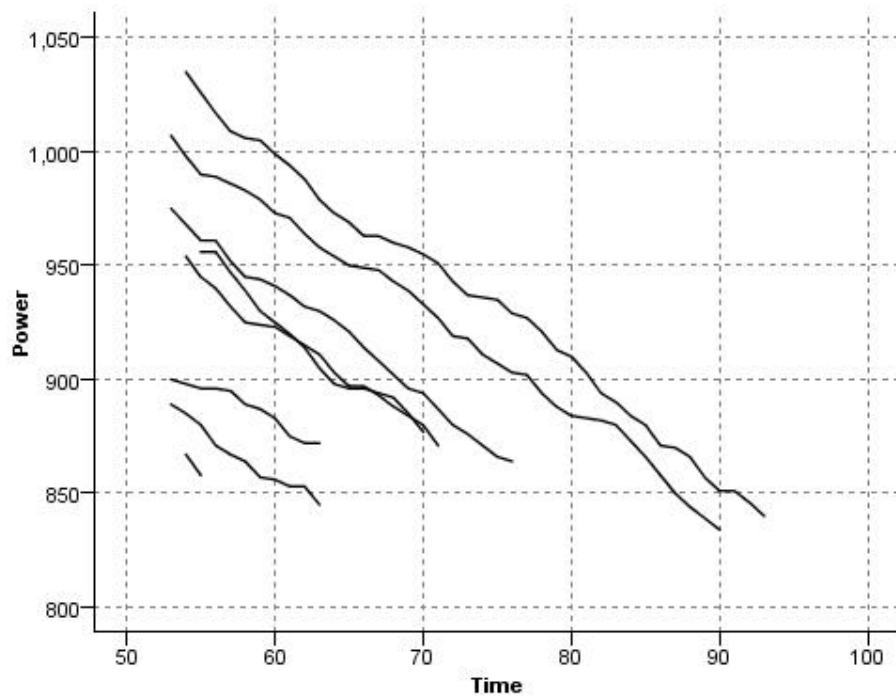
## X Mode = Overlay

---

Setting X Mode to Overlay creates multiple line plots on the same graph. Data are not sorted for an overlay plot; as long as the values on the x axis increase, data will be plotted on a single line. If the values decrease, a new line begins. For example, as x moves from 0 to 100, the y values will be plotted on a single line. When x falls below 100, a new line will be plotted in addition to the first one. The finished plot might have numerous plots useful for comparing several series of y values. This type of plot is useful for data with a periodic time component, such as electricity demand over successive 24-hour periods.

Figure 2. Line plot with X Mode set to Overlay

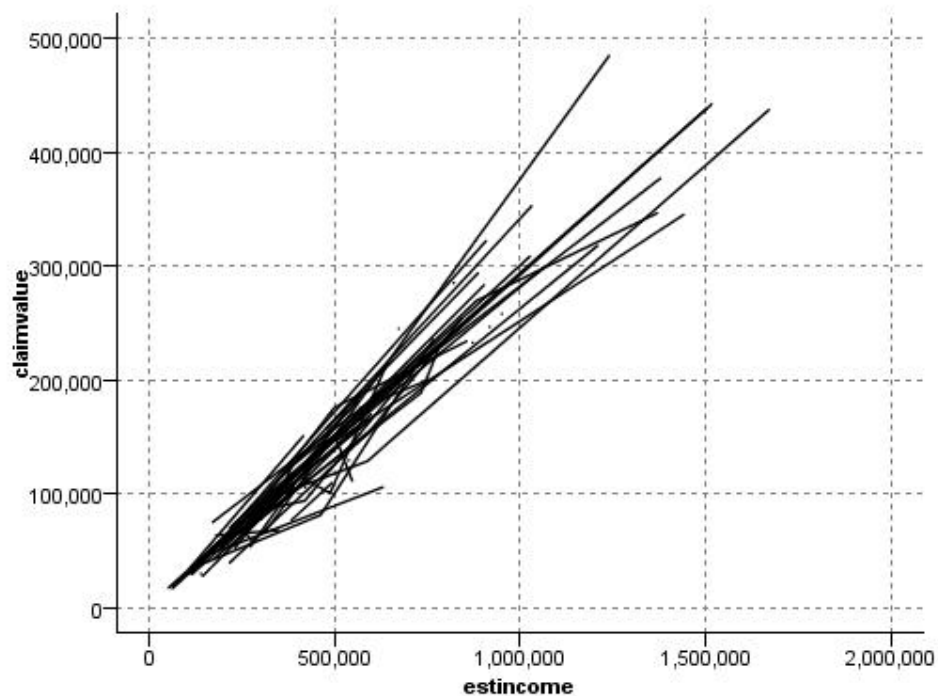




## X Mode = As Read

Setting X Mode to As Read plots  $x$  and  $y$  values as they are read from the data source. This option is useful for data with a time series component where you are interested in trends or patterns that depend on the order of the data. You may need to sort the data before creating this type of plot. It may also be useful to compare two similar plots with X Mode set to Sort and As Read in order to determine how much of a pattern depends on the sorting.

Figure 3. Line plot shown earlier as Sort, executed again with X Mode set to As Read



## Multiplot node

---

A multiplot is a special type of plot that displays multiple  $Y$  fields over a single  $X$  field. The  $Y$  fields are plotted as colored lines and each is equivalent to a Plot node with Style set to Line and X Mode set to Sort. Multiplots are useful when you have time sequence data and want to explore the fluctuation of several variables over time.

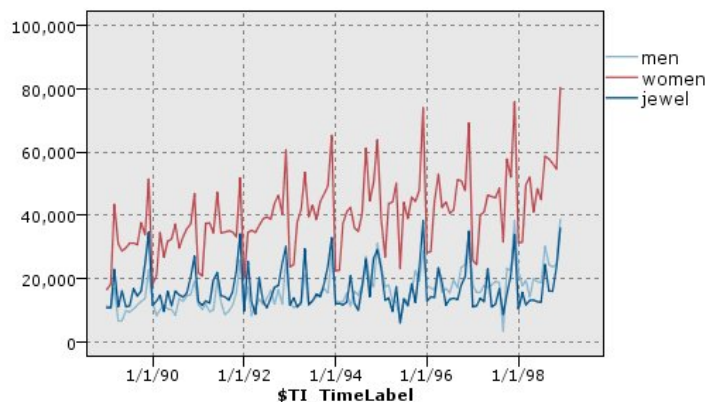
Parent topic: [Graphs](#)

## Time Plot node

---

Time Plot nodes allow you to view one or more time series plotted over time. The series you plot must contain numeric values and are assumed to occur over a range of time in which the periods are uniform.

Figure 1. Plotting sales of men's and women's clothing and jewelry over time



## Creating interventions and events

---

You can create Event and Intervention fields from the time plot by generating a derive (flag or nominal) node from the context menus. For example, you could create an event field in the case of a rail strike, where the drive state is True if the event happened and False otherwise. For an Intervention field, for a price rise for example, you could use a derive count to identify the date of the rise, with 0 for the old price and 1 for the new price. See [Derive node](#) for more information.

Parent topic: [Graphs](#)

## Distribution node

---

A distribution graph or table shows the occurrence of symbolic (non-numeric) values, such as mortgage type or gender, in a dataset. A typical use of the Distribution node is to show imbalances in the data that can be rectified by using a Balance node before creating a model. You can automatically generate a Balance node using the Generate menu in the distribution graph or table window.

Note: To show the occurrence of numeric values, you should use a Histogram node.

Parent topic: [Graphs](#)

## Histogram node

---

Histogram nodes show the occurrence of values for numeric fields. They are often used to explore the data before manipulations and model building. Similar to the Distribution node, Histogram nodes are frequently used to reveal imbalances in the data.

Note: To show the occurrence of values for symbolic fields, you should use a Distribution node.

**Parent topic:** [Graphs](#)

## Collection node

Collections are similar to histograms, but collections show the distribution of values for one numeric field relative to the values of another, rather than the occurrence of values for a single field. A collection is useful for illustrating a variable or field whose values change over time.

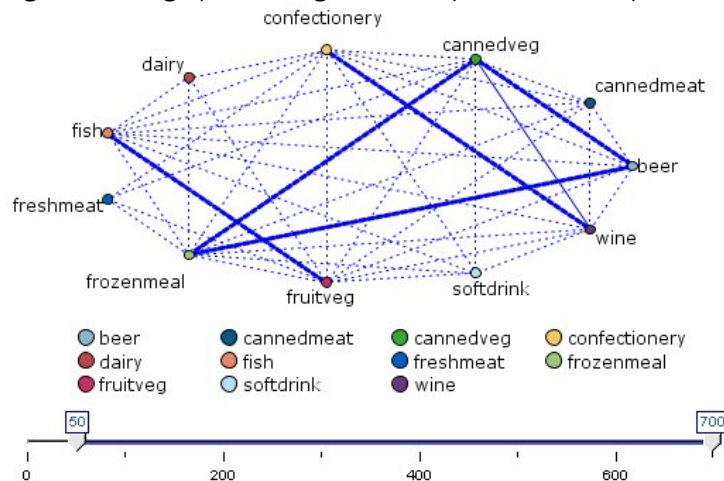
Using 3-D graphing, you can also include a symbolic axis displaying distributions by category. Two-dimensional collections are shown as stacked bar charts, with overlays where used.

**Parent topic:** [Graphs](#)

## Web node

Web nodes show the strength of relationships between values of two or more symbolic fields. The graph displays connections using varying types of lines to indicate connection strength. You can use a Web node, for example, to explore the relationship between the purchase of various items at an e-commerce site or a traditional retail outlet.

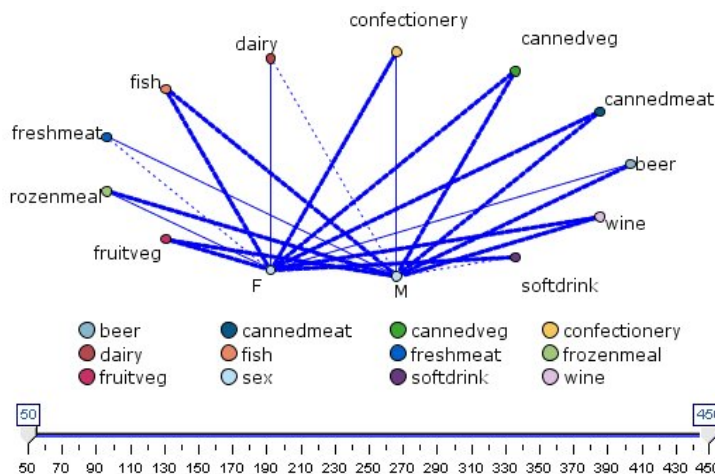
Figure 1. Web graph showing relationships between the purchase of grocery items



## Directed Webs

Directed Web nodes are similar to Web nodes in that they show the strength of relationships between symbolic fields. However, directed web graphs show connections only from one or more **From** fields to a single **To** field. The connections are unidirectional in the sense that they are one-way connections.

Figure 2. Directed web graph showing the relationship between the purchase of grocery items and gender



Like Web nodes, the graph displays connections using varying types of lines to indicate connection strength. You can use a Directed Web node, for example, to explore the relationship between gender and a proclivity for certain purchase items.

**Parent topic:** [Graphs](#)

## Evaluation node

The Evaluation node offers an easy way to evaluate and compare predictive models to choose the best model for your application. Evaluation charts show how models perform in predicting particular outcomes. They work by sorting records based on the predicted value and confidence of the prediction, splitting the records into groups of equal size (*quantiles*), and then plotting the value of the business criterion for each quantile, from highest to lowest. Multiple models are shown as separate lines in the plot.

Outcomes are handled by defining a specific value or range of values as a *hit*. Hits usually indicate success of some sort (such as a sale to a customer) or an event of interest (such as a specific medical diagnosis). You can define hit criteria under the OPTIONS section of the node properties, or you can use the default hit criteria as follows:

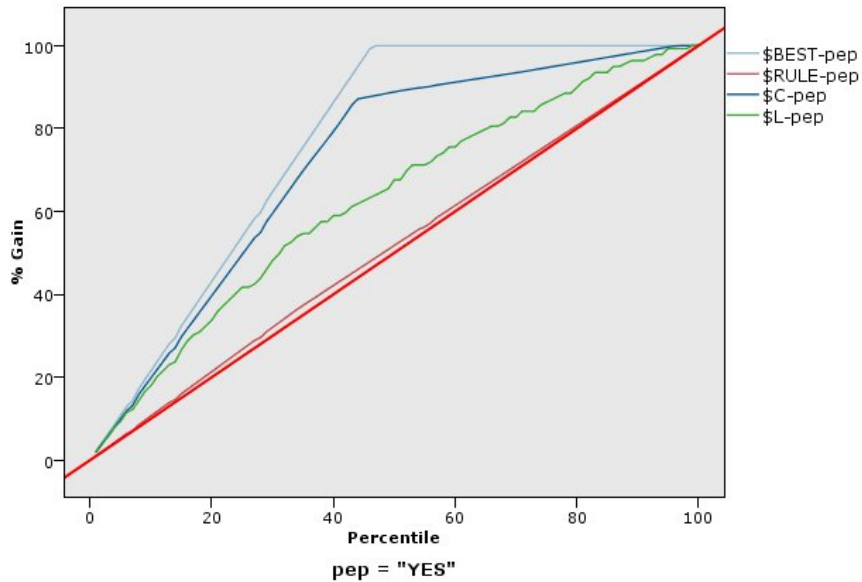
- Flag output fields are straightforward; hits correspond to *true* values.
- For Nominal output fields, the first value in the set defines a hit.
- For Continuous output fields, hits equal values greater than the midpoint of the field's range.

There are six types of evaluation charts, each of which emphasizes a different evaluation criterion.

## Gains charts

Gains are defined as the proportion of total hits that occurs in each quantile. Gains are computed as  $(\text{number of hits in quantile} / \text{total number of hits}) \times 100\%$ .

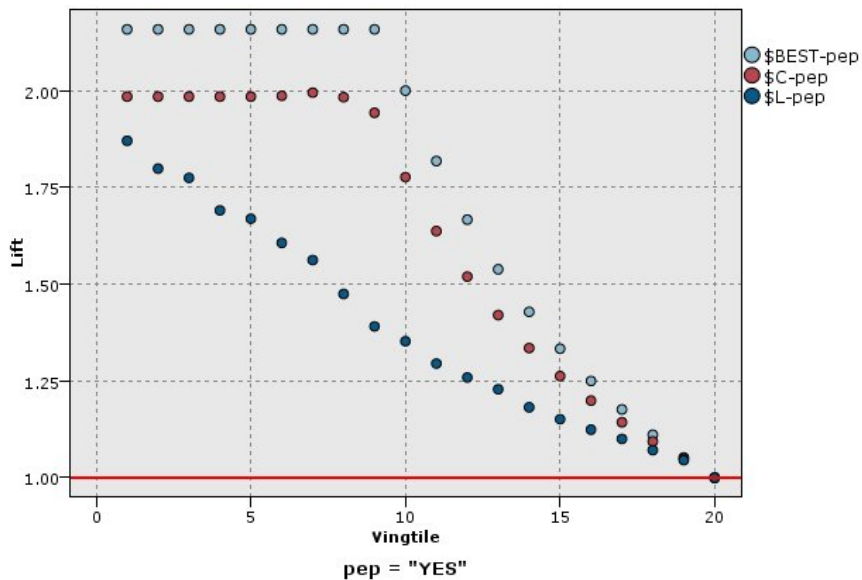
Figure 1. Gains chart (cumulative) with baseline, best line, and business rule displayed



## Lift charts

Lift compares the percentage of records in each quantile that are hits with the overall percentage of hits in the training data. It is computed as  $(\text{hits in quantile} / \text{records in quantile}) / (\text{total hits} / \text{total records})$ .

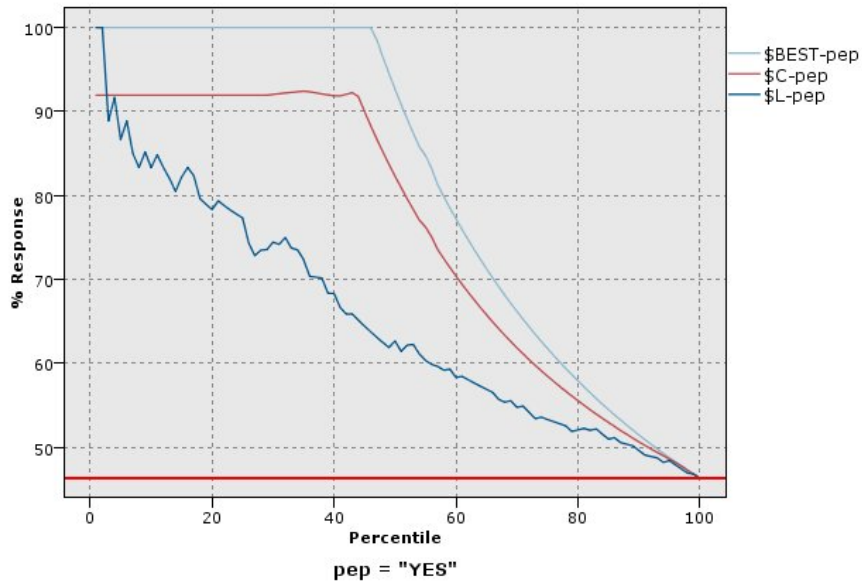
Figure 2. Lift chart (cumulative) using points and best line



## Response charts

Response is simply the percentage of records in the quantile that are hits. Response is computed as  $(\text{hits in quantile} / \text{records in quantile}) \times 100\%$ .

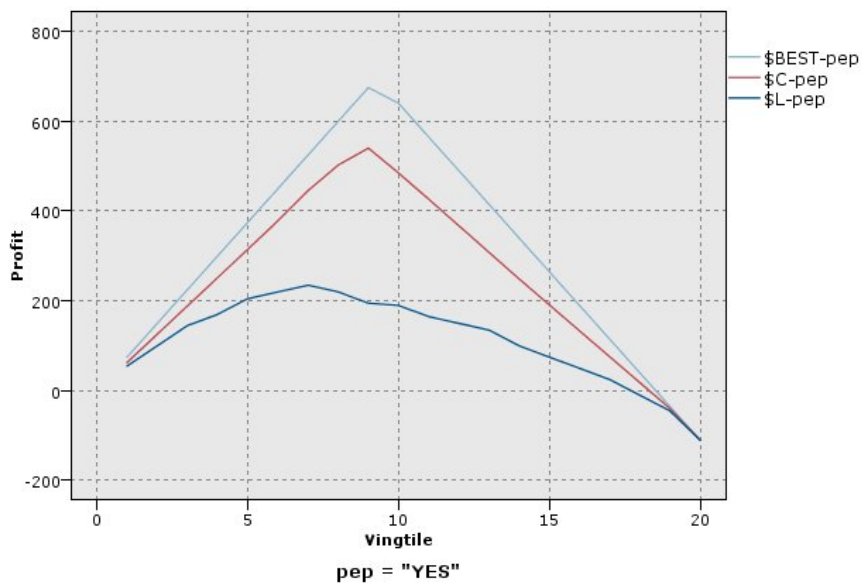
Figure 3. Response chart (cumulative) with best line



## Profit charts

Profit equals the *revenue* for each record minus the *cost* for the record. Profits for a quantile are simply the sum of profits for all records in the quantile. Revenues are assumed to apply only to hits, but costs apply to all records. Profits and costs can be fixed or can be defined by fields in the data. Profits are computed as (sum of revenue for records in quantile - sum of costs for records in quantile).

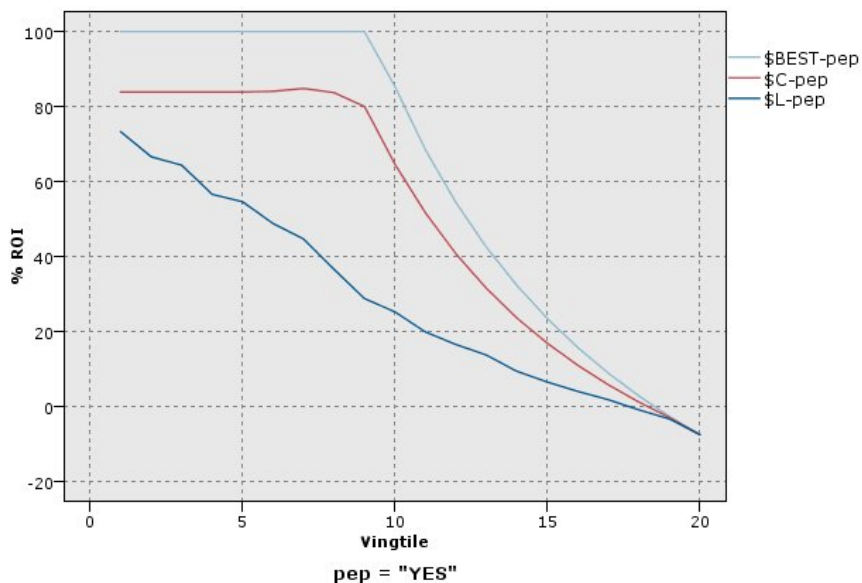
Figure 4. Profit chart (cumulative) with best line



## ROI charts

ROI (return on investment) is similar to profit in that it involves defining revenues and costs. ROI compares profits to costs for the quantile. ROI is computed as (profits for quantile / costs for quantile) x 100%.

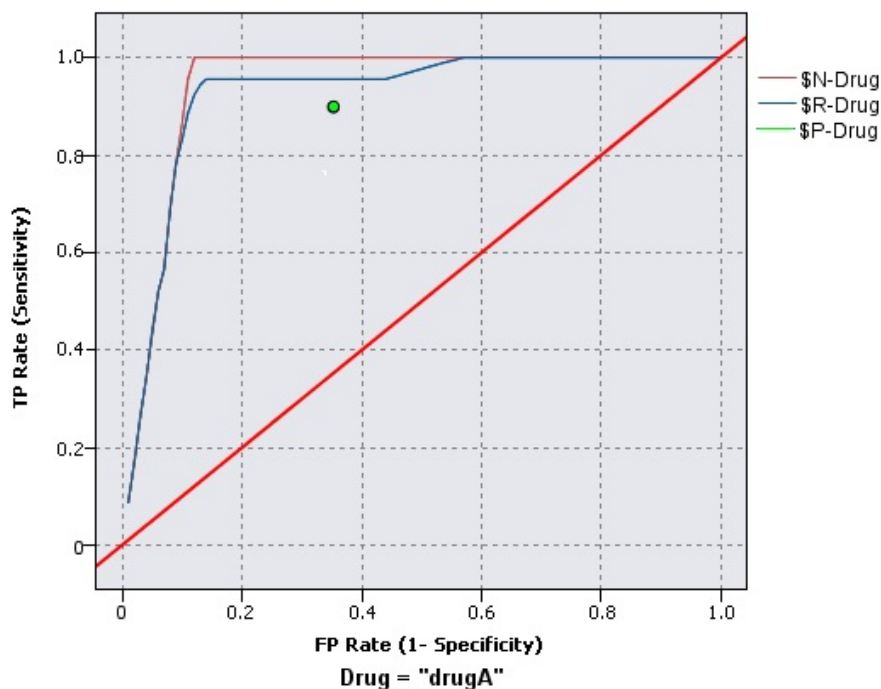
Figure 5. ROI chart (cumulative) with best line



## ROC charts

ROC (receiver operator characteristic) can only be used with binary classifiers. ROC can be used to visualize, organize and select classifiers based on their performance. A ROC chart plots the true positive rate (or sensitivity) against the false positive rate of the classifier. A ROC chart depicts the relative trade-offs between benefits (true positives) and costs (false positives). A true positive is an instance that is a hit and is classified as a hit. Therefore the true positive rate is calculated as the number of true positives / number of instances that are actually hits. A false positive is an instance that is a miss and is classified as a hit. Therefore the false positive rate is calculated as the number of false positives / number of instances that are actually misses.

Figure 6. ROC chart with best line



Evaluation charts can also be cumulative, so that each point equals the value for the corresponding quantile plus all higher quantiles. Cumulative charts usually convey the overall performance of models better, whereas noncumulative charts often excel at indicating particular problem areas for models.



Note: The Evaluation node doesn't support the use of commas in field names. If you have field names containing commas, you must either remove the commas or surround the field name in quotes.

**Parent topic:** [Graphs](#)

## Modeling

---

Watson Studio offers a variety of modeling methods taken from machine learning, artificial intelligence, and statistics.

The methods available on the palette allow you to derive new information from your data and to develop predictive models. Each method has certain strengths and is best suited for particular types of problems. For more information about modeling, see [Creating SPSS Modeler flows](#).

- **Auto Classifier node**  
The Auto Classifier node estimates and compares models for either nominal (set) or binary (yes/no) targets, using a number of different methods, enabling you to try out a variety of approaches in a single modeling run. You can select the algorithms to use, and experiment with multiple combinations of options. For example, rather than choose between Radial Basis Function, polynomial, sigmoid, or linear methods for an SVM, you can try them all. The node explores every possible combination of options, ranks each candidate model based on the measure you specify, and saves the best models for use in scoring or further analysis.
- **Auto Numeric node**  
The Auto Numeric node estimates and compares models for continuous numeric range outcomes using a number of different methods, enabling you to try out a variety of approaches in a single modeling run. You can select the algorithms to use, and experiment with multiple combinations of options. For example, you could predict housing values using neural net, linear regression, C&RT, and CHAID models to see which performs best, and you could try out different combinations of stepwise, forward, and backward regression methods. The node explores every possible combination of options, ranks each candidate model based on the measure you specify, and saves the best for use in scoring or further analysis.
- **Auto Cluster node**  
The Auto Cluster node estimates and compares clustering models that identify groups of records with similar characteristics. The node works in the same manner as other automated modeling nodes, enabling you to experiment with multiple combinations of options in a single modeling pass. Models can be compared using basic measures with which to attempt to filter and rank the usefulness of the cluster models, and provide a measure based on the importance of particular fields.
- **TCM node**  
Use this node to create a temporal causal model (TCM).
- **Bayes Net node**  
The Bayesian Network node enables you to build a probability model by combining observed and recorded evidence with "common-sense" real-world knowledge to establish the likelihood of occurrences by using seemingly unlinked attributes. The node focuses on Tree Augmented Naïve Bayes (TAN) and Markov Blanket networks that are primarily used for classification.
- **C5.0 node**  
This node uses the C5.0 algorithm to build either a *decision tree* or a *rule set*. A C5.0 model works by splitting the sample based on the field that provides the maximum *information gain*. Each sub-sample defined by the first split is then split again, usually based on a different field, and the process repeats until the subsamples cannot be split any further. Finally, the lowest-level splits are reexamined, and those that do not contribute significantly to the value of the model are removed or *pruned*.
- **C&R Tree node**  
The Classification and Regression (C&R) Tree node is a tree-based classification and prediction method. Similar to C5.0, this method uses recursive partitioning to split the training records into segments with similar output field values. The C&R Tree node starts by examining the input fields to find the best split, measured by the reduction in an impurity index that results from the split. The split defines two subgroups, each of which is subsequently split into two more subgroups, and so on, until one of the stopping criteria is triggered. All splits are binary (only two subgroups).
- **CHAID node**  
CHAID, or Chi-squared Automatic Interaction Detection, is a classification method for building decision trees by using chi-square statistics to identify optimal splits.



- **QUEST node**  
QUEST - or Quick, Unbiased, Efficient Statistical Tree - is a binary classification method for building decision trees. A major motivation in its development was to reduce the processing time required for large C&R Tree analyses with either many variables or many cases. A second goal of QUEST was to reduce the tendency found in classification tree methods to favor inputs that allow more splits, that is, continuous (numeric range) input fields or those with many categories.
- **Tree-AS node**  
The Tree-AS node can be used with data in a distributed environment. With this node, you can choose to build decision trees using either a CHAID or Exhaustive CHAID model.
- **Random Trees node**  
The Random Trees node can be used with data in a distributed environment. In this node, you build an ensemble model that consists of multiple decision trees.
- **Random Forest node**  
Random Forest® is an advanced implementation of a bagging algorithm with a tree model as the base model.
- **Decision List node**  
Decision List models identify subgroups or *segments* that show a higher or lower likelihood of a binary (yes or no) outcome relative to the overall sample.
- **Time Series node**  
The Time Series node can be used with data in either a local or distributed environment. With this node, you can choose to estimate and build exponential smoothing, univariate Autoregressive Integrated Moving Average (ARIMA), or multivariate ARIMA (or transfer function) models for time series, and produce forecasts based on the time series data.
- **GenLin node**  
The generalized linear model expands the general linear model so that the dependent variable is linearly related to the factors and covariates via a specified link function. Moreover, the model allows for the dependent variable to have a non-normal distribution. It covers widely used statistical models, such as linear regression for normally distributed responses, logistic models for binary data, loglinear models for count data, complementary log-log models for interval-censored survival data, plus many other statistical models through its very general model formulation.
- **GLMM node**  
This node creates a generalized linear mixed model (GLMM).
- **GLE node**  
The GLE model identifies the dependent variable that is linearly related to the factors and covariates via a specified link function. Moreover, the model allows for the dependent variable to have a non-normal distribution. It covers widely used statistical models, such as linear regression for normally distributed responses, logistic models for binary data, loglinear models for count data, complementary log-log models for interval-censored survival data, plus many other statistical models through its very general model formulation.
- **Linear node**  
Linear regression is a common statistical technique for classifying records based on the values of numeric input fields. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values.
- **Linear-AS node**  
Linear regression is a common statistical technique for classifying records based on the values of numeric input fields. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values.
- **Regression node**  
Linear regression is a common statistical technique for classifying records based on the values of numeric input fields. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values.
- **LSVM node**  
With the LSVM node, you can use a linear support vector machine to classify data. LSVM is particularly suited for use with wide datasets--that is, those with a large number of predictor fields. You can use the default settings on the node to produce a basic model relatively quickly, or you can use the build options to experiment with different settings.
- **Logistic node**  
Logistic regression, also known as *nominal regression*, is a statistical technique for classifying records based on

values of input fields. It is analogous to linear regression but takes a categorical target field instead of a numeric one. Both binomial models (for targets with two discrete categories) and multinomial models (for targets with more than two categories) are supported.

- **Neural Net node**

A *neural network* can approximate a wide range of predictive models with minimal demands on model structure and assumption. The form of the relationships is determined during the learning process. If a linear relationship between the target and predictors is appropriate, the results of the neural network should closely approximate those of a traditional linear model. If a nonlinear relationship is more appropriate, the neural network will automatically approximate the "correct" model structure.

- **KNN node**

Nearest Neighbor Analysis is a method for classifying cases based on their similarity to other cases. In machine learning, it was developed as a way to recognize patterns of data without requiring an exact match to any stored patterns, or cases. Similar cases are near each other and dissimilar cases are distant from each other. Thus, the distance between two cases is a measure of their dissimilarity.

- **Cox node**

Cox Regression builds a predictive model for time-to-event data. The model produces a survival function that predicts the probability that the event of interest has occurred at a given time  $t$  for given values of the predictor variables. The shape of the survival function and the regression coefficients for the predictors are estimated from observed subjects; the model can then be applied to new cases that have measurements for the predictor variables.

- **PCA/Factor node**

The PCA/Factor node provides powerful data-reduction techniques to reduce the complexity of your data. Two similar but distinct approaches are provided.

- **SVM node**

The SVM node uses a support vector machine to classify data. SVM is particularly suited for use with wide datasets, that is, those with a large number of predictor fields. You can use the default settings on the node to produce a basic model relatively quickly, or you can use the Expert settings to experiment with different types of SVM models.

- **Feature Selection node**

Data mining problems may involve hundreds, or even thousands, of fields that can potentially be used as inputs. As a result, a great deal of time and effort may be spent examining which fields or variables to include in the model. To narrow down the choices, the Feature Selection algorithm can be used to identify the fields that are most important for a given analysis. For example, if you are trying to predict patient outcomes based on a number of factors, which factors are the most likely to be important?

- **Discriminant node**

Discriminant analysis builds a predictive model for group membership. The model is composed of a discriminant function (or, for more than two groups, a set of discriminant functions) based on linear combinations of the predictor variables that provide the best discrimination between the groups. The functions are generated from a sample of cases for which group membership is known; the functions can then be applied to new cases that have measurements for the predictor variables but have unknown group membership.

- **SLRM node**

Use the Self-Learning Response Model (SLRM) node to build a model that you can continually update, or reestimate, as a dataset grows without having to rebuild the model every time using the complete dataset. For example, this is useful when you have several products and you want to identify which one a customer is most likely to buy if you offer it to them. This model allows you to predict which offers are most appropriate for customers and the probability of the offers being accepted.

- **Spatio-Temporal Prediction (STP) node**

Spatio-Temporal Prediction (STP) has many potential applications such as energy management for buildings or facilities, performance analysis and forecasting for mechanical service engineers, or public transport planning. In these applications, measurements such as energy usage are often taken over space and time. Questions that might be relevant to the recording of these measurements include what factors will affect future observations, what can be done to effect a wanted change, or to better manage the system? To address these questions, you can use statistical techniques that can forecast future values at different locations, and can explicitly model adjustable factors to perform what-if analysis.

- **Association Rules node**

Association rules associate a particular conclusion (the purchase of a particular product, for example) with a

set of conditions (the purchase of several other products, for example).

- **Apriori node**  
The Apriori node discovers association rules in your data.
- **CARMA node**  
The CARMA node uses an association rules discovery algorithm to discover association rules in the data.
- **Sequence node**  
The Sequence node discovers patterns in sequential or time-oriented data, in the format `bread -> cheese`. The elements of a sequence are *item sets* that constitute a single transaction.
- **Kohonen node**  
Kohonen networks are a type of neural network that perform clustering, also known as a *knet* or a *self-organizing map*. This type of network can be used to cluster the dataset into distinct groups when you don't know what those groups are at the beginning. Records are grouped so that records within a group or cluster tend to be similar to each other, and records in different groups are dissimilar.
- **Anomaly node**  
Anomaly detection models are used to identify outliers, or unusual cases, in the data. Unlike other modeling methods that store rules about unusual cases, anomaly detection models store information on what normal behavior looks like. This makes it possible to identify outliers even if they do not conform to any known pattern, and it can be particularly useful in applications, such as fraud detection, where new patterns may constantly be emerging. Anomaly detection is an unsupervised method, which means that it does not require a training dataset containing known cases of fraud to use as a starting point.
- **K-Means node**  
The K-Means node provides a method of *cluster analysis*. It can be used to cluster the dataset into distinct groups when you don't know what those groups are at the beginning. Unlike most learning methods in SPSS Modeler, K-Means models do *not* use a target field. This type of learning, with no target field, is called *unsupervised learning*. Instead of trying to predict an outcome, K-Means tries to uncover patterns in the set of input fields. Records are grouped so that records within a group or cluster tend to be similar to each other, but records in different groups are dissimilar.
- **TwoStep cluster node**  
The TwoStep Cluster node provides a form of *cluster analysis*. It can be used to cluster the dataset into distinct groups when you don't know what those groups are at the beginning. As with Kohonen nodes and K-Means nodes, TwoStep Cluster models do *not* use a target field. Instead of trying to predict an outcome, TwoStep Cluster tries to uncover patterns in the set of input fields. Records are grouped so that records within a group or cluster tend to be similar to each other, but records in different groups are dissimilar.
- **TwoStep-AS cluster node**  
TwoStep Cluster is an exploratory tool that is designed to reveal natural groupings (or clusters) within a data set that would otherwise not be apparent. The algorithm that is employed by this procedure has several desirable features that differentiate it from traditional clustering techniques.
- **Isotonic-AS node**  
Isotonic Regression belongs to the family of regression algorithms. The Isotonic-AS node in Watson Studio is implemented in Spark.
- **XGBoost-AS node**  
XGBoost® is an advanced implementation of a gradient boosting algorithm. Boosting algorithms iteratively learn weak classifiers and then add them to a final strong classifier. XGBoost is very flexible and provides many parameters that can be overwhelming to most users, so the XGBoost-AS node in Watson Studio exposes the core features and commonly used parameters. The XGBoost-AS node is implemented in Spark.
- **K-Means-AS node**  
K-Means is one of the most commonly used clustering algorithms. It clusters data points into a predefined number of clusters. The K-Means-AS node in SPSS Modeler is implemented in Spark.
- **XGBoost Tree node**  
XGBoost Tree® is an advanced implementation of a gradient boosting algorithm with a tree model as the base model. Boosting algorithms iteratively learn weak classifiers and then add them to a final strong classifier. XGBoost Tree is very flexible and provides many parameters that can be overwhelming to most users, so the XGBoost Tree node in Watson Studio exposes the core features and commonly used parameters. The node is implemented in Python.
- **XGBoost Linear node**  
XGBoost Linear® is an advanced implementation of a gradient boosting algorithm with a linear model as the

base model. Boosting algorithms iteratively learn weak classifiers and then add them to a final strong classifier. The XGBoost Linear node in Watson Studio is implemented in Python.

- **Gaussian Mixture node**

A Gaussian Mixture<sup>®</sup> model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters.

- **KDE node**

Kernel Density Estimation (KDE)<sup>®</sup> uses the Ball Tree or KD Tree algorithms for efficient queries, and walks the line between unsupervised learning, feature engineering, and data modeling.

- **One-Class SVM node**

The One-Class SVM<sup>®</sup> node uses an unsupervised learning algorithm. The node can be used for novelty detection. It will detect the soft boundary of a given set of samples, to then classify new points as belonging to that set or not. This One-Class SVM modeling node is implemented in Python and requires the scikit-learn<sup>®</sup> Python library.

- **MultiLayerPerceptron-AS node**

Multilayer perceptron is a classifier based on the feedforward artificial neural network and consists of multiple layers.

- **HDBSCAN node**

Hierarchical Density-Based Spatial Clustering (HDBSCAN)<sup>®</sup> uses unsupervised learning to find clusters, or dense regions, of a data set.

- **Extension Model node**

With the Extension Model node, you can run R scripts or Python for Spark scripts to build and score models.

**Parent topic:** [Nodes palette](#)

## Auto Classifier node

---

The Auto Classifier node estimates and compares models for either nominal (set) or binary (yes/no) targets, using a number of different methods, enabling you to try out a variety of approaches in a single modeling run. You can select the algorithms to use, and experiment with multiple combinations of options. For example, rather than choose between Radial Basis Function, polynomial, sigmoid, or linear methods for an SVM, you can try them all. The node explores every possible combination of options, ranks each candidate model based on the measure you specify, and saves the best models for use in scoring or further analysis.

### Example

A retail company has historical data tracking the offers made to specific customers in past campaigns. The company now wants to achieve more profitable results by matching the right offer to each customer.

### Requirements

A target field with a measurement level of either `Nominal` or `Flag` (with the role set to `Target`), and at least one input field (with the role set to `Input`). For a flag field, the `True` value defined for the target is assumed to represent a hit when calculating profits, lift, and related statistics. Input fields can have a measurement level of `Continuous` or `Categorical`, with the limitation that some inputs may not be appropriate for some model types. For example, ordinal fields used as inputs in C&R Tree, CHAID, and QUEST models must have numeric storage (not string), and will be ignored by these models if specified otherwise. Similarly, continuous input fields can be binned in some cases. The requirements are the same as when using the individual modeling nodes; for example, a Bayes Net model works the same whether generated from the Bayes Net node or the Auto Classifier node.

### Frequency and weight fields

Frequency and weight are used to give extra importance to some records over others because, for example, the user knows that the build dataset under-represents a section of the parent population (`Weight`) or because one record represents a number of identical cases (`Frequency`). If specified, a frequency field can be used by C&R Tree, CHAID, QUEST, Decision List, and Bayes Net models. A weight field can be used by C&RT, CHAID, and C5.0 models. Other model types will ignore these fields and build the models anyway. Frequency and weight fields are used only for model building, and are not considered when evaluating or scoring models.

### Prefixes

If you attach a table node to the nugget for the Auto Classifier Node, there are several new variables in the table with names that begin with a \$ prefix.

The names of the fields that are generated during scoring are based on the target field, but with a standard prefix. Different model types use different sets of prefixes.

For example, the prefixes \$G, \$R, \$C are used as the prefix for predictions that are generated by the Generalized Linear model, CHAID model, and C5.0 model, respectively. \$X is typically generated by using an ensemble, and \$XR, \$XS, and \$XF are used as prefixes in cases where the target field is a Continuous, Categorical, or Flag field, respectively.

\$.C prefixes are used for prediction confidence of a Categorical, or Flag target; for example, \$XFC is used as a prefix for ensemble Flag prediction confidence. \$RC and \$CC are the prefixes for a single prediction of confidence for a CHAID model and C5.0 model respectively.

## Supported Model Types

---

Supported model types include Neural Net, C&R Tree, QUEST, CHAID, C5.0, Logistic Regression, Decision List, Bayes Net, Discriminant, Nearest Neighbor, SVM, XGBoost Tree, and XGBoost-AS.

## Cross-validation settings

---

In the node properties, note that cross-validation settings are available. Cross-validation is a valuable technique for testing the effectiveness (avoiding overfitting) of machine learning models, and it's also a re-sampling procedure you can use to evaluate a model if you have limited data.

K-fold is a popular and easy way to perform cross-validation. It generally results in a less biased model compared to a single train/test partition, because it ensures that every observation from the original dataset has the chance of appearing in training and test sets. The general procedure of k-fold cross-validation is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k-folds/groups.
3. For each unique fold/group:
  - a. Take the fold/group as a hold out or test dataset.
  - b. Take the remaining groups as a training dataset.
  - c. Fit a model on the training set and evaluate it on the test set.
  - d. Retain the evaluation score and discard the model.
4. Summarize the overall evaluation of the model using the retained k-fold evaluation scores.

Cross-validation is currently supported via the Auto Classifier node and the Auto Numeric node. Double-click the node to open its properties. By selecting the Cross-validate option, a single train/test partition is disabled and the Auto nodes will use k-fold cross-validation to evaluate the selected set of different algorithms.

You can specify the Number of folds (K), The default is 5, with a range of 3 to 10. If you want to retain repeatable sampling during cross-validation, to have consistent final evaluation measures for generated models across different executions, you can select the Repeatable Cross Validation partition assignment option. You can also set the Random seed to a specific value so the resulting model is exactly reproducible. Or click Generate to always generate the same sequence of random values, in which case running the node always yields the same generated model.

**Parent topic:** [Modeling](#)

## Auto Numeric node

---

The Auto Numeric node estimates and compares models for continuous numeric range outcomes using a number of different methods, enabling you to try out a variety of approaches in a single modeling run. You can select the algorithms to use, and experiment with multiple combinations of options. For example, you could predict housing values using neural net, linear regression, C&RT, and CHAID models to see which performs best, and you could try out different combinations of stepwise, forward, and backward regression methods. The node explores every possible combination of options, ranks each candidate model based on the measure you specify, and saves the best for use in scoring or further analysis.

### Example

A municipality wants to more accurately estimate real estate taxes and to adjust values for specific properties as needed without having to inspect every property. Using the Auto Numeric node, the analyst can generate and compare a number of models that predict property values based on building type, neighborhood, size, and other known factors.

### Requirements

A single target field (with the role set to Target), and at least one input field (with the role set to Input). The target must be a continuous (numeric range) field, such as *age* or *income*. Input fields can be continuous or categorical, with the limitation that some inputs may not be appropriate for some model types. For example, C&R Tree models can use categorical string fields as inputs, while linear regression models cannot use these fields and will ignore them if specified. The requirements are the same as when using the individual modeling nodes. For example, a CHAID model works the same whether generated from the CHAID node or the Auto Numeric node.

### Frequency and weight fields

Frequency and weight are used to give extra importance to some records over others because, for example, the user knows that the build dataset under-represents a section of the parent population (Weight) or because one record represents a number of identical cases (Frequency). If specified, a frequency field can be used by C&R Tree and CHAID algorithms. A weight field can be used by C&RT, CHAID, Regression, and GenLin algorithms. Other model types will ignore these fields and build the models anyway. Frequency and weight fields are used only for model building and are not considered when evaluating or scoring models.

### Prefixes

If you attach a table node to the nugget for the Auto Numeric Node, there are several new variables in the table with names that begin with a \$ prefix.

The names of the fields that are generated during scoring are based on the target field, but with a standard prefix. Different model types use different sets of prefixes.

For example, the prefixes \$G, \$R, \$C are used as the prefix for predictions that are generated by the Generalized Linear model, CHAID model, and C5.0 model, respectively. \$X is typically generated by using an ensemble, and \$XR, \$XS, and \$XF are used as prefixes in cases where the target field is a Continuous, Categorical, or Flag field, respectively.

\$.E prefixes are used for the prediction confidence of a Continuous target; for example, \$XRE is used as a prefix for ensemble Continuous prediction confidence. \$GE is the prefix for a single prediction of confidence for a Generalized Linear model.

## Supported model types

---

Supported model types include Neural Net, C&R Tree, CHAID, Regression, GenLin, Nearest Neighbor, SVM, XGBoost Linear, GLE, and XGBoost-AS.

## Cross-validation settings

---

In the node properties, note that cross-validation settings are available. Cross-validation is a valuable technique for testing the effectiveness (avoiding overfitting) of machine learning models, and it's also a re-sampling procedure you can use to evaluate a model if you have limited data.

K-fold is a popular and easy way to perform cross-validation. It generally results in a less biased model compared to a single train/test partition, because it ensures that every observation from the original dataset has the chance of appearing in training and test sets. The general procedure of k-fold cross-validation is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k-folds/groups.
3. For each unique fold/group:
  - a. Take the fold/group as a hold out or test dataset.
  - b. Take the remaining groups as a training dataset.
  - c. Fit a model on the training set and evaluate it on the test set.
  - d. Retain the evaluation score and discard the model.
4. Summarize the overall evaluation of the model using the retained k-fold evaluation scores.



Cross-validation is currently supported via the Auto Classifier node and the Auto Numeric node. Double-click the node to open its properties. By selecting the Cross-validate option, a single train/test partition is disabled and the Auto nodes will use k-fold cross-validation to evaluate the selected set of different algorithms.

You can specify the Number of folds (K), The default is 5, with a range of 3 to 10. If you want to retain repeatable sampling during cross-validation, to have consistent final evaluation measures for generated models across different executions, you can select the Repeatable Cross Validation partition assignment option. You can also set the Random seed to a specific value so the resulting model is exactly reproducible. Or click Generate to always generate the same sequence of random values, in which case running the node always yields the same generated model.

**Parent topic:** [Modeling](#)

## Auto Cluster node

---

The Auto Cluster node estimates and compares clustering models that identify groups of records with similar characteristics. The node works in the same manner as other automated modeling nodes, enabling you to experiment with multiple combinations of options in a single modeling pass. Models can be compared using basic measures with which to attempt to filter and rank the usefulness of the cluster models, and provide a measure based on the importance of particular fields.

Clustering models are often used to identify groups that can be used as inputs in subsequent analyses. For example, you may want to target groups of customers based on demographic characteristics such as income, or based on the services they have bought in the past. You can do this without prior knowledge about the groups and their characteristics -- you may not know how many groups to look for, or what features to use in defining them. Clustering models are often referred to as unsupervised learning models, since they do not use a target field, and do not return a specific prediction that can be evaluated as true or false. The value of a clustering model is determined by its ability to capture interesting groupings in the data and provide useful descriptions of those groupings.

**Requirements.** One or more fields that define characteristics of interest. Cluster models do not use target fields in the same manner as other models, because they do not make specific predictions that can be assessed as true or false. Instead, they are used to identify groups of cases that may be related. For example, you cannot use a cluster model to predict whether a given customer will churn or respond to an offer. But you can use a cluster model to assign customers to groups based on their tendency to do those things. Weight and frequency fields are not used.

**Evaluation fields.** While no target is used, you can optionally specify one or more evaluation fields to be used in comparing models. The usefulness of a cluster model may be evaluated by measuring how well (or badly) the clusters differentiate these fields.

## Supported model types

---

Supported model types include TwoStep, K-Means, Kohonen, One-Class SVM, and K-Means-AS.

**Parent topic:** [Modeling](#)

## TCM node

---

Use this node to create a temporal causal model (TCM).

Temporal causal modeling attempts to discover key causal relationships in time series data. In temporal causal modeling, you specify a set of target series and a set of candidate inputs to those targets. The procedure then builds an autoregressive time series model for each target and includes only those inputs that have a causal relationship with the target. This approach differs from traditional time series modeling where you must explicitly specify the predictors for a target series. Since temporal causal modeling typically involves building models for multiple related time series, the result is referred to as a *model system*.

In the context of temporal causal modeling, the term *causal* refers to Granger causality. A time series X is said to "Granger cause" another time series Y if regressing for Y in terms of past values of both X and Y results in a better model for Y than regressing only on past values of Y.

## Examples

---

Business decision makers can use temporal causal modeling to uncover causal relationships within a large set of time-based metrics that describe the business. The analysis might reveal a few controllable inputs, which have the largest impact on key performance indicators.

Managers of large IT systems can use temporal causal modeling to detect anomalies in a large set of interrelated operational metrics. The causal model then allows going beyond anomaly detection and discovering the most likely root causes of the anomalies.

## Field requirements

---

There must be at least one target. By default, fields with a predefined role of `None` are not used.

## Data structure

---

Temporal causal modeling supports two types of data structures:

### Column-based data

For column-based data, each time series field contains the data for a single time series. This structure is the traditional structure of time series data, as used by the Time Series Modeler.

### Multidimensional data

For multidimensional data, each time series field contains the data for multiple time series. Separate time series, within a particular field, are then identified by a set of values of categorical fields referred to as *dimension* fields. For example, sales data for two different sales channels (retail and web) might be stored in a single `sales` field. A dimension field named `channel`, with values `retail` and `web`, identifies the records that are associated with each of the two sales channels.

Note: To build a temporal causal model, you need enough data points. The product uses the constraint:

$$m > (L + KL + 1)$$

where  $m$  is the number of data points,  $L$  is the number of lags, and  $K$  is the number of predictors. Make sure your data set is big enough so that the number of data points ( $m$ ) satisfies the condition.

**Parent topic:** [Modeling](#)

## Bayes Net node

---

The Bayesian Network node enables you to build a probability model by combining observed and recorded evidence with "common-sense" real-world knowledge to establish the likelihood of occurrences by using seemingly unlinked attributes. The node focuses on Tree Augmented Naive Bayes (TAN) and Markov Blanket networks that are primarily used for classification.

Bayesian networks are used for making predictions in many varied situations; some examples are:

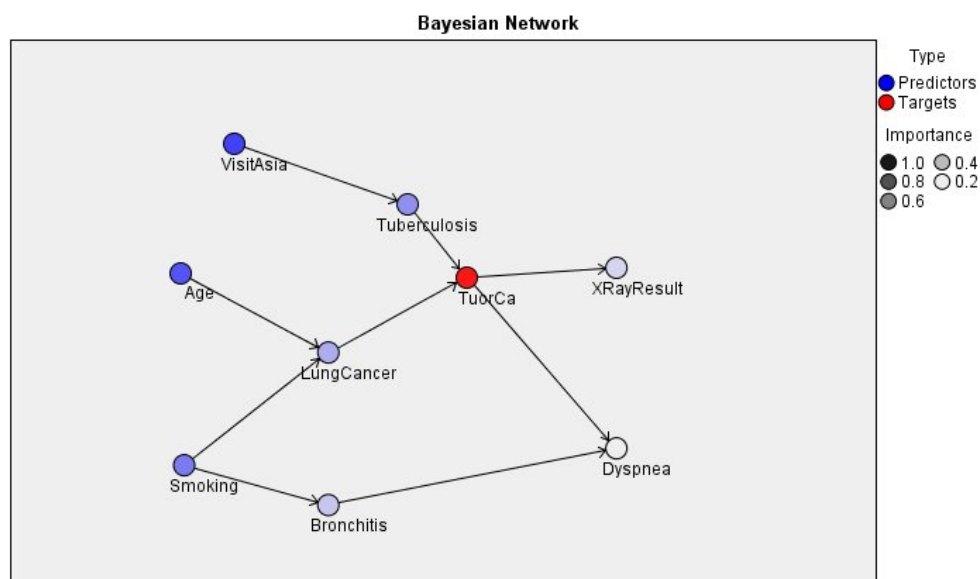
- Selecting loan opportunities with low default risk.
- Estimating when equipment will need service, parts, or replacement, based on sensor input and existing records.
- Resolving customer problems via online troubleshooting tools.
- Diagnosing and troubleshooting cellular telephone networks in real-time.
- Assessing the potential risks and rewards of research-and-development projects in order to focus resources on the best opportunities.



A Bayesian network is a graphical model that displays variables (often referred to as **nodes**) in a dataset and the probabilistic, or conditional, independencies between them. Causal relationships between nodes may be represented by a Bayesian network; however, the links in the network (also known as **arcs**) do not necessarily represent direct cause and effect. For example, a Bayesian network can be used to calculate the probability of a patient having a specific disease, given the presence or absence of certain symptoms and other relevant data, if the probabilistic independencies between symptoms and disease as displayed on the graph hold true. Networks are very robust where information is missing and make the best possible prediction using whatever information is present.

A common, basic, example of a Bayesian network was created by Lauritzen and Spiegelhalter (1988). It is often referred to as the "Asia" model and is a simplified version of a network that may be used to diagnose a doctor's new patients; the direction of the links roughly corresponding to causality. Each node represents a facet that may relate to the patient's condition; for example, "Smoking" indicates that they are a confirmed smoker, and "VisitAsia" shows if they recently visited Asia. Probability relationships are shown by the links between any nodes; for example, smoking increases the chances of the patient developing both bronchitis and lung cancer, whereas age only seems to be associated with the possibility of developing lung cancer. In the same way, abnormalities on an x-ray of the lungs may be caused by either tuberculosis or lung cancer, while the chances of a patient suffering from shortness of breath (dyspnea) are increased if they also suffer from either bronchitis or lung cancer.

Figure 1. Lauritzen and Spiegelhalter's Asia network example



There are several reasons why you might decide to use a Bayesian network:

- It helps you learn about causal relationships. From this, it enables you to understand a problem area and to predict the consequences of any intervention.
- The network provides an efficient approach for avoiding the overfitting of data.
- A clear visualization of the relationships involved is easily observed.

**Requirements.** Target fields must be categorical and can have a measurement level of *Nominal*, *Ordinal*, or *Flag*. Inputs can be fields of any type. Continuous (numeric range) input fields will be automatically binned; however, if the distribution is skewed, you may obtain better results by manually binning the fields using a Binning node before the Bayesian Network node. For example, use Optimal Binning where the Supervisor field is the same as the Bayesian Network node Target field.

**Example.** An analyst for a bank wants to be able to predict customers, or potential customers, who are likely to default on their loan repayments. You can use a Bayesian network model to identify the characteristics of customers most likely to default, and build several different types of model to establish which is the best at predicting potential defaulters.

**Example.** A telecommunications operator wants to reduce the number of customers who leave the business (known as "churn"), and update the model on a monthly basis using each preceding month's data. You can use a Bayesian

network model to identify the characteristics of customers most likely to churn, and continue training the model each month with the new data.

**Parent topic:** [Modeling](#)

## C5.0 node

---

This node uses the C5.0 algorithm to build either a *decision tree* or a *rule set*. A C5.0 model works by splitting the sample based on the field that provides the maximum *information gain*. Each sub-sample defined by the first split is then split again, usually based on a different field, and the process repeats until the subsamples cannot be split any further. Finally, the lowest-level splits are reexamined, and those that do not contribute significantly to the value of the model are removed or *pruned*.

Note: The C5.0 node can predict only a categorical target. When analyzing data with categorical (nominal or ordinal) fields, the node is likely to group categories together.

C5.0 can produce two kinds of models. A *decision tree* is a straightforward description of the splits found by the algorithm. Each terminal (or "leaf") node describes a particular subset of the training data, and each case in the training data belongs to exactly one terminal node in the tree. In other words, exactly one prediction is possible for any particular data record presented to a decision tree.

In contrast, a *rule set* is a set of rules that tries to make predictions for individual records. Rule sets are derived from decision trees and, in a way, represent a simplified or distilled version of the information found in the decision tree. Rule sets can often retain most of the important information from a full decision tree but with a less complex model. Because of the way rule sets work, they do not have the same properties as decision trees. The most important difference is that with a rule set, more than one rule may apply for any particular record, or no rules at all may apply. If multiple rules apply, each rule gets a weighted "vote" based on the confidence associated with that rule, and the final prediction is decided by combining the weighted votes of all of the rules that apply to the record in question. If no rule applies, a default prediction is assigned to the record.

Example. A medical researcher has collected data about a set of patients, all of whom suffered from the same illness. During their course of treatment, each patient responded to one of five medications. You can use a C5.0 model, in conjunction with other nodes, to help find out which drug might be appropriate for a future patient with the same illness.

Requirements. To train a C5.0 model, there must be one categorical (i.e., nominal or ordinal) **Target** field, and one or more **Input** fields of any type. Fields set to **Both** or **None** are ignored. Fields used in the model must have their types fully instantiated. A weight field can also be specified.

Strengths. C5.0 models are quite robust in the presence of problems such as missing data and large numbers of input fields. They usually do not require long training times to estimate. In addition, C5.0 models tend to be easier to understand than some other model types, since the rules derived from the model have a very straightforward interpretation. C5.0 also offers the powerful *boosting* method to increase accuracy of classification.

Tip: C5.0 model building speed may benefit from enabling parallel processing.

Note: When first creating a flow, you select which runtime to use. By default, flows use the IBM SPSS Modeler runtime. If you want to use native Spark algorithms instead of SPSS algorithms, select the Spark runtime. Properties for this node will vary depending on which runtime option you choose.

Watch this short video for an example of C5.0 modeling

Figure 1. Visual classification modeling

## Visual Classification Modeling in Watson Studio Desktop



<https://youtu.be/DPBQASDQlrU>

**Parent topic:** [Modeling](#)

## C&R Tree node

---

The Classification and Regression (C&R) Tree node is a tree-based classification and prediction method. Similar to C5.0, this method uses recursive partitioning to split the training records into segments with similar output field values. The C&R Tree node starts by examining the input fields to find the best split, measured by the reduction in an impurity index that results from the split. The split defines two subgroups, each of which is subsequently split into two more subgroups, and so on, until one of the stopping criteria is triggered. All splits are binary (only two subgroups).

### Pruning

---

C&R Trees give you the option to first grow the tree and then prune based on a cost-complexity algorithm that adjusts the risk estimate based on the number of terminal nodes. This method, which enables the tree to grow large before pruning based on more complex criteria, may result in smaller trees with better cross-validation properties.

Increasing the number of terminal nodes generally reduces the risk for the current (training) data, but the actual risk may be higher when the model is generalized to unseen data. In an extreme case, suppose you have a separate terminal node for each record in the training set. The risk estimate would be 0%, since every record falls into its own node, but the risk of misclassification for unseen (testing) data would almost certainly be greater than 0. The cost-complexity measure attempts to compensate for this.

**Example.** A cable TV company has commissioned a marketing study to determine which customers would buy a subscription to an interactive news service via cable. Using the data from the study, you can create a flow in which the target field is the intent to buy the subscription and the predictor fields include age, sex, education, income category, hours spent watching television each day, and number of children. By applying a C&R Tree node to the flow, you will be able to predict and classify the responses to get the highest response rate for your campaign.

**Requirements.** To train a C&R Tree model, you need one or more **Input** fields and exactly one **Target** field. Target and input fields can be continuous (numeric range) or categorical. Fields set to **Both** or **None** are ignored. Fields used in the model must have their types fully instantiated, and any ordinal (ordered set) fields used in the model must have numeric storage (not string). If necessary, the Reclassify node can be used to convert them.

**Strengths.** C&R Tree models are quite robust in the presence of problems such as missing data and large numbers of fields. They usually do not require long training times to estimate. In addition, C&R Tree models tend to be easier to understand than some other model types--the rules derived from the model have a very straightforward interpretation. Unlike C5.0, C&R Tree can accommodate continuous as well as categorical output fields.

**Parent topic:** [Modeling](#)

## CHAID node

---

CHAID, or Chi-squared Automatic Interaction Detection, is a classification method for building decision trees by using chi-square statistics to identify optimal splits.

CHAID first examines the crosstabulations between each of the input fields and the outcome, and tests for significance using a chi-square independence test. If more than one of these relations is statistically significant, CHAID will select the input field that is the most significant (smallest *p* value). If an input has more than two categories, these are compared, and categories that show no differences in the outcome are collapsed together. This is done by successively joining the pair of categories showing the least significant difference. This category-merging process stops when all remaining categories differ at the specified testing level. For nominal input fields, any categories can be merged; for an ordinal set, only contiguous categories can be merged.

Exhaustive CHAID is a modification of CHAID that does a more thorough job of examining all possible splits for each predictor but takes longer to compute.

**Requirements.** Target and input fields can be continuous or categorical; nodes can be split into two or more subgroups at each level. Any ordinal fields used in the model must have numeric storage (not string). If necessary, the Reclassify node can be used to convert them.

**Strengths.** Unlike the C&R Tree and QUEST nodes, CHAID can generate nonbinary trees, meaning that some splits have more than two branches. It therefore tends to create a wider tree than the binary growing methods. CHAID works for all types of inputs, and it accepts both case weights and frequency variables.

**Parent topic:** [Modeling](#)

## QUEST node

---

QUEST - or Quick, Unbiased, Efficient Statistical Tree - is a binary classification method for building decision trees. A major motivation in its development was to reduce the processing time required for large C&R Tree analyses with either many variables or many cases. A second goal of QUEST was to reduce the tendency found in classification tree methods to favor inputs that allow more splits, that is, continuous (numeric range) input fields or those with many categories.

- QUEST uses a sequence of rules, based on significance tests, to evaluate the input fields at a node. For selection purposes, as little as a single test may need to be performed on each input at a node. Unlike C&R Tree, all splits are not examined, and unlike C&R Tree and CHAID, category combinations are not tested when evaluating an input field for selection. This speeds the analysis.
- Splits are determined by running quadratic discriminant analysis using the selected input on groups formed by the target categories. This method again results in a speed improvement over exhaustive search (C&R Tree) to determine the optimal split.

**Requirements.** Input fields can be continuous (numeric ranges), but the target field must be categorical. All splits are binary. Weight fields cannot be used. Any ordinal (ordered set) fields used in the model must have numeric storage (not string). If necessary, the Reclassify node can be used to convert them.

**Strengths.** Like CHAID, but unlike C&R Tree, QUEST uses statistical tests to decide whether or not an input field is used. It also separates the issues of input selection and splitting, applying different criteria to each. This contrasts with CHAID, in which the statistical test result that determines variable selection also produces the split. Similarly, C&R Tree employs the impurity-change measure to both select the input field and to determine the split.

**Parent topic:** [Modeling](#)

## Tree-AS node

---

The Tree-AS node can be used with data in a distributed environment. With this node, you can choose to build decision trees using either a CHAID or Exhaustive CHAID model.

CHAID, or Chi-squared Automatic Interaction Detection, is a classification method for building decision trees by using chi-square statistics to identify optimal splits.

CHAID first examines the crosstabulations between each of the input fields and the outcome, and tests for significance using a chi-square independence test. If more than one of these relations is statistically significant, CHAID will select the input field that is the most significant (smallest  $p$  value). If an input has more than two categories, these are compared, and categories that show no differences in the outcome are collapsed together. This is done by successively joining the pair of categories showing the least significant difference. This category-merging process stops when all remaining categories differ at the specified testing level. For nominal input fields, any categories can be merged; for an ordinal set, only contiguous categories can be merged.

Exhaustive CHAID is a modification of CHAID that does a more thorough job of examining all possible splits for each predictor but takes longer to compute.

**Requirements.** Target and input fields can be continuous or categorical; nodes can be split into two or more subgroups at each level. Any ordinal fields used in the model must have numeric storage (not string). If necessary, use the Reclassify node to convert them.

**Strengths.** CHAID can generate nonbinary trees, meaning that some splits have more than two branches. It therefore tends to create a wider tree than the binary growing methods. CHAID works for all types of inputs, and it accepts both case weights and frequency variables.

**Parent topic:** [Modeling](#)

## Random Trees node

---

The Random Trees node can be used with data in a distributed environment. In this node, you build an ensemble model that consists of multiple decision trees.

The Random Trees node is a tree-based classification and prediction method that is built on Classification and Regression Tree methodology. As with C&R Tree, this prediction method uses recursive partitioning to split the training records into segments with similar output field values. The node starts by examining the input fields available to it to find the best split, which is measured by the reduction in an impurity index that results from the split. The split defines two subgroups, each of which is then split into two more subgroups, and so on, until one of the stopping criteria is triggered. All splits are binary (only two subgroups).

The Random Trees node uses bootstrap sampling with replacement to generate sample data. The sample data is used to grow a tree model. During tree growth, Random Trees will not sample the data again. Instead, it randomly selects part of the predictors and uses the best one to split a tree node. This process is repeated when splitting each tree node. This is the basic idea of growing a tree in random forest.

Random Trees uses C&R Tree-like trees. Since such trees are binary, each field for splitting results in two branches. For a categorical field with multiple categories, the categories are grouped into two groups based on the inner splitting criterion. Each tree grows to the largest extent possible (there is no pruning). In scoring, Random Trees combines individual tree scores by majority voting (for classification) or average (for regression).

Random Trees differ from C&R Trees as follows:

- Random Trees nodes randomly select a specified number of predictors and uses the best one from the selection to split a node. In contrast, C&R Tree finds the best one from all predictors.
- Each tree in Random Trees grows fully until each leaf node typically contains a single record. So the tree depth could be very large. But standard C&R Tree uses different stopping rules for tree growth, which usually leads to a much shallower tree.

Random Trees adds two features compared to C&R Tree:

- The first feature is *bagging*, where replicas of the training dataset are created by sampling with replacement from the original dataset. This action creates bootstrap samples that are of equal size to the original dataset, after which a *component model* is built on each replica. Together these component models form an ensemble model.
- The second feature is that, at each split of the tree, only a sampling of the input fields is considered for the impurity measure.

Requirements. To train a Random Trees model, you need one or more *Input* fields and one *Target* field. Target and input fields can be continuous (numeric range) or categorical. Fields that are set to either *Both* or *None* are ignored. Fields that are used in the model must have their types fully instantiated, and any ordinal (ordered set) fields that are used in the model must have numeric storage (not string). If necessary, the Reclassify node can be used to convert them.

Strengths. Random Trees models are robust when you are dealing with large data sets and numbers of fields. Due to the use of bagging and field sampling, they are much less prone to overfitting and thus the results that are seen in testing are more likely to be repeated when you use new data.

Note: When first creating a flow, you select which runtime to use. By default, flows use the IBM SPSS Modeler runtime. If you want to use native Spark algorithms instead of SPSS algorithms, select the Spark runtime. Properties for this node will vary depending on which runtime option you choose.

**Parent topic:** [Modeling](#)

## Random Forest node

---

Random Forest® is an advanced implementation of a bagging algorithm with a tree model as the base model.

In random forests, each tree in the ensemble is built from a sample drawn with replacement (for example, a bootstrap sample) from the training set. When splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. Because of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.<sup>1</sup>

The Random Forest node in Watson Studio is implemented in Python. The nodes palette contains this node and other Python nodes.

For more information about random forest algorithms, see <https://scikit-learn.org/stable/modules/ensemble.html#forest>.

<sup>1</sup>L. Breiman, "Random Forests," Machine Learning, 45(1), 5-32, 2001.

**Parent topic:** [Modeling](#)

## Decision List node

---

Decision List models identify subgroups or *segments* that show a higher or lower likelihood of a binary (yes or no) outcome relative to the overall sample.

For example, you might look for customers who are least likely to churn or most likely to say yes to a particular offer or campaign. The Decision List Viewer gives you complete control over the model, enabling you to edit segments, add your own business rules, specify how each segment is scored, and customize the model in a number of other ways to optimize the proportion of hits across all segments. As such, it is particularly well-suited for generating mailing lists or otherwise identifying which records to target for a particular campaign. You can also use multiple *mining tasks* to combine modeling approaches; for example, by identifying high- and low-performing segments within the same model and including or excluding each in the scoring stage as appropriate.

### Segments, rules, and conditions

---

A model consists of a list of segments, each of which is defined by a rule that selects matching records. A given rule may have multiple conditions; for example:

```
RFM_SCORE > 10 and  
MONTHS_CURRENT <= 9
```

Rules are applied in the order listed, with the first matching rule determining the outcome for a given record. Taken independently, rules or conditions may overlap, but the order of rules resolves ambiguity. If no rule matches, the record is assigned to the remainder rule.

### Complete control over scoring

---

The Decision List Viewer enables you to view, modify, and reorganize segments and to choose which to include or exclude for purposes of scoring. For example, you can choose to exclude one group of customers from future offers and include others and immediately see how this affects your overall hit rate. Decision List models return a score of *Yes* for included segments and *\$null\$* for everything else, including the remainder. This direct control over scoring makes Decision List models ideal for generating mailing lists, and they are widely used in customer relationship management, including call center or marketing applications.

### Mining tasks, measures, and selections

---

The modeling process is driven by *mining tasks*. Each mining task effectively initiates a new modeling run and returns a new set of alternative models to choose from. The default task is based on your initial specifications in the Decision List node, but you can define any number of custom tasks. You can also apply tasks iteratively; for example, you can run a high probability search on the entire training set and then run a low probability search on the remainder to weed out low-performing segments.

### Data selections

---

You can define data selections and custom model measures for model building and evaluation. For example, you can specify a data selection in a mining task to tailor the model to a specific region and create a custom measure to evaluate how well that model performs on the whole country. Unlike mining tasks, measures don't change the underlying model but provide another lens to assess how well it performs.

### Adding your business knowledge

---

By fine-tuning or extending the segments identified by the algorithm, the Decision List Viewer enables you to incorporate your business knowledge right into the model. You can edit the segments generated by the model or add additional segments based on rules that you specify. You can then apply the changes and preview the results.



For further insight, a dynamic link with Excel enables you to export your data to Excel, where it can be used to create presentation charts and to calculate custom measures, such as complex profit and ROI, which can be viewed in the Decision List Viewer while you are building the model.

**Example.** The marketing department of a financial institution wants to achieve more profitable results in future campaigns by matching the right offer to each customer. You can use a Decision List model to identify the characteristics of customers most likely to respond favorably based on previous promotions and to generate a mailing list based on the results.

**Requirements.** A single categorical target field with a measurement level of type `Flag` or `Nominal` that indicates the binary outcome you want to predict (yes/no), and at least one input field. When the target field type is `Nominal`, you must manually choose a single value to be treated as a *hit*, or *response*; all the other values are lumped together as *not hit*. An optional frequency field may also be specified. Continuous date/time fields are ignored. Continuous numeric range inputs are automatically binned by the algorithm as specified on the Expert tab in the modeling node. For finer control over binning, add an upstream binning node and use the binned field as input with a measurement level of `Ordinal`.

**Parent topic:** [Modeling](#)

## Time Series node

---

The Time Series node can be used with data in either a local or distributed environment. With this node, you can choose to estimate and build exponential smoothing, univariate Autoregressive Integrated Moving Average (ARIMA), or multivariate ARIMA (or transfer function) models for time series, and produce forecasts based on the time series data.

Exponential smoothing is a method of forecasting that uses weighted values of previous series observations to predict future values. As such, exponential smoothing is not based on a theoretical understanding of the data. It forecasts one point at a time, adjusting its forecasts as new data come in. The technique is useful for forecasting series that exhibit trend, seasonality, or both. You can choose from various exponential smoothing models that differ in their treatment of trend and seasonality.

ARIMA models provide more sophisticated methods for modeling trend and seasonal components than do exponential smoothing models, and, in particular, they allow the added benefit of including independent (predictor) variables in the model. This involves explicitly specifying autoregressive and moving average orders as well as the degree of differencing. You can include predictor variables and define transfer functions for any or all of them, as well as specify automatic detection of outliers or an explicit set of outliers.

**Note:** In practical terms, ARIMA models are most useful if you want to include predictors that might help to explain the behavior of the series that is being forecast, such as the number of catalogs that are mailed or the number of hits to a company web page. Exponential smoothing models describe the behavior of the time series without attempting to understand why it behaves as it does. For example, a series that historically peaks every 12 months will probably continue to do so even if you don't know why.

An Expert Modeler option is also available, which attempts to automatically identify and estimate the best-fitting ARIMA or exponential smoothing model for one or more target variables, thus eliminating the need to identify an appropriate model through trial and error. If in doubt, use the Expert Modeler option.

If predictor variables are specified, the Expert Modeler selects those variables that have a statistically significant relationship with the dependent series for inclusion in ARIMA models. Model variables are transformed where appropriate using differencing and/or a square root or natural log transformation. By default, the Expert Modeler considers all exponential smoothing models and all ARIMA models and picks the best model among them for each target field. You can, however, limit the Expert Modeler only to pick the best of the exponential smoothing models or only to pick the best of the ARIMA models. You can also specify automatic detection of outliers.

Watch this short video for an example of ARIMA time series modeling.

Figure 1. Time series modeling



## Time Series Modeling in Watson Studio Desktop



<https://youtu.be/EfnfKb8D-X8>

**Parent topic:** [Modeling](#)

## GenLin node

---

The generalized linear model expands the general linear model so that the dependent variable is linearly related to the factors and covariates via a specified link function. Moreover, the model allows for the dependent variable to have a non-normal distribution. It covers widely used statistical models, such as linear regression for normally distributed responses, logistic models for binary data, loglinear models for count data, complementary log-log models for interval-censored survival data, plus many other statistical models through its very general model formulation.

Examples. A shipping company can use generalized linear models to fit a Poisson regression to damage counts for several types of ships constructed in different time periods, and the resulting model can help determine which ship types are most prone to damage.

A car insurance company can use generalized linear models to fit a gamma regression to damage claims for cars, and the resulting model can help determine the factors that contribute the most to claim size.

Medical researchers can use generalized linear models to fit a complementary log-log regression to interval-censored survival data to predict the time to recurrence for a medical condition.

Generalized linear models work by building an equation that relates the input field values to the output field values. Once the model is generated, it can be used to estimate values for new data. For each record, a probability of membership is computed for each possible output category. The target category with the highest probability is assigned as the predicted output value for that record.

Requirements. You need one or more input fields and exactly one target field (which can have a measurement level of `Continuous` or `Flag`) with two or more categories. Fields used in the model must have their types fully instantiated.

Strengths. The generalized linear model is extremely flexible, but the process of choosing the model structure is not automated and thus demands a level of familiarity with your data that is not required by "black box" algorithms.

**Parent topic:** [Modeling](#)

## GLMM node

---

This node creates a generalized linear mixed model (GLMM).

Generalized linear mixed models extend the linear model so that:

- The target is linearly related to the factors and covariates via a specified link function
- The target can have a non-normal distribution
- The observations can be correlated

Generalized linear mixed models cover a wide variety of models, from simple linear regression to complex multilevel models for non-normal longitudinal data.

Examples. The district school board can use a generalized linear mixed model to determine whether an experimental teaching method is effective at improving math scores. Students from the same classroom should be correlated since they are taught by the same teacher, and classrooms within the same school may also be correlated, so we can include random effects at school and class levels to account for different sources of variability.

Medical researchers can use a generalized linear mixed model to determine whether a new anticonvulsant drug can reduce a patient's rate of epileptic seizures. Repeated measurements from the same patient are typically positively correlated so a mixed model with some random effects should be appropriate. The target field - the number of seizures - takes positive integer values, so a generalized linear mixed model with a Poisson distribution and log link may be appropriate.

Executives at a cable provider of television, phone, and internet services can use a generalized linear mixed model to learn more about potential customers. Since possible answers have nominal measurement levels, the company analyst uses a generalized logit mixed model with a random intercept to capture correlation between answers to the service usage questions across service types (tv, phone, internet) within a given survey responder's answers.

In the node properties, data structure options allow you to specify the structural relationships between records in your dataset when observations are correlated. If the records in the dataset represent independent observations, you don't need to specify any data structure options.

Subjects. The combination of values of the specified categorical fields should uniquely define subjects within the dataset. For example, a single `Patient ID` field should be sufficient to define subjects in a single hospital, but the combination of `Hospital ID` and `Patient ID` may be necessary if patient identification numbers are not unique across hospitals. In a repeated measures setting, multiple observations are recorded for each subject, so each subject may occupy multiple records in the dataset.

A *subject* is an observational unit that can be considered independent of other subjects. For example, the blood pressure readings from a patient in a medical study can be considered independent of the readings from other patients. Defining subjects becomes particularly important when there are repeated measurements per subject and you want to model the correlation between these observations. For example, you might expect that blood pressure readings from a single patient during consecutive visits to the doctor are correlated.

All of the fields specified as subjects in the node properties are used to define subjects for the residual covariance structure, and provide the list of possible fields for defining subjects for random-effects covariance structures on the Random Effect Block.

Repeated measures. The fields specified here are used to identify repeated observations. For example, a single variable `Week` might identify the 10 weeks of observations in a medical study, or `Month` and `Day` might be used together to identify daily observations over the course of a year.

Define covariance groups by. The categorical fields specified here define independent sets of repeated effects covariance parameters; one for each category defined by the cross-classification of the grouping fields. All subjects have the same covariance type, and subjects within the same covariance grouping will have the same values for the parameters.

Spatial covariance coordinates. The variables in this list specify the coordinates of the repeated observations when one of the spatial covariance types is selected for the repeated covariance type.

Repeated covariance type. This specifies the covariance structure for the residuals. The available structures are:

- First-order autoregressive (AR1)
- Autoregressive moving average (1,1) (ARMA11)
- Compound symmetry
- Diagonal
- Scaled identity
- Spatial: Power
- Spatial: Exponential
- Spatial: Gaussian
- Spatial: Linear
- Spatial: Linear-log
- Spatial: Spherical
- Toeplitz
- Unstructured
- Variance components

**Parent topic:** [Modeling](#)

## GLE node

---

The GLE model identifies the dependent variable that is linearly related to the factors and covariates via a specified link function. Moreover, the model allows for the dependent variable to have a non-normal distribution. It covers widely used statistical models, such as linear regression for normally distributed responses, logistic models for binary data, loglinear models for count data, complementary log-log models for interval-censored survival data, plus many other statistical models through its very general model formulation.

**Examples.** A shipping company can use generalized linear models to fit a Poisson regression to damage counts for several types of ships constructed in different time periods, and the resulting model can help determine which ship types are most prone to damage.

A car insurance company can use generalized linear models to fit a gamma regression to damage claims for cars, and the resulting model can help determine the factors that contribute the most to claim size.

Medical researchers can use generalized linear models to fit a complementary log-log regression to interval-censored survival data to predict the time to recurrence for a medical condition.

GLE models work by building an equation that relates the input field values to the output field values. Once the model is generated, it can be used to estimate values for new data.

For a categorical target, for each record, a probability of membership is computed for each possible output category. The target category with the highest probability is assigned as the predicted output value for that record.

**Requirements.** You need one or more input fields and exactly one target field (which can have a measurement level of *Continuous*, *Categorical*, or *Flag*) with two or more categories. Fields used in the model must have their types fully instantiated.

**Note:** When first creating a flow, you select which runtime to use. By default, flows use the IBM SPSS Modeler runtime. If you want to use native Spark algorithms instead of SPSS algorithms, select the Spark runtime. Properties for this node will vary depending on which runtime option you choose.

## Linear node

---

Linear regression is a common statistical technique for classifying records based on the values of numeric input fields. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values.

**Requirements.** Only numeric fields can be used in a linear regression model. You must have exactly one target field (with the role set to `Target`) and one or more predictors (with the role set to `Input`). Fields with a role of `Both` or `None` are ignored, as are non-numeric fields. (If necessary, non-numeric fields can be recoded using a `Derive` node.)

**Strengths.** Linear regression models are relatively simple and give an easily interpreted mathematical formula for generating predictions. Because linear regression is a long-established statistical procedure, the properties of these models are well understood. Linear models are also typically very fast to train. The Linear node provides methods for automatic field selection in order to eliminate nonsignificant input fields from the equation.

**Tip:** In cases where the target field is categorical rather than a continuous range, such as yes/no or churn/don't churn, logistic regression can be used as an alternative. Logistic regression also provides support for non-numeric inputs, removing the need to recode these fields.

**Note:** When first creating a flow, you select which runtime to use. By default, flows use the IBM SPSS Modeler runtime. If you want to use native Spark algorithms instead of SPSS algorithms, select the Spark runtime. Properties for this node will vary depending on which runtime option you choose.

Parent topic: [Modeling](#)

## Linear-AS node

---

Linear regression is a common statistical technique for classifying records based on the values of numeric input fields. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values.

**Requirements.** Only numeric fields and categorical predictors can be used in a linear regression model. You must have exactly one target field (with the role set to `Target`) and one or more predictors (with the role set to `Input`). Fields with a role of `Both` or `None` are ignored, as are non-numeric fields. (If necessary, non-numeric fields can be recoded using a `Derive` node.)

**Strengths.** Linear regression models are relatively simple and give an easily interpreted mathematical formula for generating predictions. Because linear regression is a long-established statistical procedure, the properties of these models are well understood. Linear models are also typically very fast to train. The Linear node provides methods for automatic field selection in order to eliminate non-significant input fields from the equation.

**Note:** In cases where the target field is categorical rather than a continuous range, such as yes/no or churn/don't churn, logistic regression can be used as an alternative. Logistic regression also provides support for non-numeric inputs, removing the need to recode these fields.

Parent topic: [Modeling](#)

## Regression node

---

Linear regression is a common statistical technique for classifying records based on the values of numeric input fields. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values.

**Requirements.** Only numeric fields can be used in a regression model. You must have exactly one target field (with the role set to `Target`) and one or more predictors (with the role set to `Input`). Fields with a role of `Both` or `None` are ignored, as are non-numeric fields. (If necessary, non-numeric fields can be recoded using a `Derive` node.)

**Strengths.** Regression models are relatively simple and give an easily interpreted mathematical formula for generating predictions. Because regression modeling is a long-established statistical procedure, the properties of these models are well understood. Regression models are also typically very fast to train. The Regression node provides methods for automatic field selection in order to eliminate nonsignificant input fields from the equation.

**Note:** In cases where the target field is categorical rather than a continuous range, such as `yes/no` or `churn/don't churn`, logistic regression can be used as an alternative. Logistic regression also provides support for non-numeric inputs, removing the need to recode these fields. See [Logistic node](#) for more information.

**Parent topic:** [Modeling](#)

## LSVM node

---

With the LSVM node, you can use a linear support vector machine to classify data. LSVM is particularly suited for use with wide datasets--that is, those with a large number of predictor fields. You can use the default settings on the node to produce a basic model relatively quickly, or you can use the build options to experiment with different settings.

The LSVM node is similar to the SVM node, but it is linear and is better at handling a large number of records.

After the model is built, you can:

- Browse the model nugget to display the relative importance of the input fields in building the model.
- Append a Table node to the model nugget to view the model output.

**Example.** A medical researcher has obtained a dataset containing characteristics of a number of human cell samples extracted from patients who were believed to be at risk of developing cancer. Analysis of the original data showed that many of the characteristics differed significantly between benign and malignant samples. The researcher wants to develop an LSVM model that can use the values of similar cell characteristics in samples from other patients to give an early indication of whether their samples might be benign or malignant.

**Parent topic:** [Modeling](#)

## Logistic node

---

Logistic regression, also known as *nominal regression*, is a statistical technique for classifying records based on values of input fields. It is analogous to linear regression but takes a categorical target field instead of a numeric one. Both binomial models (for targets with two discrete categories) and multinomial models (for targets with more than two categories) are supported.

Logistic regression works by building a set of equations that relate the input field values to the probabilities associated with each of the output field categories. Once the model is generated, it can be used to estimate probabilities for new data. For each record, a probability of membership is computed for each possible output category. The target category with the highest probability is assigned as the predicted output value for that record.

**Binomial example.** A telecommunications provider is concerned about the number of customers it is losing to competitors. Using service usage data, you can create a binomial model to predict which customers are liable to transfer to another provider and customize offers so as to retain as many customers as possible. A binomial model is used because the target has two distinct categories (likely to transfer or not).

**Note:** For binomial models only, string fields are limited to eight characters. If necessary, longer strings can be recoded using a Reclassify node or by using the Anonymize node.

**Multinomial example.** A telecommunications provider has segmented its customer base by service usage patterns, categorizing the customers into four groups. Using demographic data to predict group membership, you can create a multinomial model to classify prospective customers into groups and then customize offers for individual customers.

**Requirements.** One or more input fields and exactly one categorical target field with two or more categories. For a binomial model the target must have a measurement level of `Flag`. For a multinomial model the target can have a

measurement level of `Flag`, or of `Nominal` with two or more categories. Fields set to `Both` or `None` are ignored. Fields used in the model must have their types fully instantiated.

**Strengths.** Logistic regression models are often quite accurate. They can handle symbolic and numeric input fields. They can give predicted probabilities for all target categories so that a second-best guess can easily be identified. Logistic models are most effective when group membership is a truly categorical field; if group membership is based on values of a continuous range field (for example, high IQ versus low IQ), you should consider using linear regression to take advantage of the richer information offered by the full range of values. Logistic models can also perform automatic field selection, although other approaches such as tree models or Feature Selection might do this more quickly on large datasets. Finally, since logistic models are well understood by many analysts and data miners, they may be used by some as a baseline against which other modeling techniques can be compared.

When processing large datasets, you can improve performance noticeably by disabling the likelihood-ratio test, an advanced output option.

**Parent topic:** [Modeling](#)

## Neural Net node

---

A *neural network* can approximate a wide range of predictive models with minimal demands on model structure and assumption. The form of the relationships is determined during the learning process. If a linear relationship between the target and predictors is appropriate, the results of the neural network should closely approximate those of a traditional linear model. If a nonlinear relationship is more appropriate, the neural network will automatically approximate the "correct" model structure.

The trade-off for this flexibility is that the neural network is not easily interpretable. If you are trying to explain an underlying process that produces the relationships between the target and predictors, it would be better to use a more traditional statistical model. However, if model interpretability is not important, you can obtain good predictions using a neural network.

**Field requirements.** There must be at least one Target and one Input. Fields set to `Both` or `None` are ignored. There are no measurement level restrictions on targets or predictors (inputs).

The initial weights assigned to neural networks during model building, and therefore the final models produced, depend on the order of the fields in the data. Watson Studio automatically sorts data by field name before presenting it to the neural network for training. This means that explicitly changing the order of the fields in the data upstream will not affect the generated neural net models when a random seed is set in the model builder. However, changing the input field names in a way that changes their sort order will produce different neural network models, even with a random seed set in the model builder. The model quality will not be affected significantly given different sort order of field names.

**Parent topic:** [Modeling](#)

## KNN node

---

Nearest Neighbor Analysis is a method for classifying cases based on their similarity to other cases. In machine learning, it was developed as a way to recognize patterns of data without requiring an exact match to any stored patterns, or cases. Similar cases are near each other and dissimilar cases are distant from each other. Thus, the distance between two cases is a measure of their dissimilarity.

Cases that are near each other are said to be "neighbors." When a new case (holdout) is presented, its distance from each of the cases in the model is computed. The classifications of the most similar cases - the nearest neighbors - are tallied and the new case is placed into the category that contains the greatest number of nearest neighbors.

You can specify the number of nearest neighbors to examine; this value is called *k*. The pictures show how a new case would be classified using two different values of *k*. When *k* = 5, the new case is placed in category 1 because a

majority of the nearest neighbors belong to category 1. However, when  $k = 9$ , the new case is placed in category 0 because a majority of the nearest neighbors belong to category 0.

Nearest neighbor analysis can also be used to compute values for a continuous target. In this situation, the average or median target value of the nearest neighbors is used to obtain the predicted value for the new case.

**Parent topic:** [Modeling](#)

## Cox node

---

Cox Regression builds a predictive model for time-to-event data. The model produces a survival function that predicts the probability that the event of interest has occurred at a given time  $t$  for given values of the predictor variables. The shape of the survival function and the regression coefficients for the predictors are estimated from observed subjects; the model can then be applied to new cases that have measurements for the predictor variables.

Note that information from censored subjects, that is, those that do not experience the event of interest during the time of observation, contributes usefully to the estimation of the model.

**Example.** As part of its efforts to reduce customer churn, a telecommunications company is interested in modeling the "time to churn" in order to determine the factors that are associated with customers who are quick to switch to another service. To this end, a random sample of customers is selected, and their time spent as customers (whether or not they are still active customers) and various demographic fields are pulled from the database.

**Requirements.** You need one or more input fields, exactly one target field, and you must specify a survival time field within the Cox node. The target field should be coded so that the "false" value indicates survival and the "true" value indicates that the event of interest has occurred; it must have a measurement level of `Flag`, with string or integer storage. (Storage can be converted using a `Filler` or `Derive` node if necessary.) Fields set to `Both` or `None` are ignored. Fields used in the model must have their types fully instantiated. The survival time can be any numeric field.

**Note:** On scoring a Cox Regression model, an error is reported if empty strings in categorical variables are used as input to model building. Avoid using empty strings as input.

**Dates & Times.** Date & Time fields cannot be used to directly define the survival time; if you have Date & Time fields, you should use them to create a field containing survival times, based upon the difference between the date of entry into the study and the observation date.

**Kaplan-Meier Analysis.** Cox regression can be performed with no input fields. This is equivalent to a Kaplan-Meier analysis.

**Parent topic:** [Modeling](#)

## PCA/Factor node

---

The PCA/Factor node provides powerful data-reduction techniques to reduce the complexity of your data. Two similar but distinct approaches are provided.

- Principal components analysis (PCA) finds linear combinations of the input fields that do the best job of capturing the variance in the entire set of fields, where the components are orthogonal (perpendicular) to each other. PCA focuses on all variance, including both shared and unique variance.
- Factor analysis attempts to identify underlying concepts, or *factors*, that explain the pattern of correlations within a set of observed fields. Factor analysis focuses on shared variance only. Variance that is unique to specific fields is not considered in estimating the model. Several methods of factor analysis are provided by the Factor/PCA node.

For both approaches, the goal is to find a small number of derived fields that effectively summarize the information in the original set of fields.

Requirements. Only numeric fields can be used in a PCA-Factor model. To estimate a factor analysis or PCA, you need one or more fields with the role set to `Input` fields. Fields with the role set to `Target`, `Both`, or `None` are ignored, as are non-numeric fields.

Strengths. Factor analysis and PCA can effectively reduce the complexity of your data without sacrificing much of the information content. These techniques can help you build more robust models that execute more quickly than would be possible with the raw input fields.

**Parent topic:** [Modeling](#)

## SVM node

---

The SVM node uses a support vector machine to classify data. SVM is particularly suited for use with wide datasets, that is, those with a large number of predictor fields. You can use the default settings on the node to produce a basic model relatively quickly, or you can use the Expert settings to experiment with different types of SVM models.

After the model is built, you can:

- Browse the model nugget to display the relative importance of the input fields in building the model.
- Append a Table node to the model nugget to view the model output.

Example. A medical researcher has obtained a dataset containing characteristics of a number of human cell samples extracted from patients who were believed to be at risk of developing cancer. Analysis of the original data showed that many of the characteristics differed significantly between benign and malignant samples. The researcher wants to develop an SVM model that can use the values of similar cell characteristics in samples from other patients to give an early indication of whether their samples might be benign or malignant.

**Parent topic:** [Modeling](#)

## Feature Selection node

---

Data mining problems may involve hundreds, or even thousands, of fields that can potentially be used as inputs. As a result, a great deal of time and effort may be spent examining which fields or variables to include in the model. To narrow down the choices, the Feature Selection algorithm can be used to identify the fields that are most important for a given analysis. For example, if you are trying to predict patient outcomes based on a number of factors, which factors are the most likely to be important?

Feature selection consists of three steps:

- Screening. Removes unimportant and problematic inputs and records, or cases such as input fields with too many missing values or with too much or too little variation to be useful.
- Ranking. Sorts remaining inputs and assigns ranks based on importance.
- Selecting. Identifies the subset of features to use in subsequent models; for example, by preserving only the most important inputs and filtering or excluding all others.

In an age where many organizations are overloaded with too much data, the benefits of feature selection in simplifying and speeding the modeling process can be substantial. By focusing attention quickly on the fields that matter most, you can reduce the amount of computation required; more easily locate small but important relationships that might otherwise be overlooked; and, ultimately, obtain simpler, more accurate, and more easily explainable models. By reducing the number of fields used in the model, you may find that you can reduce scoring times as well as the amount of data collected in future iterations.

Example. A telephone company has a data warehouse containing information about responses to a special promotion by 5,000 of the company's customers. The data includes a large number of fields containing customers' ages, employment, income, and telephone usage statistics. Three target fields show whether or not the customer responded to each of three offers. The company wants to use this data to help predict which customers are most likely to respond to similar offers in the future.



Requirements. A single target field (one with its role set to `Target`), along with multiple input fields that you want to screen or rank relative to the target. Both target and input fields can have a measurement level of `Continuous` (numeric range) or `Categorical`.

**Parent topic:** [Modeling](#)

## Discriminant node

---

Discriminant analysis builds a predictive model for group membership. The model is composed of a discriminant function (or, for more than two groups, a set of discriminant functions) based on linear combinations of the predictor variables that provide the best discrimination between the groups. The functions are generated from a sample of cases for which group membership is known; the functions can then be applied to new cases that have measurements for the predictor variables but have unknown group membership.

Example. A telecommunications company can use discriminant analysis to classify customers into groups based on usage data. This allows them to score potential customers and target those who are most likely to be in the most valuable groups.

Requirements. You need one or more input fields and exactly one target field. The target must be a categorical field (with a measurement level of `Flag` or `Nominal`) with string or integer storage. (Storage can be converted using a `Filler` or `Derive` node if necessary. ) Fields set to `Both` or `None` are ignored. Fields used in the model must have their types fully instantiated.

Strengths. Discriminant analysis and Logistic Regression are both suitable classification models. However, Discriminant analysis makes more assumptions about the input fields; for example, they are normally distributed and should be continuous, and they give better results if those requirements are met, especially if the sample size is small.

**Parent topic:** [Modeling](#)

## SLRM node

---

Use the Self-Learning Response Model (SLRM) node to build a model that you can continually update, or reestimate, as a dataset grows without having to rebuild the model every time using the complete dataset. For example, this is useful when you have several products and you want to identify which one a customer is most likely to buy if you offer it to them. This model allows you to predict which offers are most appropriate for customers and the probability of the offers being accepted.

Initially, you can build the model using a small dataset with randomly made offers and the responses to those offers. As the dataset grows, the model can be updated and therefore becomes more able to predict the most suitable offers for customers and the probability of their acceptance based upon other input fields such as age, gender, job, and income. You can change the offers available by adding or removing them from within the node, instead of having to change the target field of the dataset.

Before running an SLRM node, you must specify both the target and target response fields in the node properties. The target field must have string storage, not numeric. The target response field must be a flag. The true value of the flag indicates offer acceptance and the false value indicates offer refusal.

Example. A financial institution wants to achieve more profitable results by matching the offer that is most likely to be accepted to each customer. You can use a self-learning model to identify the characteristics of customers most likely to respond favorably based on previous promotions and to update the model in real time based on the latest customer responses.

**Parent topic:** [Modeling](#)

## Spatio-Temporal Prediction (STP) node

Spatio-Temporal Prediction (STP) has many potential applications such as energy management for buildings or facilities, performance analysis and forecasting for mechanical service engineers, or public transport planning. In these applications, measurements such as energy usage are often taken over space and time. Questions that might be relevant to the recording of these measurements include what factors will affect future observations, what can be done to effect a wanted change, or to better manage the system? To address these questions, you can use statistical techniques that can forecast future values at different locations, and can explicitly model adjustable factors to perform what-if analysis.

STP analysis uses data that contains location information, input fields for prediction (predictors), a time field, and a target field. Each location has numerous rows in the data that represent the values of each predictor at each time of measurement. After the data is analyzed, you can use it to predict target values at any location within the shape data that is used in the analysis. It can also forecast when input data for the future points in time are known.

- **Spatio-Temporal Prediction (STP) model nugget**

The Spatio-Temporal Prediction (STP) model nugget provides details of the generated model.

**Parent topic:** [Modeling](#)

## Spatio-Temporal Prediction (STP) model nugget

The Spatio-Temporal Prediction (STP) model nugget provides details of the generated model.

The STP modeling operation creates a number of new fields with the prefix \$STP- as shown in the following table.

Table 1. New fields created by the STP modeling operation

Field name	Description
\$STP- <Time >	The time field created as part of the model build. The time intervals settings in the node properties determine how this field is created. <Time> is the original name of the field selected as the Time field. This field is only created if you converted the original Time field as part of the model build.
\$STP- <Targ et>	This field contains the predictions for the target value. <Target> is the name of the original Target field for the model.
\$STPV AR- <Targ et>	This field contains the VarianceOfPointPrediction values. <Target> is the name of the original Target field for the model.
\$STPL CI- <Targ et>	This field contains the LowerOfPredictionInterval values; that is, the lower bound of confidence. <Target> is the name of the original Target field for the model.
\$STPU CI- <Targ et>	This field contains the UpperOfPredictionInterval values; that is, the upper bound of confidence. <Target> is the name of the original Target field for the model.

**Parent topic:** [Spatio-Temporal Prediction \(STP\) node](#)

## Association Rules node

Association rules associate a particular conclusion (the purchase of a particular product, for example) with a set of conditions (the purchase of several other products, for example).

For example, the rule

```
beer <= cannedveg & frozenmeal (173, 17.0%, 0.84)
```

states that `beer` often occurs when `cannedveg` and `frozenmeal` occur together. The rule is 84% reliable and applies to 17% of the data, or 173 records. Association rule algorithms automatically find the associations that you could find manually using visualization techniques, such as the Web node.

The advantage of association rule algorithms over the more standard decision tree algorithms (C5.0 and C&R Trees) is that associations can exist between *any* of the attributes. A decision tree algorithm will build rules with only a single conclusion, whereas association algorithms attempt to find many rules, each of which may have a different conclusion.

The disadvantage of association algorithms is that they are trying to find patterns within a potentially very large search space and, hence, can require much more time to run than a decision tree algorithm. The algorithms use a *generate and test* method for finding rules--simple rules are generated initially, and these are validated against the dataset. The good rules are stored and all rules, subject to various constraints, are then specialized. *Specialization* is the process of adding conditions to a rule. These new rules are then validated against the data, and the process iteratively stores the best or most interesting rules found. The user usually supplies some limit to the possible number of antecedents to allow in a rule, and various techniques based on information theory or efficient indexing schemes are used to reduce the potentially large search space.

At the end of the processing, a table of the best rules is presented. Unlike a decision tree, this set of association rules cannot be used directly to make predictions in the way that a standard model (such as a decision tree or a neural network) can. This is due to the many different possible conclusions for the rules. Another level of transformation is required to transform the association rules into a classification rule set. Hence, the association rules produced by association algorithms are known as *unrefined models*. Although the user can browse these unrefined models, they cannot be used explicitly as classification models unless the user tells the system to generate a classification model from the unrefined model. This is done from the browser through a Generate menu option.

Two association rule algorithms are supported:

- The Apriori node extracts a set of rules from the data, pulling out the rules with the highest information content. Apriori offers five different methods of selecting rules and uses a sophisticated indexing scheme to process large data sets efficiently. For large problems, Apriori is generally faster to train; it has no arbitrary limit on the number of rules that can be retained, and it can handle rules with up to 32 preconditions. Apriori requires that input and output fields all be categorical but delivers better performance because it is optimized for this type of data.
- The Sequence node discovers association rules in sequential or time-oriented data. A sequence is a list of item sets that tends to occur in a predictable order. For example, a customer who purchases a razor and aftershave lotion may purchase shaving cream the next time he shops. The Sequence node is based on the CARMA association rules algorithm, which uses an efficient two-pass method for finding sequences.

**Parent topic:** [Modeling](#)

## Apriori node

---

The Apriori node discovers association rules in your data.

Association rules are statements of the form:

```
if antecedent(s) then consequent(s)
```

For example, "if a customer purchases a razor and after shave, then that customer will purchase shaving cream with 80% confidence." Apriori extracts a set of rules from the data, pulling out the rules with the highest information content. The Apriori node also discovers association rules in the data. Apriori offers five different methods of selecting rules and uses a sophisticated indexing scheme to efficiently process large data sets.

**Requirements.** To create an Apriori rule set, you need one or more `Input` fields and one or more `Target` fields. Input and output fields (those with the role `Input`, `Target`, or `Both`) must be symbolic. Fields with the role `None` are ignored. Fields types must be fully instantiated before executing the node. Data can be in tabular or transactional format.

**Strengths.** For large problems, Apriori is generally faster to train. It also has no arbitrary limit on the number of rules that can be retained and can handle rules with up to 32 preconditions. Apriori offers five different training methods, allowing more flexibility in matching the data mining method to the problem at hand.

**Parent topic:** [Modeling](#)

## CARMA node

---

The CARMA node uses an association rules discovery algorithm to discover association rules in the data.

Association rules are statements in the form:

**if** *antecedent(s)* **then** *consequent(s)*

For example, if a Web customer purchases a wireless card and a high-end wireless router, the customer is also likely to purchase a wireless music server if offered. The CARMA model extracts a set of rules from the data without requiring you to specify input or target fields. This means that the rules generated can be used for a wider variety of applications. For example, you can use rules generated by this node to find a list of products or services (antecedents) whose consequent is the item that you want to promote this holiday season. Using Watson Studio, you can determine which clients have purchased the antecedent products and construct a marketing campaign designed to promote the consequent product.

**Requirements.** In contrast to Apriori, the CARMA node does not require `Input` or `Target` fields. This is integral to the way the algorithm works and is equivalent to building an Apriori model with all fields set to `Both`. You can constrain which items are listed only as antecedents or consequents by filtering the model after it is built. For example, you can use the model browser to find a list of products or services (antecedents) whose consequent is the item that you want to promote this holiday season.

To create a CARMA rule set, you need to specify an ID field and one or more content fields. The ID field can have any role or measurement level. Fields with the role `None` are ignored. Field types must be fully instantiated before executing the node. Like Apriori, data may be in tabular or transactional format.

**Strengths.** The CARMA node is based on the CARMA association rules algorithm. In contrast to Apriori, the CARMA node offers build settings for rule support (support for both antecedent and consequent) rather than antecedent support. CARMA also allows rules with multiple consequents. Like Apriori, models generated by a CARMA node can be inserted into a data stream to create predictions.

**Parent topic:** [Modeling](#)

## Sequence node

---

The Sequence node discovers patterns in sequential or time-oriented data, in the format `bread -> cheese`. The elements of a sequence are *item sets* that constitute a single transaction.

For example, if a person goes to the store and purchases bread and milk and then a few days later returns to the store and purchases some cheese, that person's buying activity can be represented as two item sets. The first item set contains bread and milk, and the second one contains cheese. A *sequence* is a list of item sets that tend to occur in a predictable order. The Sequence node detects frequent sequences and creates a generated model node that can be used to make predictions.

**Requirements.** To create a Sequence rule set, you need to specify an ID field, an optional time field, and one or more content fields. Note that these settings must be made on the Fields tab of the modeling node; they cannot be read from an upstream Type node. The ID field can have any role or measurement level. If you specify a time field, it can have any role but its storage must be numeric, date, time, or timestamp. If you do not specify a time field, the Sequence node will use an implied timestamp, in effect using row numbers as time values. Content fields can have any measurement level and role, but all content fields must be of the same type. If they are numeric, they must be integer ranges (not real ranges).

**Strengths.** The Sequence node is based on the CARMA association rules algorithm, which uses an efficient two-pass method for finding sequences. In addition, the generated model node created by a Sequence node can be inserted into a data stream to create predictions. The generated model node can also generate supernodes for detecting and counting specific sequences and for making predictions based on specific sequences.

**Parent topic:** [Modeling](#)

## Kohonen node

---

Kohonen networks are a type of neural network that perform clustering, also known as a *knet* or a *self-organizing map*. This type of network can be used to cluster the dataset into distinct groups when you don't know what those groups are at the beginning. Records are grouped so that records within a group or cluster tend to be similar to each other, and records in different groups are dissimilar.

The basic units are *neurons*, and they are organized into two layers: the *input layer* and the *output layer* (also called the *output map*). All of the input neurons are connected to all of the output neurons, and these connections have *strengths*, or *weights*, associated with them. During training, each unit competes with all of the others to "win" each record.

The output map is a two-dimensional grid of neurons, with no connections between the units.

Input data is presented to the input layer, and the values are propagated to the output layer. The output neuron with the strongest response is said to be the *winner* and is the answer for that input.

Initially, all weights are random. When a unit wins a record, its weights (along with those of other nearby units, collectively referred to as a *neighborhood*) are adjusted to better match the pattern of predictor values for that record. All of the input records are shown, and weights are updated accordingly. This process is repeated many times until the changes become very small. As training proceeds, the weights on the grid units are adjusted so that they form a two-dimensional "map" of the clusters (hence the term *self-organizing map*).

When the network is fully trained, records that are similar should be close together on the output map, whereas records that are vastly different will be far apart.

Unlike most learning methods in Watson Studio, Kohonen networks do *not* use a target field. This type of learning, with no target field, is called *unsupervised learning*. Instead of trying to predict an outcome, Kohonen nets try to uncover patterns in the set of input fields. Usually, a Kohonen net will end up with a few units that summarize many observations (*strong* units), and several units that don't really correspond to any of the observations (*weak* units). The strong units (and sometimes other units adjacent to them in the grid) represent probable cluster centers.

Another use of Kohonen networks is in *dimension reduction*. The spatial characteristic of the two-dimensional grid provides a mapping from the *k* original predictors to two derived features that preserve the similarity relationships in the original predictors. In some cases, this can give you the same kind of benefit as factor analysis or PCA.

Note that the method for calculating default size of the output grid is different from older versions of SPSS Modeler. The method will generally produce smaller output layers that are faster to train and generalize better. If you find that you get poor results with the default size, try increasing the size of the output grid on the Expert tab.

**Requirements.** To train a Kohonen net, you need one or more fields with the role set to *Input*. Fields with the role set to *Target*, *Both*, or *None* are ignored.

**Strengths.** You do not need to have data on group membership to build a Kohonen network model. You don't even need to know the number of groups to look for. Kohonen networks start with a large number of units, and as training progresses, the units gravitate toward the natural clusters in the data. You can look at the number of observations captured by each unit in the model nugget to identify the strong units, which can give you a sense of the appropriate number of clusters.

**Parent topic:** [Modeling](#)

## Anomaly node

---

Anomaly detection models are used to identify outliers, or unusual cases, in the data. Unlike other modeling methods that store rules about unusual cases, anomaly detection models store information on what normal behavior looks like. This makes it possible to identify outliers even if they do not conform to any known pattern, and it can be particularly useful in applications, such as fraud detection, where new patterns may constantly be emerging. Anomaly detection is an unsupervised method, which means that it does not require a training dataset containing known cases of fraud to use as a starting point.

While traditional methods of identifying outliers generally look at one or two variables at a time, anomaly detection can examine large numbers of fields to identify clusters or peer groups into which similar records fall. Each record can then be compared to others in its peer group to identify possible anomalies. The further away a case is from the normal center, the more likely it is to be unusual. For example, the algorithm might lump records into three distinct clusters and flag those that fall far from the center of any one cluster.

Each record is assigned an anomaly index, which is the ratio of the group deviation index to its average over the cluster that the case belongs to. The larger the value of this index, the more deviation the case has than the average. Under the usual circumstance, cases with anomaly index values less than 1 or even 1.5 would not be considered as anomalies, because the deviation is just about the same or a bit more than the average. However, cases with an index value greater than 2 could be good anomaly candidates because the deviation is at least twice the average.

Anomaly detection is an exploratory method designed for quick detection of unusual cases or records that should be candidates for further analysis. These should be regarded as *suspected* anomalies, which, on closer examination, may or may not turn out to be real. You may find that a record is perfectly valid but choose to screen it from the data for purposes of model building. Alternatively, if the algorithm repeatedly turns up false anomalies, this may point to an error or artifact in the data collection process.

Note that anomaly detection identifies unusual records or cases through cluster analysis based on the set of fields selected in the model without regard for any specific target (dependent) field and regardless of whether those fields are relevant to the pattern you are trying to predict. For this reason, you may want to use anomaly detection in combination with feature selection or another technique for screening and ranking fields. For example, you can use feature selection to identify the most important fields relative to a specific target and then use anomaly detection to locate the records that are the most unusual with respect to those fields. (An alternative approach would be to build a decision tree model and then examine any misclassified records as potential anomalies. However, this method would be more difficult to replicate or automate on a large scale.)

**Example.** In screening agricultural development grants for possible cases of fraud, anomaly detection can be used to discover deviations from the norm, highlighting those records that are abnormal and worthy of further investigation. You are particularly interested in grant applications that seem to claim too much (or too little) money for the type and size of farm.

**Requirements.** One or more input fields. Note that only fields with a role set to Input using a source or Type node can be used as inputs. Target fields (role set to Target or Both) are ignored.

**Strengths.** By flagging cases that do *not* conform to a known set of rules rather than those that do, Anomaly Detection models can identify unusual cases even when they don't follow previously known patterns. When used in combination with feature selection, anomaly detection makes it possible to screen large amounts of data to identify the records of greatest interest relatively quickly.

## K-Means node

---

The K-Means node provides a method of *cluster analysis*. It can be used to cluster the dataset into distinct groups when you don't know what those groups are at the beginning. Unlike most learning methods in SPSS Modeler, K-Means models do *not* use a target field. This type of learning, with no target field, is called *unsupervised learning*. Instead of trying to predict an outcome, K-Means tries to uncover patterns in the set of input fields. Records are grouped so that records within a group or cluster tend to be similar to each other, but records in different groups are dissimilar.

K-Means works by defining a set of starting cluster centers derived from data. It then assigns each record to the cluster to which it is most similar, based on the record's input field values. After all cases have been assigned, the cluster centers are updated to reflect the new set of records assigned to each cluster. The records are then checked again to see whether they should be reassigned to a different cluster, and the record assignment/cluster iteration process continues until either the maximum number of iterations is reached, or the change between one iteration and the next fails to exceed a specified threshold.

Note: The resulting model depends to a certain extent on the order of the training data. Reordering the data and rebuilding the model may lead to a different final cluster model.

Requirements. To train a K-Means model, you need one or more fields with the role set to `Input`. Fields with the role set to `Output`, `Both`, or `None` are ignored.

Strengths. You do not need to have data on group membership to build a K-Means model. The K-Means model is often the fastest method of clustering for large datasets.

Parent topic: [Modeling](#)

## TwoStep cluster node

---

The TwoStep Cluster node provides a form of *cluster analysis*. It can be used to cluster the dataset into distinct groups when you don't know what those groups are at the beginning. As with Kohonen nodes and K-Means nodes, TwoStep Cluster models do *not* use a target field. Instead of trying to predict an outcome, TwoStep Cluster tries to uncover patterns in the set of input fields. Records are grouped so that records within a group or cluster tend to be similar to each other, but records in different groups are dissimilar.

TwoStep Cluster is a two-step clustering method. The first step makes a single pass through the data, during which it compresses the raw input data into a manageable set of subclusters. The second step uses a hierarchical clustering method to progressively merge the subclusters into larger and larger clusters, without requiring another pass through the data. Hierarchical clustering has the advantage of not requiring the number of clusters to be selected ahead of time. Many hierarchical clustering methods start with individual records as starting clusters and merge them recursively to produce ever larger clusters. Though such approaches often break down with large amounts of data, TwoStep's initial preclustering makes hierarchical clustering fast even for large datasets.

Note: The resulting model depends to a certain extent on the order of the training data. Reordering the data and rebuilding the model may lead to a different final cluster model.

Requirements. To train a TwoStep Cluster model, you need one or more fields with the role set to `Input`. Fields with the role set to `Target`, `Both`, or `None` are ignored. The TwoStep Cluster algorithm does not handle missing values. Records with blanks for any of the input fields will be ignored when building the model.

Strengths. TwoStep Cluster can handle mixed field types and is able to handle large datasets efficiently. It also has the ability to test several cluster solutions and choose the best, so you don't need to know how many clusters to ask for at the outset. TwoStep Cluster can be set to automatically exclude *outliers*, or extremely unusual cases that can contaminate your results.



Important:

Watson Studio has two different versions of the TwoStep Cluster node:

- TwoStep Cluster is the traditional node that runs on the SPSS Modeler Server.
- TwoStep-AS Cluster can run when connected to IBM SPSS Analytic Server.

**Parent topic:** [Modeling](#)

## TwoStep-AS cluster node

---

TwoStep Cluster is an exploratory tool that is designed to reveal natural groupings (or clusters) within a data set that would otherwise not be apparent. The algorithm that is employed by this procedure has several desirable features that differentiate it from traditional clustering techniques.

- Handling of categorical and continuous variables. By assuming variables to be independent, a joint multinomial-normal distribution can be placed on categorical and continuous variables.
- Automatic selection of number of clusters. By comparing the values of a model-choice criterion across different clustering solutions, the procedure can automatically determine the optimal number of clusters.
- Scalability. By constructing a cluster feature (CF) tree that summarizes the records, the TwoStep algorithm can analyze large data files.

For example, retail and consumer product companies regularly apply clustering techniques to information that describes their customers' buying habits, gender, age, income level, and other attributes. These companies tailor their marketing and product development strategies to each consumer group to increase sales and build brand loyalty.

**Parent topic:** [Modeling](#)

## Isotonic-AS node

---

Isotonic Regression belongs to the family of regression algorithms. The Isotonic-AS node in Watson Studio is implemented in Spark.

For details about Isotonic Regression algorithms, see <https://spark.apache.org/docs/2.2.0/mllib-isotonic-regression.html>.<sup>1</sup>

<sup>1</sup> "Regression - RDD-based API." *Apache Spark*. MLLib: Main Guide. Web. 3 Oct 2017.

**Parent topic:** [Modeling](#)

## XGBoost-AS node

---

XGBoost® is an advanced implementation of a gradient boosting algorithm. Boosting algorithms iteratively learn weak classifiers and then add them to a final strong classifier. XGBoost is very flexible and provides many parameters that can be overwhelming to most users, so the XGBoost-AS node in Watson Studio exposes the core features and commonly used parameters. The XGBoost-AS node is implemented in Spark.

For more information about boosting algorithms, see the XGBoost Tutorials available at <http://xgboost.readthedocs.io/en/latest/tutorials/index.html>.<sup>1</sup>

Note that the XGBoost cross-validation function is not supported in Watson Studio. You can use the Partition node for this functionality. Also note that XGBoost in Watson Studio performs one-hot encoding automatically for categorical variables.

Note: On Mac, version 10.12.3 or higher is required for building XGBoost-AS models.



<sup>1</sup> "XGBoost Tutorials." *Scalable and Flexible Gradient Boosting*. Web. © 2015-2016 DMLC.

**Parent topic:** [Modeling](#)

## K-Means-AS node

---

K-Means is one of the most commonly used clustering algorithms. It clusters data points into a predefined number of clusters. The K-Means-AS node in SPSS Modeler is implemented in Spark.

For details about K-Means algorithms, see <https://spark.apache.org/docs/2.2.0/ml-clustering.html>.<sup>1</sup>

Note that the K-Means-AS node performs one-hot encoding automatically for categorical variables.

<sup>1</sup> "Clustering." *Apache Spark*. MLib: Main Guide. Web. 3 Oct 2017.

**Parent topic:** [Modeling](#)

## XGBoost Tree node

---

XGBoost Tree<sup>®</sup> is an advanced implementation of a gradient boosting algorithm with a tree model as the base model. Boosting algorithms iteratively learn weak classifiers and then add them to a final strong classifier. XGBoost Tree is very flexible and provides many parameters that can be overwhelming to most users, so the XGBoost Tree node in Watson Studio exposes the core features and commonly used parameters. The node is implemented in Python.

For more information about boosting algorithms, see the XGBoost Tutorials available at <http://xgboost.readthedocs.io/en/latest/tutorials/index.html>.<sup>1</sup>

Note that the XGBoost cross-validation function is not supported in Watson Studio. You can use the Partition node for this functionality. Also note that XGBoost in Watson Studio performs one-hot encoding automatically for categorical variables.

<sup>1</sup> "XGBoost Tutorials." *Scalable and Flexible Gradient Boosting*. Web. © 2015-2016 DMLC.

**Parent topic:** [Modeling](#)

## XGBoost Linear node

---

XGBoost Linear<sup>®</sup> is an advanced implementation of a gradient boosting algorithm with a linear model as the base model. Boosting algorithms iteratively learn weak classifiers and then add them to a final strong classifier. The XGBoost Linear node in Watson Studio is implemented in Python.

For more information about boosting algorithms, see the XGBoost Tutorials available at <http://xgboost.readthedocs.io/en/latest/tutorials/index.html>.<sup>1</sup>

Note that the XGBoost cross-validation function is not supported in Watson Studio. You can use the Partition node for this functionality. Also note that XGBoost in Watson Studio performs one-hot encoding automatically for categorical variables.

<sup>1</sup> "XGBoost Tutorials." *Scalable and Flexible Gradient Boosting*. Web. © 2015-2016 DMLC.

**Parent topic:** [Modeling](#)

## Gaussian Mixture node

---

A Gaussian Mixture© model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters.

One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.<sup>1</sup>

The Gaussian Mixture node in Watson Studio exposes the core features and commonly used parameters of the Gaussian Mixture library. The node is implemented in Python.

For more information about Gaussian Mixture modeling algorithms and parameters, see the Gaussian Mixture documentation available at <http://scikit-learn.org/stable/modules/mixture.html> and <https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html>.<sup>2</sup>

<sup>1</sup> "User Guide." *Gaussian mixture models*. Web. © 2007 - 2017. scikit-learn developers.

<sup>2</sup> [Scikit-learn: Machine Learning in Python](#), Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

**Parent topic:** [Modeling](#)

## KDE node

---

Kernel Density Estimation (KDE)© uses the Ball Tree or KD Tree algorithms for efficient queries, and walks the line between unsupervised learning, feature engineering, and data modeling.

Neighbor-based approaches such as KDE are some of the most popular and useful density estimation techniques. KDE can be performed in any number of dimensions, though in practice high dimensionality can cause a degradation of performance. The KDE Modeling node and the KDE Simulation node in Watson Studio expose the core features and commonly used parameters of the KDE library. The nodes are implemented in Python.<sup>1</sup>

To use a KDE node, you must set up an upstream Type node. The KDE node will read input values from the Type node (or from the Types of an upstream import node).

The KDE Modeling node is available under the Modeling node palette. The KDE Modeling node generates a model nugget, and the nugget's scored values are kernel density values from the input data.

The KDE Simulation node is available under the Outputs node palette. The KDE Simulation node generates a KDE Gen source node that can create some records that have the same distribution as the input data. In the KDE Gen node properties, you can specify how many records the node will create (default is 1) and generate a random seed.

For more information about KDE, including examples, see the KDE documentation available at <http://scikit-learn.org/stable/modules/density.html#kernel-density-estimation>.<sup>1</sup>

<sup>1</sup> "User Guide." *Kernel Density Estimation*. Web. © 2007-2018, scikit-learn developers.

**Parent topic:** [Modeling](#)

## One-Class SVM node

---

The One-Class SVM© node uses an unsupervised learning algorithm. The node can be used for novelty detection. It will detect the soft boundary of a given set of samples, to then classify new points as belonging to that set or not. This One-Class SVM modeling node is implemented in Python and requires the scikit-learn© Python library.

For details about the scikit-learn library, see <http://contrib.scikit-learn.org/imbalanced-learn/about.html><sup>1</sup>.

The Modeling tab on the palette contains the One-Class SVM node and other Python nodes.

Note: One-Class SVM is used for unsupervised outlier and novelty detection. In most cases, we recommend using a known, "normal" dataset to build the model so the algorithm can set a correct boundary for the given samples. Parameters for the model - such as nu, gamma, and kernel - impact the result significantly. So you may need to experiment with these options until you find the optimal settings for your situation.

<sup>1</sup>Smola, Schölkopf. "A Tutorial on Support Vector Regression." *Statistics and Computing Archive*, vol. 14, no. 3, August 2004, pp. 199-222. (<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.114.4288>)

**Parent topic:** [Modeling](#)

## MultiLayerPerceptron-AS node

---

Multilayer perceptron is a classifier based on the feedforward artificial neural network and consists of multiple layers.

Each layer is fully connected to the next layer in the network. For details about the multilayer perceptron classifier (MLPC), see <https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier>.<sup>1</sup>

The MultiLayerPerceptron-AS node in Watson Studio is implemented in Spark. To use a this node, you must set up an upstream Type node. The MultiLayerPerceptron-AS node will read input values from the Type node (or from the Types of an upstream import node).

<sup>1</sup> "Multilayer perceptron classifier." *Apache Spark*. MLlib: Main Guide. Web. 5 Oct 2018.

**Parent topic:** [Modeling](#)

## HDBSCAN node

---

Hierarchical Density-Based Spatial Clustering (HDBSCAN)© uses unsupervised learning to find clusters, or dense regions, of a data set.

The HDBSCAN node in Watson Studio exposes the core features and commonly used parameters of the HDBSCAN library. The node is implemented in Python, and you can use it to cluster your dataset into distinct groups when you don't know what those groups are at first. Unlike most learning methods in Watson Studio, HDBSCAN models do *not* use a target field. This type of learning, with no target field, is called *unsupervised learning*. Rather than trying to predict an outcome, HDBSCAN tries to uncover patterns in the set of input fields. Records are grouped so that records within a group or cluster tend to be similar to each other, but records in different groups are dissimilar. The HDBSCAN algorithm views clusters as areas of high density separated by areas of low density. Due to this rather generic view, clusters found by HDBSCAN can be any shape, as opposed to k-means which assumes that clusters are convex shaped. Outlier points that lie alone in low-density regions are also marked. HDBSCAN also supports scoring of new samples.<sup>1</sup>

To use the HDBSCAN node, you must set up an upstream Type node. The HDBSCAN node will read input values from the Type node (or from the Types of an upstream import node).

For more information about HDBSCAN clustering algorithms, see the HDBSCAN documentation available at <http://hdbscan.readthedocs.io/en/latest/>.<sup>1</sup>

<sup>1</sup> "User Guide / Tutorial." *The hdbscan Clustering Library*. Web. © 2016, Leland McInnes, John Healy, Steve Astels.

**Parent topic:** [Modeling](#)

## Extension Model node

---

With the Extension Model node, you can run R scripts or Python for Spark scripts to build and score models.

After adding the node to your canvas, double-click the node to open its properties.

## Syntax tab

---

Select your type of syntax - R or Python for Spark. Then enter or paste your custom scripting syntax. When your syntax is ready, you can run the node.

## Model Options tab

---

You can generate the model name automatically based on the target or ID field (or model type in cases where no such field is specified), or type a custom name.

## Console Output tab

---

The Console Output tab contains any output that's received when the R script or Python for Spark script runs (for example, if using an R script, it shows output received from the R console when the R script in the R Syntax field on the Syntax tab is executed). This output might include R or Python error messages or warnings that are produced when the R script or Python script is executed. The output can be used, primarily, to debug the script. The Console Output tab also contains the script from the R Syntax or Python Syntax field.

Every time the Extension Model script runs, the content of the Console Output tab is overwritten with the output received from the R console or Python for Spark. You can't edit the output.

- **Extension model nugget**

The Extension model nugget is generated and placed on your flow canvas after running the Extension Model node, which contains your R script or Python for Spark script that defines the model building and model scoring.

**Parent topic:** [Modeling](#)

## Extension model nugget

---

The Extension model nugget is generated and placed on your flow canvas after running the Extension Model node, which contains your R script or Python for Spark script that defines the model building and model scoring.

By default, the Extension model nugget contains the script that's used for model scoring, options for reading the data, and any output from the R console or Python for Spark. Optionally, the Extension model nugget can also contain various other forms of model output, such as graphs and text output. After the Extension model nugget is generated and added to your flow canvas, an output node can be connected to it. The output node is then used in the usual way within your flow to obtain information about the data and models, and for exporting data in various formats.

## Syntax tab

---

R model scoring syntax. If using R, the R script that's used for model scoring is displayed in this field. By default, this field is enabled but not editable. To edit the Python model scoring script, click Edit.

Python model scoring syntax. If using Python for Spark, the Python script that's used for model scoring is displayed in this field. By default, this field is enabled but not editable. To edit the Python model scoring script, click Edit.

If you click Edit to make the scoring syntax field editable, you can then edit your model scoring script by typing in the scoring syntax field. For example, you might want to edit your model scoring script if you identify an error in your model scoring script after you have run the Extension Model node to generate an Extension model nugget. Any changes you make to the model scoring script in the Extension model nugget will be lost if you regenerate the model by running the Extension Model node again.

## Model Options tab

---

Read Data Options. These options only apply to R, not Python for Spark. With these options, you can specify how missing values, flag fields, and variables with date or datetime formats are handled.

- Read data in batches. If you're processing a large amount of data (that's too big to fit into the R engine's memory, for example), use this option to break the data down into batches that can be sent and processed individually. Specify the maximum number of data records to include in each batch.

For both the Extension Transform node and the Extension model nugget, data passes through the R script (in batch). For this reason, scripts for model scoring and process nodes that run in either a Hadoop or database environment shouldn't include operations that span or combine rows in the data, such as sorting or aggregation. This limitation is imposed to ensure that data can be split up in a Hadoop environment, and during in-database mining. Extension Output and Extension Model nodes don't have this limitation.

- Convert flag fields. Specifies how flag fields are treated. There are two options: Strings to factor, Integers and Reals to double, and Logical values (True, False). If you select Logical values (True, False) the original values of the flag fields are lost. For example, if a field has values `Male` and `Female`, these are changed to `True` and `False`.
- Convert missing values to the R 'not available' value (NA). When selected, any missing values are converted to the R `NA` value. The value `NA` is used by R to identify missing values. Some R functions that you use might have an argument that can control how the function behaves when the data contains `NA`. For example, the function might allow you to choose to automatically exclude records that contain `NA`. If this option isn't selected, any missing values are passed to R unchanged, and might cause errors when your R script runs.
- Convert date/time fields to R classes with special control for time zones When selected, variables with date or datetime formats are converted to R date/time objects. You must select one of the following options:
  - **R POSIXct.** Variables with date or datetime formats are converted to R `POSIXct` objects.
  - **R POSIXlt (list).** Variables with date or datetime formats are converted to R `POSIXlt` objects.Note: The POSIX formats are advanced options. Use these options only if your R script specifies that datetime fields are treated in ways that require these formats. The POSIX formats don't apply to variables with time formats.

The options you select for the Convert flag fields, Convert missing values to the R 'not available' value (NA), and Convert date/time fields to R classes with special control for time zones controls aren't recognized when the Extension model nugget runs against a database. When the node runs against a database, the default values for these controls are used instead:

- Convert flag fields is set to Strings to factor, Integers and Reals to double
- Convert missing values to the R 'not available' value (NA) is selected
- Convert date/time fields to R classes with special control for time zones is not selected

## Console Output tab

---

The Console Output tab contains any output that's received when the R script or Python for Spark script on the Syntax tab runs (for example, if using an R script, it shows output received from the R console when the R script in the R model scoring syntax field on the Syntax tab of the Extension model nugget runs). This output includes any R or Python error messages or warnings that are produced when the R or Python script runs, and any text output from the R console. The output can be used, primarily, to debug the script.

Every time the model scoring script runs, the content of the Console Output tab is overwritten with the output received from the R console or Python for Spark. You can't edit the console output.

**Parent topic:** [Extension Model node](#)

## Text Analytics

---

The Text Analytics nodes offer powerful text analytic capabilities, which use advanced linguistic technologies and Natural Language Processing (NLP) to rapidly process a large variety of unstructured text data and, from this text, extract and organize the key concepts. Text Analytics can also group these concepts into categories.

Around 80% of data held within an organization is in the form of text documents; for example, reports, web pages, e-mails, and call center notes. Text is a key factor in enabling an organization to gain a better understanding of their customers' behavior. A system that incorporates NLP can intelligently extract concepts, including compound phrases. Moreover, knowledge of the underlying language allows classification of terms into related groups, such as products, organizations, or people, using meaning and context. As a result, you can quickly determine the relevance of the information to your needs. These extracted concepts and categories can be combined with existing structured data, such as demographics, and applied to modeling in Watson Studio to yield better and more-focused decisions.

Linguistic systems are knowledge sensitive; the more information contained in their dictionaries, the higher the quality of the results. Text Analytics provides a set of linguistic resources, such as dictionaries for terms and synonyms, libraries, and templates. These nodes further allow you to develop and refine these linguistic resources to your context. Fine-tuning of the linguistic resources is often an iterative process and is necessary for accurate concept retrieval and categorization. Custom templates, libraries, and dictionaries for specific domains, such as CRM and genomics, are also included.

Watch the following short video for an overview of Text Analytics. Also see the [Hotel satisfaction example for Text Analytics](#).

Figure 1. Video introducing Text Analytics



<https://www.youtube.com/watch?v=zCZpT-fAuQ8>

## Applications

---

In general, anyone who routinely needs to review large volumes of documents to identify key elements for further exploration can benefit from using Text Analytics. Examples of some specific applications include:

- Scientific and medical research. Explore secondary research materials, such as patent reports, journal articles, and protocol publications. Identify associations that were previously unknown (such as a doctor associated with a particular product), presenting avenues for further exploration. Minimize the time spent in the drug discovery process. Use as an aid in genomics research.

- Investment research. Review daily analyst reports, news articles, and company press releases to identify key strategy points or market shifts. Trend analysis of such information reveals emerging issues or opportunities for a firm or industry over a period of time.
- Fraud detection. Use in banking and health-care fraud to detect anomalies and discover red flags in large amounts of text.
- Market research. Use in market research endeavors to identify key topics in open-ended survey responses.
- Blog and Web feed analysis. Explore and build models using the key ideas found in news feeds, blogs, etc.
- CRM. Build models using data from all customer touch points, such as e-mail, transactions, and surveys.

## Nodes

---

Along with the many standard SPSS Modeler nodes in Watson Studio, you can also work with text mining nodes to incorporate the power of text analysis into your flows. These nodes are available on the node palette, under Text Analytics:

- The Language Identifier node is a process node that scans source text to determine which human language it's written in and then marks that up in a new field. Primarily designed to be used with large amounts of data, this node is particularly useful when you have more than one language in your data sources and want to process just one language.
- The Text Mining node uses linguistic methods to extract key concepts from the text, allows you to create categories with these concepts and other data, and offers the ability to identify relationships and associations between concepts based on known patterns (called text link analysis). You can use this node to explore the text data contents or to produce either a concept model or category model. The concepts and categories can be combined with existing structured data, such as demographics, and applied to modeling.
- The Text Link Analysis node extracts concepts and also identifies relationships between concepts based on known patterns within the text. You can use pattern extraction to discover relationships between your concepts, as well as any opinions or qualifiers attached to these concepts. The Text Link Analysis (TLA) node offers a more direct way to identify and extract patterns from your text and then add the pattern results to the dataset in the flow. But you can also perform TLA using an interactive workbench session in the Text Mining modeling node.
- **About text mining**  
Today, an increasing amount of information is being held in unstructured and semistructured formats, such as customer e-mails, call center notes, open-ended survey responses, news feeds, web forms, etc. This abundance of information poses a problem to many organizations that ask themselves, "How can we collect, explore, and leverage this information?"
- **Reading in source text**  
You can use the Language Identifier node to identify the natural language of a text field within your source data. The output of this node is a derived field that contains the detected language code.
- **Mining for text links**  
The Text Link Analysis (TLA) node adds pattern-matching technology to text mining's concept extraction in order to identify relationships between the concepts in the text data based on known patterns. These relationships can describe how a customer feels about a product, which companies are doing business together, or even the relationships between genes or pharmaceutical agents.
- **Mining for concepts and categories**  
The Text Mining node uses linguistic and frequency techniques to extract key concepts from the text and create categories with these concepts and other data. Use the node to explore the text data contents or to produce either a concept model nugget or category model nugget.

**Parent topic:** [Nodes palette](#)

## About text mining

---

Today, an increasing amount of information is being held in unstructured and semistructured formats, such as customer e-mails, call center notes, open-ended survey responses, news feeds, web forms, etc. This abundance of

information poses a problem to many organizations that ask themselves, "How can we collect, explore, and leverage this information?"

*Text mining* is the process of analyzing collections of textual materials in order to capture key concepts and themes and uncover hidden relationships and trends without requiring that you know the precise words or terms that authors have used to express those concepts. Although they are quite different, text mining is sometimes confused with information retrieval. While the accurate retrieval and storage of information is an enormous challenge, the extraction and management of quality content, terminology, and relationships contained within the information are crucial and critical processes.

## Text mining and data mining

---

For each article of text, linguistic-based text mining returns an index of concepts, as well as information about those concepts. This distilled, structured information can be combined with other data sources to address questions such as:

- Which concepts occur together?
- What else are they linked to?
- What higher level categories can be made from extracted information?
- What do the concepts or categories predict?
- How do the concepts or categories predict behavior?

Combining text mining with data mining offers greater insight than is available from either structured or unstructured data alone. This process typically includes the following steps:

1. Identify the text to be mined. Prepare the text for mining. If the text exists in multiple files, save the files to a single location. For databases, determine the field containing the text.
2. Mine the text and extract structured data. Apply the text mining algorithms to the source text.
3. Build concept and category models. Identify the key concepts and/or create categories. The number of concepts returned from the unstructured data is typically very large. Identify the best concepts and categories for scoring.
4. Analyze the structured data. Employ traditional data mining techniques, such as clustering, classification, and predictive modeling, to discover relationships between the concepts. Merge the extracted concepts with other structured data to predict future behavior based on the concepts.

## Text analysis and categorization

---

Text analysis, a form of qualitative analysis, is the extraction of useful information from text so that the key ideas or concepts contained within this text can be grouped into an appropriate number of categories. Text analysis can be performed on all types and lengths of text, although the approach to the analysis will vary somewhat.

Shorter records or documents are most easily categorized, since they are not as complex and usually contain fewer ambiguous words and responses. For example, with short, open-ended survey questions, if we ask people to name their three favorite vacation activities, we might expect to see many short answers, such as *going to the beach*, *visiting national parks*, or *doing nothing*. Longer, open-ended responses, on the other hand, can be quite complex and very lengthy, especially if respondents are educated, motivated, and have enough time to complete a questionnaire. If we ask people to tell us about their political beliefs in a survey or have a blog feed about politics, we might expect some lengthy comments about all sorts of issues and positions.

The ability to extract key concepts and create insightful categories from these longer text sources in a very short period of time is a key advantage of using Text Analytics. This advantage is obtained through the combination of automated linguistic and statistical techniques to yield the most reliable results for each stage of the text analysis process.

## Linguistic processing and NLP

---

The primary problem with the management of all of this unstructured text data is that there are no standard rules for writing text so that a computer can understand it. The language, and consequently the meaning, varies for every



document and every piece of text. The only way to accurately retrieve and organize such unstructured data is to analyze the language and thus uncover its meaning. There are several different automated approaches to the extraction of concepts from unstructured information. These approaches can be broken down into two kinds, linguistic and nonlinguistic.

Some organizations have tried to employ automated nonlinguistic solutions based on statistics and neural networks. Using computer technology, these solutions can scan and categorize key concepts more quickly than human readers can. Unfortunately, the accuracy of such solutions is fairly low. Most statistics-based systems simply count the number of times words occur and calculate their statistical proximity to related concepts. They produce many irrelevant results, or noise, and miss results they should have found, referred to as silence.

To compensate for their limited accuracy, some solutions incorporate complex nonlinguistic rules that help to distinguish between relevant and irrelevant results. This is referred to as *rule-based text mining*.

*Linguistics-based text mining*, on the other hand, applies the principles of natural language processing (NLP) - the computer-assisted analysis of human language - to the analysis of words, phrases, and syntax, or structure, of text. A system that incorporates NLP can intelligently extract concepts, including compound phrases. Moreover, knowledge of the underlying language allows classification of concepts into related groups, such as products, organizations, or people, using meaning and context.

Linguistics-based text mining finds meaning in text much as people do: by recognizing a variety of word forms as having similar meanings and by analyzing sentence structure to provide a framework for understanding the text. This approach offers the speed and cost-effectiveness of statistics-based systems, but it offers a far higher degree of accuracy while requiring far less human intervention.

To illustrate the difference between statistics-based and linguistics-based approaches during the extraction process, consider how each would respond to a query about `reproduction of documents`. Both statistics-based and linguistics-based solutions would have to expand the word `reproduction` to include synonyms, such as `copy` and `duplication`. Otherwise, relevant information will be overlooked. But if a statistics-based solution attempts to do this type of synonymy - searching for other terms with the same meaning - it is likely to include the term `birth` as well, generating a number of irrelevant results. The understanding of language cuts through the ambiguity of text, making linguistics-based text mining, by definition, the more reliable approach.

Understanding how the extraction process works can help you make key decisions when fine-tuning your linguistic resources (libraries, types, synonyms, and more). Steps in the extraction process include:

- Converting source data to a standard format
- Identifying candidate terms
- Identifying equivalence classes and integration of synonyms
- Assigning a type
- Indexing and, when requested, pattern matching with a secondary analyzer

#### Step 1. Converting source data to a standard format

In this first step, the data you import is converted to a uniform format that can be used for further analysis. This conversion is performed internally and does not change your original data.

#### Step 2. Identifying candidate terms

It is important to understand the role of linguistic resources in the identification of candidate terms during linguistic extraction. Linguistic resources are used every time an extraction is run. They exist in the form of templates, libraries, and compiled resources. Libraries include lists of words, relationships, and other information used to specify or tune the extraction. The compiled resources cannot be viewed or edited. However, the remaining resources can be edited in the Template Editor or, if you're in an interactive workbench session, in the Resource Editor.

Compiled resources are core, internal components of the extraction engine within Text Analytics. These resources include a general dictionary containing a list of base forms with a part-of-speech code (noun, verb, adjective, and so on).

In addition to those compiled resources, several libraries are delivered with the product and can be used to complement the types and concept definitions in the compiled resources, as well as to offer synonyms. These libraries - and any custom ones you create - are made up of several dictionaries. These include type dictionaries, synonym dictionaries, and exclude dictionaries.

Once the data have been imported and converted, the extraction engine will begin identifying candidate terms for extraction. Candidate terms are words or groups of words that are used to identify concepts in the text. During the processing of the text, single words (*uniterms*) and compound words (*multiterms*) are identified using part-of-speech pattern extractors. Then, candidate sentiment keywords are identified using sentiment text link analysis.

Note: The terms in the aforementioned compiled general dictionary represent a list of all of the words that are likely to be uninteresting or linguistically ambiguous as uniterms. These words are excluded from extraction when you are identifying the uniterms. However, they are reevaluated when you are determining parts of speech or looking at longer candidate compound words (multiterms).

### Step 3. Identifying equivalence classes and integration of synonyms

After candidate uniterms and multiterms are identified, the software uses a normalization dictionary to identify equivalence classes. An equivalence class is a base form of a phrase or a single form of two variants of the same phrase. The purpose of assigning phrases to equivalence classes is to ensure that, for example, *side effect* and 副作用 are not treated as separate concepts. To determine which concept to use for the equivalence class - that is, whether *side effect* or 副作用 is used as the lead term - the extraction engine applies the following rules in the order listed:

- The user-specified form in a library.
- The most frequent form, as defined by precompiled resources.

### Step 4. Assigning type

Next, types are assigned to extracted concepts. A type is a semantic grouping of concepts. Both compiled resources and the libraries are used in this step. Types include such things as higher-level concepts, positive and negative words, first names, places, organizations, and more.

Linguistic systems are knowledge sensitive - the more information contained in their dictionaries, the higher the quality of the results. Modification of the dictionary content, such as synonym definitions, can simplify the resulting information. This is often an iterative process and is necessary for accurate concept retrieval. NLP is a core element of Text Analytics.

- **How extraction works**  
During the extraction of key concepts and ideas from your responses, Text Analytics relies on linguistics-based text analysis. This approach offers the speed and cost effectiveness of statistics-based systems. But it offers a far higher degree of accuracy, while requiring far less human intervention. Linguistics-based text analysis is based on the field of study known as natural language processing, also known as computational linguistics.
- **How categorization works**  
When creating category models in Text Analytics, there are several different techniques you can choose from to create categories. Because every dataset is unique, the number of techniques and the order in which you apply them may change.

**Parent topic:** [Text Analytics](#)

## How extraction works

---

During the extraction of key concepts and ideas from your responses, Text Analytics relies on linguistics-based text analysis. This approach offers the speed and cost effectiveness of statistics-based systems. But it offers a far higher degree of accuracy, while requiring far less human intervention. Linguistics-based text analysis is based on the field of study known as natural language processing, also known as computational linguistics.

Understanding how the extraction process works can help you make key decisions when fine-tuning your linguistic resources (libraries, types, synonyms, and more). Steps in the extraction process include:

- Converting source data to a standard format
- Identifying candidate terms
- Identifying equivalence classes and integration of synonyms
- Assigning a type
- Indexing
- Matching patterns and events extraction

## Step 1. Converting source data to a standard format

---

In this first step, the data you import is converted to a uniform format that can be used for further analysis. This conversion is performed internally and does not change your original data.

## Step 2. Identifying candidate terms

---

It is important to understand the role of linguistic resources in the identification of candidate terms during linguistic extraction. Linguistic resources are used every time an extraction is run. They exist in the form of templates, libraries, and compiled resources. Libraries include lists of words, relationships, and other information used to specify or tune the extraction. The compiled resources cannot be viewed or edited. However, the remaining resources (templates) can be edited in the Template Editor or, if you're in an interactive workbench session, in the Resource Editor.

Compiled resources are core, internal components of the extraction engine. These resources include a general dictionary containing a list of base forms with a part-of-speech code (noun, verb, adjective, adverb, participle, coordinator, determiner, or preposition). The resources also include reserved, built-in types used to assign many extracted terms to the following types, `<Location>`, `<Organization>`, or `<Person>`.

In addition to those compiled resources, several libraries are delivered with the product and can be used to complement the types and concept definitions in the compiled resources, as well as to offer other types and synonyms. These libraries - and any custom ones you create - are made up of several dictionaries. These include type dictionaries, substitution dictionaries (synonyms and optional elements), and exclude dictionaries.

Once the data have been imported and converted, the extraction engine will begin identifying candidate terms for extraction. Candidate terms are words or groups of words that are used to identify concepts in the text. During the processing of the text, single words (*uniterms*) that are not in the compiled resources are considered as candidate term extractions. Candidate compound words (*multiterms*) are identified using part-of-speech pattern extractors. For example, the multiterm `sports car`, which follows the "adjective noun" part-of-speech pattern, has two components. The multiterm `fast sports car`, which follows the "adjective adjective noun" part-of-speech pattern, has three components.

Note: The terms in the aforementioned compiled general dictionary represent a list of all of the words that are likely to be uninteresting or linguistically ambiguous as uniterms. These words are excluded from extraction when you are identifying the uniterms. However, they are reevaluated when you are determining parts of speech or looking at longer candidate compound words (multiterms).

Finally, a special algorithm is used to handle uppercase letter strings, such as job titles, so that these special patterns can be extracted.

## Step 3. Identifying equivalence classes and integration of synonyms

---

After candidate uniterms and multiterms are identified, the software uses a set of algorithms to compare them and identify equivalence classes. An equivalence class is a base form of a phrase or a single form of two variants of the same phrase. The purpose of assigning phrases to equivalence classes is to ensure that, for example, `president of the company` and `company president` are not treated as separate concepts. To determine which concept to use for the equivalence class - that is, whether `president of the`

company or company president is used as the lead term, the extraction engine applies the following rules in the order listed:

- The user-specified form in a library.
- The most frequent form in the full body of text.
- The shortest form in the full body of text (which usually corresponds to the base form).

## Step 4. Assigning type

---

Next, types are assigned to extracted concepts. A type is a semantic grouping of concepts. Both compiled resources and the libraries are used in this step. Types include such things as higher-level concepts, positive and negative words, first names, places, organizations, and more. Additional types can be defined by the user.

## Step 5. Indexing

---

The entire set of records or documents is indexed by establishing a pointer between a text position and the representative term for each equivalence class. This assumes that all of the inflected form instances of a candidate concept are indexed as a candidate base form. The global frequency is calculated for each base form.

## Step 6. Matching patterns and events extraction

---

Text Analytics can discover not only types and concepts but also relationships among them. Several algorithms and libraries are available with this tool and provide the ability to extract relationship patterns between types and concepts. They are particularly useful when attempting to discover specific opinions (for example, product reactions) or the relational links between people or objects (for example, links between political groups or genomes).

**Parent topic:** [About text mining](#)

## How categorization works

---

When creating category models in Text Analytics, there are several different techniques you can choose from to create categories. Because every dataset is unique, the number of techniques and the order in which you apply them may change.

Since your interpretation of the results may be different from someone else's, you may need to experiment with the different techniques to see which one produces the best results for your text data. In Text Analytics, you can create category models in a workbench session in which you can explore and fine-tune your categories further.

In this documentation, category building refers to the generation of category definitions and classification through the use of one or more built-in techniques, and categorization refers to the scoring, or labeling, process whereby unique identifiers (name/ID/value) are assigned to the category definitions for each record or document.

During category building, the concepts and types that were extracted are used as the building blocks for your categories. When you build categories, the records or documents are automatically assigned to categories if they contain text that matches an element of a category's definition.

Text Analytics offers you several automated category building techniques to help you categorize your documents or records quickly.

## Grouping techniques

---

Each of the techniques available is well suited to certain types of data and situations, but often it is helpful to combine techniques in the same analysis to capture the full range of documents records. You may see a concept in multiple categories or find redundant categories.

**Semantic Network.** This technique begins by identifying the possible senses of each concept from its extensive index of word relationships and then creates categories by grouping related concepts. This technique is best when the

concepts are known to the semantic network and are not too ambiguous. It is less helpful when text contains specialized terminology or jargon unknown to the network. In one example, the concept `granny smith apple` could be grouped with `gala apple` and `winesap apple` since they are siblings of the `granny smith`. In another example, the concept `animal` might be grouped with `cat` and `kangaroo` since they are hyponyms of `animal`. This technique is available for English text only.

**Concept Inclusion.** This technique builds categories by grouping multiterm concepts (compound words) based on whether they contain words that are subsets or supersets of a word in the other. For example, the concept `seat` would be grouped with `safety seat`, `seat belt`, and `seat belt buckle`.

**Parent topic:** [About text mining](#)

## Reading in source text

---

You can use the Language Identifier node to identify the natural language of a text field within your source data. The output of this node is a derived field that contains the detected language code.

Data for text mining can be in any of the standard formats that are used by SPSS Modeler flows, including databases or other "rectangular" formats that represent data in rows and columns.

- To read in text from any of the standard data formats used by SPSS Modeler flows, such as a database with one or more text fields for customer comments, you can use an Import node.
- When you're processing large amounts of data, which might include text in several different languages, use the Language Identifier node to identify the language used in a specific field.

**Parent topic:** [Text Analytics](#)

## Mining for text links

---

The Text Link Analysis (TLA) node adds pattern-matching technology to text mining's concept extraction in order to identify relationships between the concepts in the text data based on known patterns. These relationships can describe how a customer feels about a product, which companies are doing business together, or even the relationships between genes or pharmaceutical agents.

For example, extracting your competitor's product name may not be interesting enough to you. Using this node, you could also learn how people feel about this product, if such opinions exist in the data. The relationships and associations are identified and extracted by matching known patterns to your text data.

You can use the TLA pattern rules inside certain resource templates shipped with Text Analytics or create/edit your own. Pattern rules are made up of macros, word lists, and word gaps to form a Boolean query, or rule, that is compared to your input text. Whenever a TLA pattern rule matches text, this text can be extracted as a TLA result and restructured as output data.

The Text Link Analysis node offers a more direct way to identify and extract TLA pattern results from your text and then add the results to the dataset in the flow. But the Text Link Analysis node is not the only way in which you can perform text link analysis. You can also use an interactive workbench session in the Text Mining modeling node.

In the interactive workbench, you can explore the TLA pattern results and use them as category descriptors and/or to learn more about the results using drill-down and graphs. In fact, using the Text Mining node to extract TLA results is a great way to explore and fine-tune templates to your data for later use directly in the TLA node.

The output can be represented in up to 6 slots, or parts.

You can find this node under the Text Analytics section of the node palette.

**Requirements.** The Text Link Analysis node accepts text data read into a field using an Input node.

Strengths. The Text Link Analysis node goes beyond basic concept extraction to provide information about the relationships *between* concepts, as well as related opinions or qualifiers that may be revealed in the data.

- **Expert options**

With the Text Link Analysis (TLA) node, the extraction of text link analysis pattern results is automatically enabled. In the node's properties, the expert options include certain additional parameters that impact how text is extracted and handled. The expert parameters control the basic behavior, as well as a few advanced behaviors, of the extraction process. There are also a number of linguistic resources and options that also impact the extraction results, which are controlled by the resource template you select.

- **TLA node output**

After running a Text Link Analysis node, the data is restructured. It's important to understand the way text mining restructures your data.

**Parent topic:** [Text Analytics](#)

## Expert options

---

With the Text Link Analysis (TLA) node, the extraction of text link analysis pattern results is automatically enabled. In the node's properties, the expert options include certain additional parameters that impact how text is extracted and handled. The expert parameters control the basic behavior, as well as a few advanced behaviors, of the extraction process. There are also a number of linguistic resources and options that also impact the extraction results, which are controlled by the resource template you select.

Limit extraction to concepts with a global frequency of at least [n]. This option specifies the minimum number of times a word or phrase must occur in the text in order for it to be extracted. In this way, a value of 5 limits the extraction to those words or phrases that occur at least five times in the entire set of records or documents.

In some cases, changing this limit can make a big difference in the resulting extraction results, and consequently, your categories. Let's say that you're working with some restaurant data and you don't increase the limit above 1 for this option. In this case, you might find `pizza (1)`, `thin pizza (2)`, `spinach pizza (2)`, and `favorite pizza (2)` in your extraction results. However, if you were to limit the extraction to a global frequency of 5 or more and re-extract, you would no longer get three of these concepts. Instead you would get `pizza (7)`, since `pizza` is the simplest form and this word already existed as a possible candidate. And depending on the rest of your text, you might actually have a frequency of more than seven, depending on whether there are still other phrases with `pizza` in the text. Additionally, if `spinach pizza` was already a category descriptor, you might need to add `pizza` as a descriptor instead to capture all of the records. For this reason, change this limit with care whenever categories have already been created.

Note that this is an extraction-only feature; if your template contains terms (they usually do), and a term for the template is found in the text, then the term will be indexed regardless of its frequency.

For example, suppose you use a Basic Resources template that includes "los angeles" under the `<Location>` type in the Core library; if your document contains Los Angeles only once, then Los Angeles will be part of the list of concepts. To prevent this, you'll need to set a filter to display concepts occurring at least the same number of times as the value entered in the Limit extraction to concepts with a global frequency of at least [n] field.

Accommodate punctuation errors. This option temporarily normalizes text containing punctuation errors (for example, improper usage) during extraction to improve the extractability of concepts. This option is extremely useful when text is short and of poor quality (as, for example, in open-ended survey responses, e-mail, and CRM data), or when the text contains many abbreviations.

Accommodate spelling for a minimum word character length of [n]. This option applies a fuzzy grouping technique that helps group commonly misspelled words or closely spelled words under one concept. The fuzzy grouping algorithm temporarily strips all vowels (except the first one) and strips double/triple consonants from extracted words and then compares them to see if they're the same so that `modeling` and `modelling` would be grouped together.

However, if each term is assigned to a different type, excluding the `<Unknown>` type, the fuzzy grouping technique won't be applied.

You can also define the minimum number of *root* characters required before fuzzy grouping is used. The number of root characters in a term is calculated by totaling all of the characters and subtracting any characters that form inflection suffixes and, in the case of compound-word terms, determiners and prepositions. For example, the term `exercises` is counted as 8 root characters in the form "exercise," since the letter `s` at the end of the word is an inflection (plural form). Similarly, `apple sauce` counts as 10 root characters ("apple sauce") and `manufacturing of cars` counts as 16 root characters ("manufacturing car"). This method of counting is only used to check whether the fuzzy grouping should be applied but doesn't influence how the words are matched.

Note: If you find that certain words are later grouped incorrectly, you can exclude word pairs from this technique by explicitly declaring them in the Fuzzy Grouping: Exceptions section under the Advanced Resources properties.

Extract uniterms. This option extracts single words (uniterms) as long as the word isn't already part of a compound word and if it's either a noun or an unrecognized part of speech.

Extract nonlinguistic entities. This option extracts nonlinguistic entities, such as phone numbers, social security numbers, times, dates, currencies, digits, percentages, e-mail addresses, and HTTP addresses. You can include or exclude certain types of nonlinguistic entities in the Nonlinguistic Entities: Configuration section under the Advanced Resources properties. By disabling any unnecessary entities, the extraction engine won't waste processing time.

Uppercase algorithm. This option extracts simple and compound terms that aren't in the built-in dictionaries as long as the first letter of the term is in uppercase. This option offers a good way to extract most proper nouns.

Group partial and full person names together when possible. This option groups names that appear differently in the text together. This feature is helpful since names are often referred to in their full form at the beginning of the text and then only by a shorter version. This option attempts to match any uniterm with the `<Unknown>` type to the last word of any of the compound terms that is typed as `<Person>`. For example, if `doe` is found and initially typed as `<Unknown>`, the extraction engine checks to see if any compound terms in the `<Person>` type include `doe` as the last word, such as `john doe`. This option doesn't apply to first names since most are never extracted as uniterms.

Maximum nonfunction word permutation. This option specifies the maximum number of nonfunction words that can be present when applying the permutation technique. This permutation technique groups similar phrases that differ from each other only by the nonfunction words (for example, `of` and `the`) contained, regardless of inflection. For example, let's say that you set this value to - at most - two words, and both `company officials` and `officials of the company` were extracted. In this case, both extracted terms would be grouped together in the final concept list since both terms are deemed to be the same when `of the` is ignored.

Use derivation when grouping multiterms. When processing Big Data, select this option to group multiterms by using derivation rules.

**Parent topic:** [Mining for text links](#)

## TLA node output

---

After running a Text Link Analysis node, the data is restructured. It's important to understand the way text mining restructures your data.

If you desire a different structure for data mining, you can use nodes on the Field Operations palette to accomplish this. For example, if you were working with data in which each row represented a text record, then one row is created for each pattern uncovered in the source text data. For each row in the output, there are 15 fields:

- Six fields (Concept#, such as Concept1, Concept2, ..., and Concept6) represent any concepts found in the pattern match
- Six fields (Type#, such as Type1, Type2, ..., and Type6) represent the type for each concept
- Rule Name represents the name of the text link rule used to match the text and produce the output



- A field using the name of the ID field you specified in the node and representing the record or document ID as it was in the input data
- Matched Text represents the portion of the text data in the original record or document that was matched to the TLA pattern

**Parent topic:** [Mining for text links](#)

## Mining for concepts and categories

---

The Text Mining node uses linguistic and frequency techniques to extract key concepts from the text and create categories with these concepts and other data. Use the node to explore the text data contents or to produce either a concept model nugget or category model nugget.

When you run this modeling node, an internal linguistic extraction engine extracts and organizes the concepts, patterns, and/or categories using natural language processing methods.

You can run the Text Mining node and automatically produce a concept or category model nugget using the Generate directly option. Alternatively, you can use a more hands-on, exploratory approach using the Build interactively mode in which not only can you extract concepts, create categories, and refine your linguistic resources, but also perform text link analysis and explore clusters.

**Requirements.** Text Mining modeling nodes accept text data from Import nodes.

Use the Text Mining node to generate one of two text mining model nuggets:

- *Concept model nuggets* uncover and extract salient concepts from your structured or unstructured text data.
- *Category model nuggets* score and assign documents and records to categories, which are made up of the extracted concepts (and patterns).

The extracted concepts and patterns and the categories from your model nuggets can all be combined with existing structured data, such as demographics, to yield better and more focused decisions. For example, if customers frequently list login issues as the primary impediment to completing online account management tasks, you might want to incorporate "login issues" into your models.

In Text Analytics, we often refer to extracted concepts and categories. It is important to understand the meaning of concepts and categories since they can help you make more informed decisions during your exploratory work and model building.

## Concepts and concept model nuggets

---

During the extraction process, text data is scanned and analyzed to identify interesting or relevant single words, such as `election` or `peace`, and word phrases such as `presidential election`, `election of the president`, or `peace treaties`. These words and phrases are collectively referred to as *terms*. Using the linguistic resources, the relevant terms are extracted, and similar terms are grouped together under a lead term called a *concept*.

In this way, a concept could represent multiple underlying terms depending on your text and the set of linguistic resources you are using. For example, let's say we have an employee satisfaction survey and the concept `salary` was extracted. Let's also say that when you looked at the records associated with `salary`, you noticed that `salary` isn't always present in the text but instead certain records contained something similar, such as the terms `wage`, `wages`, and `salaries`. These terms are grouped under `salary` since the extraction engine deemed them as similar or determined they were synonyms based on processing rules or linguistic resources. In this case, any documents or records containing any of those terms would be treated as if they contained the word `salary`.

If you want to see what terms are grouped under a concept, you can explore the concept within an interactive workbench or look at which synonyms are shown in the concept model.



A *concept model nugget* contains a set of concepts you can use to identify records or documents that also contain the concept (including any of its synonyms or grouped terms). A concept model can be used in two ways. The first would be to explore and analyze the concepts that were discovered in the original source text or to quickly identify documents of interest. The second would be to apply this model to new text records or documents to quickly identify the same key concepts in the new documents/records, such as the real-time discovery of key concepts in scratch-pad data from a call center.

## Categories and category model nuggets

---

You can create *categories* that represent, in essence, higher-level concepts or topics to capture the key ideas, knowledge, and attitudes expressed in the text. Categories are made up of a set of descriptors, such as *concepts*, *types*, and *rules*. Together, these descriptors are used to identify whether or not a record or document belongs in a given category. A document or record can be scanned to see whether any of its text matches a descriptor. If a match is found, the document/record is assigned to that category. This process is called *categorization*.

Categories can be built automatically using Watson Studio's robust set of automated techniques, manually using additional insight you may have regarding the data, or a combination of both. You can also load a set of prebuilt categories from a text analysis package through the Model settings of this node. Manual creation of categories or refining categories can only be done through the interactive workbench.

A *category model nugget* contains a set of categories along with its descriptors. The model can be used to categorize a set of documents or records based on the text in each document/record. Every document or record is read and then assigned to each category for which a descriptor match was found. In this way, a document or record could be assigned to more than one category. You can use category model nuggets to see the essential ideas in open-ended survey responses or in a set of blog entries, for example.

- **Text Mining model nuggets**

You can run a Text Mining node to automatically generate a *concept* or *category* model nugget using the Generate directly option in the node settings. Or you can use a more hands-on, exploratory approach using the Build interactively mode to generate model nuggets from within the interactive workbench.

- **Interactive workbench mode**

From a Text Mining modeling node, you can choose to launch an interactive workbench session when your flow runs. In this workbench, you can extract key concepts from your text data, build categories, explore text link analysis patterns, and generate category models.

**Parent topic:** [Text Analytics](#)

## Text Mining model nuggets

---

You can run a Text Mining node to automatically generate a *concept* or *category* model nugget using the Generate directly option in the node settings. Or you can use a more hands-on, exploratory approach using the Build interactively mode to generate model nuggets from within the interactive workbench.

### Text Mining nugget: Concept model

---

A Text Mining concept model nugget is created whenever you successfully run a Text Mining node where you've selected the option to Generate a model directly in the node settings. Use a text mining concept model nugget for the real-time discovery of key concepts in other text data, such as scratch-pad data from a call center.

The concept model nugget itself comprises a list of concepts, which have been assigned to types. You can select any or all of the concepts in that model for scoring against other data. When you run a flow containing a Text Mining model nugget, new fields are added to the data according to the build mode selected in the settings of the Text Mining modeling node prior to building the model.

If the model nugget was generated using translated documents, the scoring will be performed in the translated language. Similarly, if the model nugget was generated using English as the language, you can specify a translation

language in the model nugget, since the documents will then be translated into English.

Text Mining model nuggets are placed in the Outputs pane at the upper-right are of the application when they're generated.

## Text Mining nugget: Category model

---

A Text Mining category model nugget is created whenever you generate a category model from within the interactive workbench. This modeling nugget contains a set of categories, whose definition is made up of concepts, types, TLA patterns, and/or category rules. The nugget is used to categorize survey responses, blog entries, other web feeds, and any other text data.

If you launch an interactive workbench session in the modeling node, you can explore the extraction results, refine the resources, and fine-tune your categories before you generate category models. When you run a flow containing a Text Mining model nugget, new fields are added to the data according to the build mode selected in the settings of the Text Mining node prior to building the model.

If the model nugget was generated using translated documents, the scoring will be performed in the translated language. Similarly, if the model nugget was generated using English as the language, you can specify a translation language in the model nugget, since the documents will then be translated into English.

Text Mining model nuggets are placed in the Outputs pane at the upper-right are of the application when they're generated.

**Parent topic:** [Mining for concepts and categories](#)

## Interactive workbench mode

---

From a Text Mining modeling node, you can choose to launch an interactive workbench session when your flow runs. In this workbench, you can extract key concepts from your text data, build categories, explore text link analysis patterns, and generate category models.

This section provides an overview of the interactive workbench interface, along with the major elements with which you will work, including:

- **Extraction results.** After performing an extraction, these are the key words and phrases identified and extracted from your text data, also referred to as *concepts*. These concepts are grouped into *types*. Using these concepts and types, you can explore your data and create your categories. These are managed in the Categories and concepts view.
- **Categories.** Using descriptors (such as extraction results, patterns, and rules) as a definition, you can manually or automatically create a set of categories to which documents and records are assigned based on whether or not they contain a part of the category definition. These are managed in the Categories and concepts view.
- **Text link analysis patterns.** If you have text link analysis (TLA) pattern rules in your linguistic resources or are using a resource template that already has some TLA rules, you can extract patterns from your text data. These patterns can help you uncover interesting relationships between concepts in your data. You can also use these patterns as descriptors in your categories. These are managed in the Text link analysis view.
- **[The Categories and concepts view](#)**  
The interactive workbench includes several views. With the Categories and concepts view, you can create and explore categories as well as explore and tweak the extraction results.
- **[The Clusters view](#)**  
In the Clusters view, you can build and explore cluster results found in your text data.
- **[The Text Link Analysis view](#)**  
In the Text link analysis view, you can build and explore text link analysis patterns found in your text data. Text link analysis (TLA) is a pattern-matching technology that enables you to define TLA rules and compare them to actual extracted concepts and relationships found in your text.

- **The Resource Editor view**  
Text Analytics rapidly and accurately captures key concepts from text data using a robust extraction engine. This engine relies heavily on linguistic resources to dictate how large amounts of unstructured, textual data should be analyzed and interpreted.
- **Setting options**  
You can click the Settings icon in various panes to change settings for the interactive workbench, such as extraction settings for concepts.
- **Generating a model nugget**  
When you're in an interactive session, you may want to use the work you've done to generate a category model nugget.

**Parent topic:** [Mining for concepts and categories](#)

## The Categories and concepts view

The interactive workbench includes several views. With the Categories and concepts view, you can create and explore categories as well as explore and tweak the extraction results.

*Categories* refers to a group of closely related ideas and patterns to which documents and records are assigned through a scoring process. *Concepts* refer to the most basic level of extraction results available to use as building blocks, called descriptors, for your categories.

Figure 1. Categories and concepts view

The screenshot shows the 'Categories and concepts' view in Watson Studio Desktop 2.0. The interface is divided into two main sections: 'Category data' and 'Concepts data'.

**Category data:** This section contains a table with columns for 'Category', 'Descriptors', and 'Docs'. It also has tabs for 'Build', 'Extend', 'Score', and 'Display', and a 'Collapse all' button. The table lists various categories and their associated descriptors and document counts.

Category	Descriptors	Docs
All	-	491
Uncategorized	-	132
No concepts extracted	-	0
leisure	28	14
house rooms	26	124
occupation	13	63
hotel facilities	13	24
trip	12	58

**Concepts data:** This section contains a table with columns for 'Concept', 'In', 'Global', 'Docs', and 'Type'. It also has tabs for 'Extract' and 'Display', and a 'Concept' dropdown. The table lists various concepts and their associated document counts and types.

Concept	In	Global	Docs	Type
female	157	26	<Unknown>	
excellent	147	95	<Positive>	
good	106	65	<Positive>	
leisure	106	14	<Unknown>	
male	99	10	<Unknown>	
room	99	81	<Unknown>	
business	78	11	<Unknown>	
friendly	65	47	<PositiveAttitude>	

The 'Category data' table also has a 'Full Path' column showing the hierarchy of categories. The 'Concepts data' table also has a 'Type' column showing the type of concept.

The Categories and concepts view is organized into panes.

### Categories pane

Located in the upper left corner, this area presents a table in which you can manage any categories you build. After extracting the concepts and types from your text data, you can begin building categories by using techniques such as semantic networks and concept inclusion, or by creating them manually. If you select a category name and click the Settings icon, the category settings open and display all of the descriptors that make up its definition, such as concepts, types, and rules. Not all automatic techniques are available for all languages.

When you select a row in the pane, you can then display information about corresponding documents/records or descriptors.

### Extraction results pane

Located in the lower left corner, this area presents the extraction results. When you run an extraction, the extraction engine reads through the text data, identifies the relevant concepts, and assigns a type to each. *Concepts* are words or phrases extracted from your text data. *Types* are semantic groupings of concepts stored in the form of type dictionaries. When the extraction is complete, concepts and types appear with color coding in this pane.

Text mining is an iterative process in which extraction results are reviewed according to the context of the text data, fine-tuned to produce new results, and then reevaluated. Extraction results can be refined by modifying the linguistic resources. This fine-tuning can be done in part directly from the extraction results or data pane.

## Data pane

---

The Data pane is located on the right. This pane presents a table containing the documents or records corresponding to a selection in another area of the view. Depending on what is selected, only the corresponding text appears in the Data pane. Once you make a selection, click Display to populate the Data pane with the corresponding text.

If you have a selection in another pane, the corresponding documents or records show the concepts highlighted in color to help you easily identify them in the text. You can also hover your mouse over color-coded items to display a tooltip showing name of the concept under which it was extracted and the type to which it was assigned.

## Searching and finding in the categories and concepts view

---

In some cases, you may need to locate information quickly in a particular section. Using the Find toolbar, you can enter the string you want to search for and define other search criteria such as case sensitivity or search direction. Then you can choose the pane in which you want to search.

1. In the Find field at the top of your screen, type the word string you want to search for. You can use the up and down arrow buttons to control the direction of the search.
2. From the drop-down, select the name of the pane in which you want to search and then click one of the arrow buttons. If a match is found, the text is highlighted in the window.
3. To look for the next match, click the arrow button again.

**Parent topic:** [Interactive workbench mode](#)

## The Clusters view

---

In the Clusters view, you can build and explore cluster results found in your text data.

*Clusters* are groupings of concepts generated by clustering algorithms based on how often concepts occur and how often they appear together. The goal of clusters is to group concepts that co-occur together while the goal of categories is to group documents or records based on how the text they contain matches the descriptors (concepts, rules, patterns) for each category.

The more often the concepts within a cluster occur together coupled with the less frequently they occur with other concepts, the better the cluster is at identifying interesting concept relationships. Two concepts co-occur when they both appear (or one of their synonyms or terms appear) in the same document or record.

You can build clusters and explore them in a set of charts and graphs that could help you uncover relationships among concepts that would otherwise be too time-consuming to find. While you cannot add entire clusters to your categories, you can add the concepts in a cluster to a category through the Cluster Definitions dialog box.

You can make changes to the settings for clustering to influence the results.

The Clusters view is organized into panes, each of which can be hidden or shown by selecting its name from the drop-down. Typically, only the Clusters pane is visible.

## Clusters pane

---

Located on the left side, this pane presents the clusters that were discovered in the text data. You can create clustering results by clicking Build. Clusters are formed by a clustering algorithm, which attempts to identify concepts that occur together frequently.

Whenever a new extraction takes place, the cluster results are cleared, and you have to rebuild the clusters to get the latest results. When building the clusters, you can change some settings, such as the maximum number of clusters to create, the maximum number of concepts it can contain, or the maximum number of links with external concepts it can have.

## Data pane

---

The Data pane is located in the lower right corner and is hidden by default. You cannot display any Data pane results from the Clusters pane since these clusters span multiple documents/records, making the data results uninteresting. However, you can see the data corresponding to a selection within the Cluster Definitions dialog box. Depending on what is selected in that dialog box, only the corresponding text appears in the Data pane. Once you make a selection, click Display & to populate the Data pane with the documents or records that contain all of the concepts together.

The corresponding documents or records show the concepts highlighted in color to help you easily identify them in the text. You can also hover your mouse over color-coded items to display the concept under which it was extracted and the type to which it was assigned. The Data pane can contain multiple columns but the text field column is always shown. It carries the name of the text field that was used during extraction or a document name if the text data is in many different files. Other columns are available.

**Parent topic:** [Interactive workbench mode](#)

## The Text Link Analysis view

---

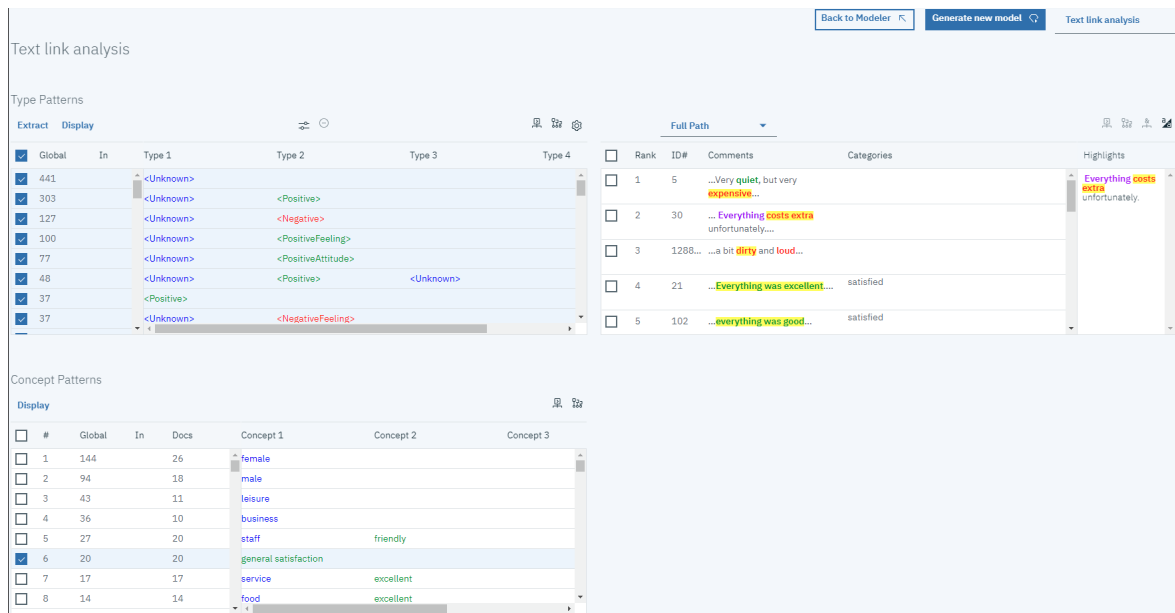
In the Text link analysis view, you can build and explore text link analysis patterns found in your text data. Text link analysis (TLA) is a pattern-matching technology that enables you to define TLA rules and compare them to actual extracted concepts and relationships found in your text.

Patterns are most useful when you are attempting to discover relationships between concepts or opinions about a particular subject. Some examples include wanting to extract opinions on products from survey data, genomic relationships from within medical research papers, or relationships between people or places from intelligence data.

Once you've extracted some TLA patterns, you can explore them in the Data pane and even add them to categories in the Categories and concepts view. There must be some TLA rules defined in the resource template or libraries you are using in order to extract TLA results.

If you chose to extract TLA pattern results, the results are presented in this view. If you haven't chosen to do so, you'll have to click Extract. You can click the Settings icon to change the extraction settings.

Figure 1. Text Link Analysis view



The Text link analysis view is organized into panes.

## Type and Concept Patterns panes

Located on the left side, the Type and Concept Pattern panes are two interconnected panes in which you can explore and select your TLA pattern results. Patterns are made up of a series of up to either six types or six concepts. The TLA pattern rule as it is defined in the linguistic resources dictates the complexity of the pattern results.

Pattern results are first grouped at the type level and then divided into concept patterns. For this reason, there are two different result panes: Type Patterns (upper left) and Concept Patterns (lower left).

- **Type Patterns.** The Type Patterns pane presents extracted patterns consisting of two or more related types matching a TLA pattern rule. Type patterns are shown as `<Organization> + <Location> + <Positive>`, which might provide positive feedback about an organization in a specific location.
- **Concept Patterns.** The Concept Patterns pane presents the extracted patterns at the concept level for all of the type pattern(s) currently selected in the Type Patterns pane above it. Concept patterns follow a structure such as `hotel + paris + wonderful`.

Just as with the extraction results in the Categories and concepts view, you can review the results here. If you see any refinements you would like to make to the types and concepts that make up these patterns, you make those in the Extraction Results pane in the Categories and concepts view and reextract your patterns.

## Data pane

The Data pane is located on the right. This pane presents a table containing the documents or records corresponding to a selection in another area of the view. Depending on what is selected, only the corresponding text appears in the Data pane. Once you make a selection, click Display to populate the Data pane with the corresponding text.

If you have a selection in another pane, the corresponding documents or records show the concepts highlighted in color to help you easily identify them in the text. You can also hover your mouse over color-coded items to display a tooltip showing name of the concept under which it was extracted and the type to which it was assigned.

**Parent topic:** [Interactive workbench mode](#)

## The Resource Editor view

Text Analytics rapidly and accurately captures key concepts from text data using a robust extraction engine. This engine relies heavily on linguistic resources to dictate how large amounts of unstructured, textual data should be analyzed and interpreted.

The Resource Editor view is where you can view and fine-tune the linguistic resources used to extract concepts, group them under types, discover patterns in the text data, and much more. Text Analytics offers several preconfigured resource templates. Also, in some languages, you can also use the resources in text analysis packages.

Since these resources may not always be perfectly adapted to the context of your data, you can create, edit, and manage your own resources for a particular context or domain in the Resource Editor.

To simplify the process of fine-tuning your linguistic resources, you can perform common dictionary tasks directly from the Categories and concepts view through context menus in the Extraction Results and Data panes.

The operations you perform in the Resource Editor view revolve around the management and fine-tuning of the linguistic resources. These resources are stored in the form of templates and libraries. The Resource Editor view is organized into four parts: Library Tree pane, Type Dictionary pane, Substitution Dictionary pane, and Exclude Dictionary pane.

**Parent topic:** [Interactive workbench mode](#)

## Setting options

---

You can click the Settings icon in various panes to change settings for the interactive workbench, such as extraction settings for concepts.

In the Categories and concepts view, categories are built from descriptors derived from either types or type patterns. In the table, you can select the individual types or patterns to include in the category building process. Click the Settings icons to change settings for the Categories and Extraction results (Concepts) panes.

### Settings for categories (Category data)

---

The following Build settings are available:

- Build categories from. If you select Types, the categories will be built from the concepts belonging to the selected types. So if you select the <Budget> type in the table, categories such as `cost` or `price` could be produced since `cost` and `price` are concepts assigned to the <Budget> type.

By default, only the types that capture the most records or documents are selected. This pre-selection allows you to quickly focus in on the most interesting types and avoid building uninteresting categories. The table displays the types in descending order starting with the one with the greatest number of records or documents (Doc. count). Types from the `Opinions` library are deselected by default in the types table.

The input you choose affects the categories you obtain. When you choose to use Types as input, you can see the clearly related concepts more easily. For example, if you build categories using Types as input, you could obtain a category `Fruit` with concepts such as `apple`, `pear`, `citrus fruits`, `orange` and so on. If you choose Type Patterns as input instead and select the pattern <Unknown> + <Positive>, for example, then you might get a category `fruit + <Positive>` with one or two kinds of fruit such as `fruit + tasty` and `apple + good`. This second result only shows 2 concept patterns because the other occurrences of fruit are not necessarily positively qualified. And while this might be good enough for your current text data, in longitudinal studies where you use different document sets, you may want to manually add in other descriptors such as `citrus fruit + positive` or use types. Using types alone as input will help you to find all possible fruit.

If you select Type Patterns, categories are built from patterns rather than types and concepts on their own. In that way, any records or documents containing a concept pattern belonging to the selected type pattern are categorized. So, if you select the <Budget> and <Positive> type pattern in the table, categories such as `cost & <Positive>` or `rates & excellent` could be produced.



When using type patterns as input for automated category building, there are times when the techniques identify multiple ways to form the category structure. Technically, there is no single right way to produce the categories; however you might find one structure more suited to your analysis than another. To help customize the output in this case, you can designate a type as the preferred focus. All the top-level categories produced will come from a concept of the type you select here (and no other type). Every subcategory will contain a text link pattern from this type. Choose this type in the Structure categories by pattern type: field and the table will be updated to show only the applicable patterns containing the selected type. More often than not, <Unknown> will be preselected for you. This results in all of the patterns containing the type <Unknown> being selected. The table displays the types in descending order starting with the one with the greatest number of records or documents (Doc. count).

- **Techniques.** Because every dataset is unique, the number of methods and the order in which you apply them may change over time. Since your text mining goals may be different from one set of data to the next, you may need to experiment with the different techniques to see which one produces the best results for the given text data.

You do not need to be an expert in these settings to use them. By default, the most common and average settings are already selected. Therefore, you can bypass the advanced setting dialogs and go straight to building your categories. Likewise, if you make changes here, you do not have to come back to the settings dialog each time since the latest settings are always retained.

Select one of the following techniques and then click Advanced settings. None of the automatic techniques will perfectly categorize your data; therefore we recommend finding and applying one or more automatic techniques that work well with your data. You can't build using linguistic and frequency techniques simultaneously.

- Use linguistic techniques to build categories. See [Advanced linguistic settings](#).
- Use frequencies to build categories. See [Advanced frequency settings](#).

The following Extend settings are available:

- **Category input.** Select the option Unused extraction results if you want categories to be built from extraction results that aren't used in any existing categories. This minimizes the tendency for records to match multiple categories and limits the number of categories produced. Or select the option All extraction results if you want categories to be built using any of the extraction results. This is most useful when no or few categories already exist.

Each of the grouping techniques available is well suited to certain types of data and situations, but often it's helpful to combine techniques in the same analysis to capture the full range of documents or records. You may see a concept in multiple categories or find redundant categories. The concept inclusion technique builds categories by grouping multiterm concepts (compound words) based on whether they contain words that are subsets or supersets of a word in the other. For example, the concept *seat* would be grouped with *safety seat*, *seat belt*, and *seat belt buckle*. . The semantic network technique begins by identifying the possible senses of each concept from its extensive index of word relationships and then creates categories by grouping related concepts. This technique is best when the concepts are known to the semantic network and are not too ambiguous. It's less helpful when text contains specialized terminology or jargon unknown to the network. In one example, the concept *granny smith apple* could be grouped with *gala apple* and *winesap apple* since they're siblings of the *granny smith*. In another example, the concept *animal* might be grouped with *cat* and *kangaroo* since they're hyponyms of *animal*. This technique is available for English text only.

The Maximum search distance option is only available if you select the semantic network technique. Select how far you want the techniques to search before producing categories. The lower the value, the fewer results you will get; however, these results will be less noisy and are more likely to be significantly linked or associated with each other. The higher the value, the more results you might get; however, these results may be less reliable or relevant. While this option is globally applied to all techniques, its effect is greatest on co-occurrences and semantic networks.

Select Prevent pairing of specific concepts if you want to stop the process from grouping or pairing two concepts together in the output. To create or manage concept pairs, click Manage pairs.



- Where possible. Choose whether to simply extend, generalize the descriptors using wildcards, or both.
  - Extend and generalize. This option will extend the selected categories and then generalize the descriptors. When you choose to generalize, the product will create generic category rules in categories using the asterisk wildcard. For example, instead of producing multiple descriptors such as [apple tart + .] and [apple sauce + .], using wildcards might produce [apple \* + .]. If you generalize with wildcards, you'll often get exactly the same number of records or documents as you did before. However, this option has the advantage of reducing the number and simplifying category descriptors. Additionally, this option increases the ability to categorize more records or documents using these categories on new text data (for example, in longitudinal/wave studies).
  - Extend only. This option will extend your categories without generalizing. It can be helpful to first choose the Extend only option for manually-created categories and then extend the same categories again using the Extend and generalize option.
  - Generalize only. This option will generalize the descriptors without extending your categories in any other way.
  - Maximum number of items to extend a descriptor by. When extending a descriptor with items (concepts, types, and other expressions), define the maximum number of items that can be added to a single descriptor. If you set this limit to 10, then no more than 10 additional items can be added to an existing descriptor. If there are more than 10 items to be added, the techniques stop adding new items after the tenth is added. Doing so can make a descriptor list shorter but doesn't guarantee that the most interesting items were used first.
  - Also extend subcategories. This option will also extend any subcategories below the selected categories.
  - Extend empty categories with descriptors generated from the category name. This method applies only to empty categories, which have 0 descriptors. If a category already contains descriptors, it won't be extended in this way. This option attempts to automatically create descriptors for each category based on the words that make up the name of the category. The category name is scanned to see if words in the name match any extracted concepts. If a concept is recognized, it's used to find matching concept patterns and these both are used to form descriptors for the category. This option produces the best results when the category names are both long and descriptive. This is a quick method for generating category descriptors, which in turn enable the category to capture records that contain those descriptors. This option is most useful when you import categories from somewhere else or when you create categories manually with long descriptive names.
  - Generate descriptors as. This option only applies if the preceding option is selected. Choose the Concepts option to produce the resulting descriptors in the form of concepts, regardless of whether they have been extracted from the source text. Or choose the Patterns option to produce the resulting descriptors in the form of patterns, regardless of whether the resulting patterns or any patterns have been extracted.

## Settings for extraction results (Concepts data)

---

- Enable Text Link Analysis pattern extraction. Specifies that you want to extract TLA patterns from your text data. This option may significantly lengthen the extraction time.
- Accommodate punctuation errors. This option temporarily normalizes text containing punctuation errors (for example, improper usage) during extraction to improve the extractability of concepts. This option is extremely useful when text is short and of poor quality (as, for example, in open-ended survey responses, e-mail, and CRM data), or when the text contains many abbreviations.
- Accommodate spelling for a minimum root character limit. This option applies a fuzzy grouping technique that helps group commonly misspelled words or closely spelled words under one concept. The fuzzy grouping algorithm temporarily strips all vowels (except the first one) and strips double/triple consonants from extracted words and then compares them to see if they are the same so, for example, `modeling` and `modelling` would be grouped together. However, if each term is assigned to a different type, excluding the <Unknown> type, the fuzzy grouping technique won't be applied.
- Extract uniterms. This option extracts single words (uniterms) as long as the word isn't already part of a compound word and if it's either a noun or an unrecognized part of speech.
- Extract non-linguistic entities. This option extracts non-linguistic entities, such as phone numbers, social security numbers, times, dates, currencies, digits, percentages, e-mail addresses, and HTTP addresses. You

can include or exclude certain types of nonlinguistic entities. By disabling any unnecessary entities, the extraction engine won't waste processing time.

- Uppercase algorithm. This option extracts simple and compound terms that aren't in the built-in dictionaries as long as the first letter of the term is in uppercase. This option offers a good way to extract most proper nouns.
- Group partial and full person names together when possible. This option groups names that appear differently in the text together. This feature is helpful since names are often referred to in their full form at the beginning of the text and then only by a shorter version. This option attempts to match any uniterm with the <Unknown> type to the last word of any of the compound terms that is typed as <Person>. For example, if *doe* is found and initially typed as <Unknown>, the extraction engine checks to see if any compound terms in the <Person> type include *doe* as the last word, such as *john doe*. This option doesn't apply to first names since most are never extracted as uniterms.
- Maximum nonfunction word permutation. This option specifies the maximum number of nonfunction words that can be present when applying the permutation technique. This permutation technique groups similar phrases that differ from each other only by the nonfunction words (for example, *of* and *the*) contained, regardless of inflection. For example, let's say you set this value to at most two words and both *company officials* and *officials of the company* were extracted. In this case, both extracted terms would be grouped together in the final concept list since both terms are deemed to be the same when *of the* is ignored.
- Use derivation when grouping multiterms. When processing Big Data, select this option to group multiterms by using derivation rules.

- **Advanced linguistic settings**

When you build categories, you can select from a number of advanced linguistic category building techniques such as *concept inclusion* and *semantic networks* (English text only). These techniques can be used individually or in combination with each other to create categories.

- **Advanced frequency settings**

You can build categories based on a straightforward and mechanical frequency technique. With this technique, you can build one category for each item (type, concept, or pattern) that was found above a given record or document count. Additionally, you can build a single category for all of the less frequently occurring items. By count, we refer to the number of records or documents containing the extracted concept (and any of its synonyms), type, or pattern in question as opposed to the total number of occurrences in the entire text.

**Parent topic:** [Interactive workbench mode](#)

## Advanced linguistic settings

---

When you build categories, you can select from a number of advanced linguistic category building techniques such as *concept inclusion* and *semantic networks* (English text only). These techniques can be used individually or in combination with each other to create categories.

Keep in mind that because every dataset is unique, the number of methods and the order in which you apply them may change over time. Since your text mining goals may be different from one set of data to the next, you may need to experiment with the different techniques to see which one produces the best results for the given text data. None of the automatic techniques will perfectly categorize your data; therefore we recommend finding and applying one or more automatic techniques that work well with your data.

The following advanced settings are available for the Use linguistic techniques to build categories option in the category settings.

### Category input

---

Select what the categories will be built from:

- Unused extraction results. This option enables categories to be built from extraction results that aren't used in any existing categories. This minimizes the tendency for records to match multiple categories and limits the number of categories produced.

- All extraction results. This option enables categories to be built using any of the extraction results. This is most useful when no or few categories already exist.

## Category output

---

Select the general structure for the categories that will be built:

- Hierarchical with subcategories. This option creates subcategories and sub-subcategories. You can set the depth of your categories by choosing the maximum number of levels that can be created. For example, if you choose 3, categories could contain subcategories and those subcategories could also have subcategories.
- Flat categories (single level only). This option builds only one level of categories, meaning that no subcategories will be generated.

## Grouping techniques

---

Each of the techniques available is well suited to certain types of data and situations, but often it's helpful to combine techniques in the same analysis to capture the full range of documents or records. You may see a concept in multiple categories or find redundant categories.

- Group by concept inclusion. This technique builds categories by grouping multiterm concepts (compound words) based on whether they contain words that are subsets or supersets of a word in the other. For example, the concept `seat` would be grouped with `safety seat`, `seat belt`, and `seat belt buckle`.
- Group by semantic network. This technique begins by identifying the possible senses of each concept from its extensive index of word relationships and then creates categories by grouping related concepts. This technique is best when the concepts are known to the semantic network and are not too ambiguous. It is less helpful when text contains specialized terminology or jargon unknown to the network. In one example, the concept `granny smith apple` could be grouped with `gala apple` and `winesap apple` since they are siblings of the `granny smith`. In another example, the concept `animal` might be grouped with `cat` and `kangaroo` since they are hyponyms of `animal`. This technique is available for English text only.
- Maximum search distance. This setting is only available if you select the Group by semantic network option. Select how far you want the techniques to search before producing categories. The lower the value, the fewer results you will get; however, these results will be less noisy and are more likely to be significantly linked or associated with each other. The higher the value, the more results you might get; however, these results may be less reliable or relevant. While this option is globally applied to all techniques, its effect is greatest on co-occurrences and semantic networks.
- Prevent pairing of specific concepts. Select this option to stop the process from grouping or pairing two concepts together in the output. To create or manage concept pairs, click Manage pairs.
- Generalize with wildcards where possible. Select this option to allow Modeler to generate generic rules in categories using the asterisk wildcard. For example, instead of producing multiple descriptors such as `[apple tart + .]` and `[apple sauce + .]`, using wildcards might produce `[apple * + .]`. If you generalize with wildcards, you'll often get exactly the same number of records or documents as you did before. However, this option has the advantage of reducing the number and simplifying category descriptors. Additionally, this option increases the ability to categorize more records or documents using these categories on new text data (for example, in longitudinal/wave studies).

## Other options for building categories

---

Maximum number of top level categories created. Use this option to limit the number of categories that can be generated the next time you click Build in the categories pane. In some cases, you might get better results if you set this value high and then delete any of the uninteresting categories.

Minimum number of descriptors and/or subcategories per descriptor. Use this option to define the minimum number of descriptors and subcategories a category must contain in order to be created. This option helps limit the creation of categories that don't capture a significant number of records or documents.

Allow descriptors to appear in more than one category. When selected, this option allows descriptors to be used in more than one of the categories that will be built next. This option is generally selected since items commonly or "naturally" fall into two or more categories, and allowing them to do so usually leads to higher quality categories. If you don't select this option, you reduce the overlap of records in multiple categories and -depending on the type of data you have - this might be desirable. However, with most types of data, restricting descriptors to a single category usually results in a loss of quality or category coverage. For example, let's say you have the concept `car seat manufacturer`. With this option, this concept could appear in one category based on the text `car seat` and in another one based on `manufacturer`. But if this option is not selected, although you may still get both categories, the concept `car seat manufacturer` will only appear as a descriptor in the category it best matches based on several factors including the number of records in which `car seat` and `manufacturer` each occur.

Resolve duplicate category names by. Select how to handle any new categories or subcategories whose names would be the same as existing categories. You can either merge the new ones (and their descriptors) with the existing categories with the same name, or you can choose to skip the creation of any categories if a duplicate name is found in the existing categories.

**Parent topic:** [Setting options](#)

## Advanced frequency settings

---

You can build categories based on a straightforward and mechanical frequency technique. With this technique, you can build one category for each item (type, concept, or pattern) that was found above a given record or document count. Additionally, you can build a single category for all of the less frequently occurring items. By count, we refer to the number of records or documents containing the extracted concept (and any of its synonyms), type, or pattern in question as opposed to the total number of occurrences in the entire text.

Grouping frequently occurring items can yield interesting results, since it may indicate a common or significant response. The technique is very useful on the unused extraction results after other techniques have been applied. Another application is to run this technique immediately after extraction when no other categories exist, edit the results to delete uninteresting categories, and then extend those categories so that they match even more records or documents.

Instead of using this technique, you could sort the concepts or concept patterns by descending number of records or documents in the extraction results pane and then drag-and-drop the top ones into the categories pane to create the corresponding categories.

The following advanced settings are available for the Use frequencies to build categories option in the category settings.

Generate category descriptors at. Select the kind of input for descriptors.

- **Concepts level.** Selecting this option means that concepts or concept patterns frequencies will be used. Concepts will be used if types were selected as input for category building and concept patterns are used, if type patterns were selected. In general, applying this technique to the concept level will produce more specific results, since concepts and concept patterns represent a lower level of measurement.
- **Types level.** Selecting this option means that type or type patterns frequencies will be used. Types will be used if types were selected as input for category building and type patterns are used, if type patterns were selected. By applying this technique to the type level, you can get a quick view of the kind of information given.

Minimum record/doc. count for items to have their own category. With this option, you can build categories from frequently occurring items. This option restricts the output to only those categories containing a descriptor that occurred in at least X number of records or documents, where X is the value to enter for this option.

Group all remaining items into a category called. Use this option if you want to group all concepts or types occurring infrequently into a single catch-all category with the name of your choice. By default, this category is named *Other*.

Category input. Select the group to which to apply the techniques:

- **Unused extraction results.** This option enables categories to be built from extraction results that aren't used in any existing categories. This minimizes the tendency for records to match multiple categories and limits the number of categories produced.
- **All extraction results.** This option enables categories to be built using any of the extraction results. This is most useful when no or few categories already exist.

Resolve duplicate category names by. Select how to handle any new categories or subcategories whose names would be the same as existing categories. You can either merge the new ones (and their descriptors) with the existing categories with the same name, or you can choose to skip the creation of any categories if a duplicate name is found in the existing categories.

**Parent topic:** [Setting options](#)

## Generating a model nugget

---

When you're in an interactive session, you may want to use the work you've done to generate a category model nugget.

A model generated from an interactive workbench session is a category model nugget. You must have at least one category in the Categories and concepts view to generate a category model nugget.

### To generate a category model nugget

---

Click the Generate new model button at the top right of the interactive workbench session. A model nugget is generated directly onto your flow canvas with the default name.

**Parent topic:** [Interactive workbench mode](#)

## Outputs

---

Output nodes provide the means to obtain information about your data and models. They also provide a mechanism for exporting data in various formats to interface with your other software tools.

- **Table node**  
The Table node creates a table that lists the values in your data. All fields and all values in the stream are included, making this an easy way to inspect your data values or export them in an easily readable form. Optionally, you can highlight records that meet a certain condition.
- **Matrix node**  
Use the Matrix to create a table that shows relationships between fields. It is most commonly used to show the relationship between two categorical fields (flag, nominal, or ordinal), but it can also be used to show relationships between continuous (numeric range) fields.
- **Analysis node**  
With the Analysis node, you can evaluate the ability of a model to generate accurate predictions. Analysis nodes perform various comparisons between predicted values and actual values (your target field) for one or more model nuggets. You can also use Analysis nodes to compare predictive models to other predictive models.
- **Data Audit node**  
The Data Audit node provides a comprehensive first look at the data you bring into Watson Studio, presented in an easy-to-read matrix that can be sorted.
- **Transform node**  
Normalizing input fields is an important step before using traditional scoring techniques such as regression, logistic regression, and discriminant analysis. These techniques carry assumptions about normal distributions of data that may not be true for many raw data files. One approach to dealing with real-world data is to apply transformations that move a raw data element toward a more normal distribution. In addition, normalized

fields can easily be compared with each other; for example, income and age are on totally different scales in a raw data file but, when normalized, the relative impact of each can be easily interpreted.

- **Statistics node**

The Statistics node gives you basic summary information about numeric fields. You can get summary statistics for individual fields and correlations between fields.

- **Means node**

The Means node compares the means between independent groups or between pairs of related fields to test whether a significant difference exists. For example, you can compare mean revenues before and after running a promotion or compare revenues from customers who didn't receive the promotion with those who did.

- **Report node**

You can use the Report node to create formatted reports containing fixed text, data, or other expressions derived from the data. Specify the format of the report by using text templates to define the fixed text and the data output constructions. You can provide custom text formatting using HTML tags in the template and by setting output options. Data values and other conditional output are included in the report using CLEM expressions in the template.

- **Set Globals node**

The Set Globals node scans the data and computes summary values that can be used in CLEM expressions.

- **Sim Fit node**

The Simulation Fitting node fits a set of candidate statistical distributions to each field in the data. The fit of each distribution to a field is assessed using a goodness of fit criterion. When a Simulation Fitting node runs, a Simulation Generate node is built (or an existing node is updated). Each field is assigned its best fitting distribution. The Simulation Generate node can then be used to generate simulated data for each field.

- **KDE Simulation node**

Kernel Density Estimation (KDE)® uses the Ball Tree or KD Tree algorithms for efficient queries, and walks the line between unsupervised learning, feature engineering, and data modeling.

- **Extension Output node**

With the Extension Output node, you can run R scripts or Python for Spark scripts to produce output.

**Parent topic:** [Nodes palette](#)

## Table node

---

The Table node creates a table that lists the values in your data. All fields and all values in the stream are included, making this an easy way to inspect your data values or export them in an easily readable form. Optionally, you can highlight records that meet a certain condition.

Note: Unless you are working with small datasets, we recommend that you select a subset of the data to pass into the Table node. The Table node cannot display properly when the number of records surpasses a size that can be contained in the display structure (for example, 100 million rows).

**Parent topic:** [Outputs](#)

## Matrix node

---

Use the Matrix to create a table that shows relationships between fields. It is most commonly used to show the relationship between two categorical fields (flag, nominal, or ordinal), but it can also be used to show relationships between continuous (numeric range) fields.

**Parent topic:** [Outputs](#)

## Analysis node

---

With the Analysis node, you can evaluate the ability of a model to generate accurate predictions. Analysis nodes perform various comparisons between predicted values and actual values (your target field) for one or more model nuggets. You can also use Analysis nodes to compare predictive models to other predictive models.

When you execute an Analysis node, a summary of the analysis results is automatically added to the Analysis section on the Summary tab for each model nugget in the executed flow. The detailed analysis results appear on the Outputs tab of the manager window or can be written directly to a file.

Note: Because Analysis nodes compare predicted values to actual values, they are only useful with supervised models (those that require a target field). For unsupervised models such as clustering algorithms, there are no actual results available to use as a basis for comparison.

**Parent topic:** [Outputs](#)

## Data Audit node

---

The Data Audit node provides a comprehensive first look at the data you bring into Watson Studio, presented in an easy-to-read matrix that can be sorted.

When you run a Data Audit node, output is generated that includes:

- Summary statistics, histograms, and distribution graphs that may be useful in gaining a preliminary understanding of the data.
- Information about outliers, extremes, and missing values.

## Using the Data Audit node

---

The Data Audit node can be attached directly to an Import node or downstream from an instantiated Type node.

Screening or sampling the data. Because an initial audit is particularly effective when dealing with big data, you might use a Sample node to reduce processing time during the initial exploration by selecting only a subset of records. The Data Audit node can also be used in combination with nodes such as Feature Selection and Anomaly Detection in the exploratory stages of analysis.

**Parent topic:** [Outputs](#)

## Transform node

---

Normalizing input fields is an important step before using traditional scoring techniques such as regression, logistic regression, and discriminant analysis. These techniques carry assumptions about normal distributions of data that may not be true for many raw data files. One approach to dealing with real-world data is to apply transformations that move a raw data element toward a more normal distribution. In addition, normalized fields can easily be compared with each other; for example, income and age are on totally different scales in a raw data file but, when normalized, the relative impact of each can be easily interpreted.

The Transform node provides an output viewer that enables you to perform a rapid visual assessment of the best transformation to use. You can see at a glance whether variables are normally distributed and, if necessary, choose the transformation you want and apply it. You can pick multiple fields and perform one transformation per field.

After selecting the preferred transformations for the fields, you can generate Derive or Filler nodes that perform the transformations and attach these nodes to the flow. The Derive node creates new fields, while the Filler node transforms the existing ones.

## Transform node fields settings

---

Under the FIELDS section in the node properties, you can specify which fields of the data you want to use for viewing possible transformations and applying them. Only numeric fields can be transformed. Select one or more numeric fields.

**Parent topic:** [Outputs](#)



## Statistics node

---

The Statistics node gives you basic summary information about numeric fields. You can get summary statistics for individual fields and correlations between fields.

**Parent topic:** [Outputs](#)

## Means node

---

The Means node compares the means between independent groups or between pairs of related fields to test whether a significant difference exists. For example, you can compare mean revenues before and after running a promotion or compare revenues from customers who didn't receive the promotion with those who did.

You can compare means in two different ways, depending on your data:

- Between groups within a field. To compare independent groups, select a test field and a grouping field. For example, you could exclude a sample of "holdout" customers when sending a promotion and compare mean revenues for the holdout group with all of the others. In this case, you would specify a single test field that indicates the revenue for each customer, with a flag or nominal field that indicates whether they received the offer. The samples are independent in the sense that each record is assigned to one group or another, and there is no way to link a specific member of one group to a specific member of another. You can also specify a nominal field with more than two values to compare the means for multiple groups. When executed, the node calculates a one-way ANOVA test on the selected fields. In cases where there are only two field groups, the one-way ANOVA results are essentially the same as an independent-samples  $t$  test.
- Between pairs of fields. When comparing means for two related fields, the groups must be paired in some way for the results to be meaningful. For example, you could compare the mean revenues from the same group of customers before and after running a promotion or compare usage rates for a service between husband-wife pairs to see if they are different. Each record contains two separate but related measures that can be compared meaningfully. When executed, the node calculates a paired-samples  $t$  test on each field pair selected.

**Parent topic:** [Outputs](#)

## Report node

---

You can use the Report node to create formatted reports containing fixed text, data, or other expressions derived from the data. Specify the format of the report by using text templates to define the fixed text and the data output constructions. You can provide custom text formatting using HTML tags in the template and by setting output options. Data values and other conditional output are included in the report using CLEM expressions in the template.

## Alternatives to the Report node

---

The Report node is most typically used to list records or cases output from a flow, such as all records meeting a certain condition. In this regard, it can be thought of as a less-structured alternative to the Table node.

- If you want a report that lists field information or anything else that is defined in the flow, rather than the data itself (such as field definitions specified in a Type node), then you can use a script instead.
- To produce a list of field names without using scripting, you can use a Table node preceded by a Sample node that discards all records. This produces a table with no rows, which can be transposed on export to produce a list of field names in a single column. (Select the Transpose data option in the Table node to do this.)

**Parent topic:** [Outputs](#)



## Set Globals node

---

The Set Globals node scans the data and computes summary values that can be used in CLEM expressions.

For example, you can use a Set Globals node to compute statistics for a field called `age` and then use the overall mean of `age` in CLEM expressions by inserting the function `@GLOBAL_MEAN (age)`.

**Parent topic:** [Outputs](#)

## Sim Fit node

---

The Simulation Fitting node fits a set of candidate statistical distributions to each field in the data. The fit of each distribution to a field is assessed using a goodness of fit criterion. When a Simulation Fitting node runs, a Simulation Generate node is built (or an existing node is updated). Each field is assigned its best fitting distribution. The Simulation Generate node can then be used to generate simulated data for each field.

Although the Simulation Fitting node is a terminal node, it does not add output to the Outputs panel, or export data.

Note: If the historical data is sparse (that is, there are many missing values), it may be difficult for the fitting component to find enough valid values to fit distributions to the data. In cases where the data is sparse, before fitting you should either remove the sparse fields if they are not required, or impute the missing values. Using the QUALITY options in the Data Audit node, you can view the number of complete records, identify which fields are sparse, and select an imputation method. If there are an insufficient number of records for distribution fitting, you can use a Balance node to increase the number of records.

### Using a Sim Fit node to automatically create a Sim Gen node

---

The first time the Simulation Fitting node is run, a Simulation Generate node is generated with an update link to the Simulation Fitting node. If the Simulation Fitting node is run again, a new Simulation Generate node will be generated only if the update link has been removed. You can also use a Simulation Fitting node to update a connected Simulation Generate node. The result depends on whether the same fields are present in both nodes, and if the fields are unlocked in the Simulation Generate node. See [Sim Gen node](#) for more information.

A Simulation Fitting node can only have an update link to a Simulation Generate node. To define an update link to a Simulation Generate node, follow these steps:

1. Right-click the Simulation Fitting node and select Define Update Link.
2. Click the Simulation Generate node to which you want to define an update link.

To remove an update link between a Simulation Fitting node and a Simulation Generate node, right-click the update link and select Remove Link.

## Distribution fitting

---

A statistical distribution is the theoretical frequency of the occurrence of values that a variable can take. In the Simulation Fitting node, a set of theoretical statistical distributions is compared to each field of data. The parameters of the theoretical distribution are adjusted to give the best fit to the data according to a measurement of the goodness of fit; either the Anderson-Darling criterion or the Kolmogorov-Smirnov criterion. The results of the distribution fitting by the Simulation Fitting node show which distributions were fitted, the best estimates of the parameters for each distribution, and how well each distribution fits the data. During distribution fitting, correlations between fields with numeric storage types, and contingencies between fields with a categorical distribution, are also calculated. The results of the distribution fitting are used to create a Simulation Generate node.

Before any distributions are fitted to your data, the first 1000 records are examined for missing values. If there are too many missing values, distribution fitting is not possible. If so, you must decide whether either of the following options are appropriate:

- Use an upstream node to remove records with missing values
- Use an upstream node to impute values for missing value.

Distribution fitting does not exclude user-missing values. If your data has user-missing values and you want those values to be excluded from distribution fitting, then you should set those values to system missing.

The role of a field is not taken into account when the distributions are fitted. For example, fields with the role Target are treated the same as fields with roles of Input, None, Both, Partition, Split, Frequency, and ID.

Fields are treated differently during distribution fitting according to their storage type and measurement level. The treatment of fields during distribution fitting is described in the following table.

Table 1. Distribution fitting according to storage type and measurement level of fields

Storage type			Measurement Level			
	Continuous	Categorical	Flag	Nominal	Ordinal	Typeless
String	Impossible		Categorical, dice and fixed distributions are fitted			
Integer						
Real						
Time	All distributions are fitted. Correlations and contingencies are calculated.		The categorical distribution is fitted. Correlations are not calculated.		Binomial, negative binomial and Poisson distributions are fitted, and correlations are calculated.	Field is ignored and not passed to the Simulation Generate node.
Date						
Timestamp						
Unknown			Appropriate storage type is determined from the data.			

Fields with the measurement level ordinal are treated like continuous fields and are included in the correlations table in the Simulation Generate node. If you want a distribution other than binomial, negative binomial, or Poisson to be fitted to an ordinal field, you must change the measurement level of the field to continuous. If you have previously defined a label for each value of an ordinal field, and then change the measurement level to continuous, the labels will be lost.

Fields that have single values are not treated differently during distribution fitting to fields with multiple values. Fields with the storage type time, date, or timestamp are treated as numeric.

## Fitting distributions to split fields

If your data contains a split field, and you want distribution fitting to be carried out separately for each split, you must transform the data by using an upstream Restructure node. Using the Restructure node, generate a new field for each value of the split field. You can then use this restructured data for distribution fitting in the Simulation Fitting node.

**Parent topic:** [Outputs](#)

## KDE Simulation node

---

Kernel Density Estimation (KDE)© uses the Ball Tree or KD Tree algorithms for efficient queries, and walks the line between unsupervised learning, feature engineering, and data modeling.

Neighbor-based approaches such as KDE are some of the most popular and useful density estimation techniques. KDE can be performed in any number of dimensions, though in practice high dimensionality can cause a degradation of performance. The KDE Modeling node and the KDE Simulation node in Watson Studio expose the core features and commonly used parameters of the KDE library. The nodes are implemented in Python. <sup>1</sup>

To use a KDE node, you must set up an upstream Type node. The KDE node will read input values from the Type node (or from the Types of an upstream import node).

The KDE Modeling node is available under the Modeling node palette. The KDE Modeling node generates a model nugget, and the nugget's scored values are kernel density values from the input data.

The KDE Simulation node is available under the Outputs node palette. The KDE Simulation node generates a KDE Gen source node that can create some records that have the same distribution as the input data. In the KDE Gen node properties, you can specify how many records the node will create (default is 1) and generate a random seed.

For more information about KDE, including examples, see the KDE documentation available at <http://scikit-learn.org/stable/modules/density.html#kernel-density-estimation>. <sup>1</sup>

<sup>1</sup> "User Guide." *Kernel Density Estimation*. Web. © 2007-2018, scikit-learn developers.

**Parent topic:** [Outputs](#)

## Extension Output node

---

With the Extension Output node, you can run R scripts or Python for Spark scripts to produce output.

After adding the node to your canvas, double-click the node to open its properties.

### Syntax tab

---

Select your type of syntax - R or Python for Spark. Then enter or paste your custom script for outputting data. When your syntax is ready, you can run the node. The following options are available for R syntax:

- **Convert flag fields.** Specifies how flag fields are treated. There are two options: Strings to factor, Integers and Reals to double, and Logical values (True, False). If you select Logical values (True, False) the original values of the flag fields are lost. For example, if a field has values `Male` and `Female`, these are changed to `True` and `False`.
- **Convert missing values to the R 'not available' value (NA).** When selected, any missing values are converted to the R `NA` value. The value `NA` is used by R to identify missing values. Some R functions that you use might have an argument that can control how the function behaves when the data contains `NA`. For example, the function might allow you to choose to automatically exclude records that contain `NA`. If this option isn't selected, any missing values are passed to R unchanged, and might cause errors when your R script runs.
- **Convert date/time fields to R classes with special control for time zones** When selected, variables with date or datetime formats are converted to R date/time objects. You must select one of the following options:
  - **R `POSIXct`.** Variables with date or datetime formats are converted to R `POSIXct` objects.

- **R POSIXlt (list).** Variables with date or datetime formats are converted to R `POSIXlt` objects.

Note: The POSIX formats are advanced options. Use these options only if your R script specifies that datetime fields are treated in ways that require these formats. The POSIX formats don't apply to variables with time formats.

- 

## Console Output tab

---

The Console Output tab contains any output that's received when the R script or Python for Spark script runs (for example, if using an R script, it shows output received from the R console when the R script in the R Syntax field on the Syntax tab is executed). This output might include R or Python error messages or warnings that are produced when the R script or Python script is executed. The output can be used, primarily, to debug the script. The Console Output tab also contains the script from the R Syntax or Python Syntax field.

Every time the Extension Import script runs, the content of the Console Output tab is overwritten with the output received from the R console or Python for Spark. You can't edit the output.

Note: R or Python for Spark error messages or warnings that result from running your Extension Output script are always displayed on the Console Output tab.

**Parent topic:** [Outputs](#)

## Export

---

Export nodes provide a mechanism for exporting data in various formats to interface with your other software tools.

- **Data Asset Export node**  
You can use the Data Asset Export node to write to remote data sources using connections, write to a data file on your local computer (.csv, .txt, or .sav), or write data to a project (delimited or .sav).
- **Extension Export node**  
You can use the Extension Export node to run R scripts or Python for Spark scripts to export data.

**Parent topic:** [Nodes palette](#)

## Data Asset Export node

---

You can use the Data Asset Export node to write to remote data sources using connections, write to a data file on your local computer (.csv, .txt, or .sav), or write data to a project (delimited or .sav).

Double-click the node to open its properties. Various options are available as described below.

If the data already exists, specify whether to replace it, append the output to the existing data set, or stop with an error.

After running the node, you can find the data at the export location you specified.

## Exporting to a project

---

Under Export to, select This project and then select the project path. For File type, select either Delimited or SAV.

## Exporting to a connection

---

Under Export to, select Save to a connection to open the Asset Browser and then select the connection to export to. See [Adding connections to projects](#) for details about creating connections.

Note: Export to OData or SAP OData isn't supported. You can only import data from OData or SAP OData.

## Exporting to a local file

---

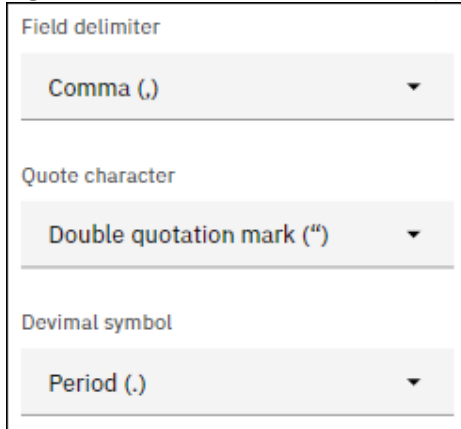
Under Export to, select Save to local computer to open a dialog and select a location on your computer.

## Setting the field delimiter, quote character, and decimal symbol

---

Different countries use different symbols to separate the integer part from the fractional part of a number and to separate fields in data. For example, you might use a comma instead of a period to separate the integer part from the fractional part of numbers. And, rather than using commas to separate fields in your data, you might use colons or tabs. With a Data Asset import or export node, you can specify these symbols for field delimiter and decimal. Double-click the node to open its properties and select the Field delimiter and the Decimal symbol as desired. Available delimiters are Comma, Tab, Colon, or Other. Select Other if you need to specify your own custom delimiter.

Figure 1. Field delimiter and decimal symbol options



The image shows a settings panel with three sections, each with a label and a dropdown menu. The first section is labeled 'Field delimiter' and has a dropdown menu showing 'Comma (,)'. The second section is labeled 'Quote character' and has a dropdown menu showing 'Double quotation mark (")'. The third section is labeled 'Devimal symbol' (note the typo) and has a dropdown menu showing 'Period (.)'.

## Exporting data to an SPSS Statistics data file

---

When you export data to an SPSS Statistics .sav file, the following additional options are available:

Export field names. The option you choose controls the method of handling variable names and labels upon export from SPSS Modeler to an SPSS Statistics .sav file.

- Names and variable labels. Select to export both SPSS Modeler field names and field labels. Names are exported as SPSS Statistics variable names, while labels are exported as SPSS Statistics variable labels.
- Names as variable labels. Select to use the SPSS Modeler field names as variable labels in SPSS Statistics. SPSS Modeler allows characters in field names that are not valid in SPSS Statistics variable names. To prevent possibly creating SPSS Statistics names that aren't valid, select Names as variable labels instead, or adjust field names.

## Exporting data to Planning Analytics (TM1)

---

Planning Analytics version 2.0.5 or higher is supported.

If you export to a Planning Analytics database connection, you select a *cube* (not a *view*) and the dimensions will be automatically mapped to fields by name. Before exporting to Planning Analytics, make sure the dimensions can be mapped to your data fields (rename or derive the required fields if needed). You'll encounter errors if the input schema doesn't exactly match the cube of the export schema. Since the target schema is fixed, renaming of derivation must be from the incoming data.

Important:

Note that you can only overwrite an existing data asset - you can't append to a data asset, stop with an error, do nothing, or create a new one as you can with other connection types. The data will be replaced completely. You must select the option Replace the data set when exporting to Planning Analytics.

You'll receive a `WDP Connection Error` in the following situations:

- Export to an unsupported version of Planning Analytics (2.0.5 or higher is required)
- Export to an existing cube that has a different schema than the incoming data
- Export to a nonexistent cube
- Export using any option under If the data set already exists other than Replace the data set.

**Parent topic:** [Export](#)

## Extension Export node

---

You can use the Extension Export node to run R scripts or Python for Spark scripts to export data.

### Syntax tab

---

Select your type of syntax - R or Python for Spark. Then enter or paste your custom scripting syntax. When your syntax is ready, you can run the node. The following options are available for R syntax:

- **Convert flag fields.** Specifies how flag fields are treated. There are two options: Strings to factor, Integers and Reals to double, and Logical values (True, False). If you select Logical values (True, False) the original values of the flag fields are lost. For example, if a field has values `Male` and `Female`, these are changed to `True` and `False`.
- **Convert missing values to the R 'not available' value (NA).** When selected, any missing values are converted to the R `NA` value. The value `NA` is used by R to identify missing values. Some R functions that you use might have an argument that can control how the function behaves when the data contains `NA`. For example, the function might allow you to choose to automatically exclude records that contain `NA`. If this option isn't selected, any missing values are passed to R unchanged, and might cause errors when your R script runs.
- **Convert date/time fields to R classes with special control for time zones** When selected, variables with date or datetime formats are converted to R date/time objects. You must select one of the following options:
  - **R `POSIXct`.** Variables with date or datetime formats are converted to R `POSIXct` objects.
  - **R `POSIXlt (list)`.** Variables with date or datetime formats are converted to R `POSIXlt` objects.

Note: The POSIX formats are advanced options. Use these options only if your R script specifies that datetime fields are treated in ways that require these formats. The POSIX formats don't apply to variables with time formats.

### Console Output tab

---

The Console Output tab contains any output that's received when the R script or Python for Spark script runs (for example, if using an R script, it shows output received from the R console when the R script in the R Syntax field on the Syntax tab is executed). This output might include R or Python error messages or warnings that are produced when the R script or Python script is executed. The output can be used, primarily, to debug the script. The Console Output tab also contains the script from the R Syntax or Python Syntax field.

Every time the Extension Export script runs, the content of the Console Output tab is overwritten with the output received from the R console or Python for Spark. You can't edit the output.

**Parent topic:** [Export](#)

## Working with your data

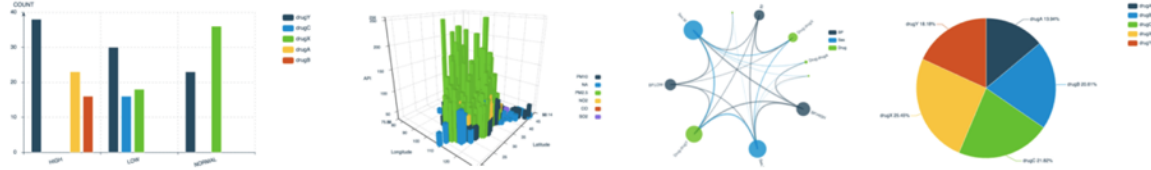
---

To see a quick sample of a flow's data, right-click a node and select Preview. To more thoroughly examine your data, use a Charts node to launch the chart builder.

With the chart builder, you can use advanced visualizations to explore your data from different perspectives and identify patterns, connections, and relationships within your data. You can also visualize your data with these same

charts in a Data Refinery flow.

Figure 1. Sample visualizations available for a flow



For more information, see [Visualizing your data](#).

- **Missing data values**

During the data preparation phase of data mining, you will often want to replace missing values in the data.

**Parent topic:** [Creating SPSS Modeler flows](#)

## Missing data values

During the data preparation phase of data mining, you will often want to replace missing values in the data.

*Missing values* are values in the data set that are unknown, uncollected, or incorrectly entered. Usually, such values aren't valid for their fields. For example, the field `Sex` should contain the values `M` and `F`. If you discover the values `Y` or `Z` in the field, you can safely assume that such values aren't valid and should therefore be interpreted as blanks. Likewise, a negative value for the field `Age` is meaningless and should also be interpreted as a blank. Frequently, such obviously wrong values are purposely entered, or fields left blank, during a questionnaire to indicate a nonresponse. At times, you may want to examine these blanks more closely to determine whether a nonresponse, such as the refusal to give one's age, is a factor in predicting a specific outcome.

Some modeling techniques handle missing data better than others. For example, the [C5.0 node](#) and the [Apriori node](#) cope well with values that are explicitly declared as "missing" in a [Type node](#). Other modeling techniques have trouble dealing with missing values and experience longer training times, resulting in less-accurate models.

There are several types of missing values recognized by SPSS Modeler:

- Null or system-missing values. These are nonstring values that have been left blank in the database or source file and have not been specifically defined as "missing" in an [Import](#) or [Type](#) node. System-missing values are displayed as `$null$`. Note that empty strings are not considered nulls in SPSS Modeler, although they may be treated as nulls by certain databases.
- Empty strings and white space. Empty string values and white space (strings with no visible characters) are treated as distinct from null values. Empty strings are treated as equivalent to white space for most purposes. For example, if you select the option to treat white space as blanks in an [Import](#) or [Type](#) node, this setting applies to empty strings as well.
- Blank or user-defined missing values. These are values such as `unknown`, `99`, or `-1` that are explicitly defined in an [Import](#) node or [Type](#) node as missing. Optionally, you can also choose to treat nulls and white space as blanks, which allows them to be flagged for special treatment and to be excluded from most calculations. For example, you can use the `@BLANK` function to treat these values, along with other types of missing values, as blanks.

**Reading in mixed data.** Note that when you're reading in fields with numeric storage (either integer, real, time, timestamp, or date), any non-numeric values are set to `null` or `system missing`. This is because, unlike some applications, SPSS Modeler doesn't allow mixed storage types within a field. To avoid this, you should read in any fields with mixed data as strings by changing the storage type in the [Import](#) node or external application as necessary.

**Reading empty strings from Oracle.** When reading from or writing to an Oracle database, be aware that, unlike SPSS Modeler and unlike most other databases, Oracle treats and stores empty string values as equivalent to null values. This means that the same data extracted from an Oracle database may behave differently than when extracted from a file or another database, and the data may return different results.

- **Handling missing values**  
You should decide how to treat missing values in light of your business or domain knowledge. To ease training time and increase accuracy, you may want to remove blanks from your data set. On the other hand, the presence of blank values may lead to new business opportunities or additional insights.
- **Functions available for missing values**  
Different methods are available for dealing with missing values in your data. You may choose to use functionality available in Data Refinery or in SPSS Modeler nodes.

**Parent topic:** [Working with your data](#)

## Handling missing values

---

You should decide how to treat missing values in light of your business or domain knowledge. To ease training time and increase accuracy, you may want to remove blanks from your data set. On the other hand, the presence of blank values may lead to new business opportunities or additional insights.

In choosing the best technique, you should consider the following aspects of your data:

- Size of the data set
- Number of fields containing blanks
- Amount of missing information

In general terms, there are two approaches you can follow:

- You can exclude fields or records with missing values
- You can impute, replace, or coerce missing values using a variety of methods

Both of these approaches can be largely automated using the [Data Audit node](#). For example, you can generate a [Filter node](#) that excludes fields with too many missing values to be useful in modeling, and generate a SuperNode that imputes missing values for any or all of the fields that remain. This is where the real power of the audit comes in, allowing you not only to assess the current state of your data, but to take action based on the assessment.

- **Handling records with missing values**  
If the majority of missing values are concentrated in a small number of records, you can just exclude those records. For example, a bank usually keeps detailed and complete records on its loan customers.
- **Handling fields with missing values**  
If the majority of missing values are concentrated in a small number of fields, you can address them at the field level rather than at the record level. This approach also allows you to experiment with the relative importance of particular fields before deciding on an approach for handling missing values. If a field is unimportant in modeling, it probably isn't worth keeping, regardless of how many missing values it has.
- **Handling records with system missing values**

**Parent topic:** [Missing data values](#)

## Handling records with missing values

---

If the majority of missing values are concentrated in a small number of records, you can just exclude those records. For example, a bank usually keeps detailed and complete records on its loan customers.

If, however, the bank is less restrictive in approving loans for its own staff members, data gathered for staff loans is likely to have several blank fields. In such a case, there are two options for handling these missing values:

- You can use a [Select node](#) to remove the staff records
- If the data set is large, you can discard all records with blanks

**Parent topic:** [Handling missing values](#)



## Handling fields with missing values

---

If the majority of missing values are concentrated in a small number of fields, you can address them at the field level rather than at the record level. This approach also allows you to experiment with the relative importance of particular fields before deciding on an approach for handling missing values. If a field is unimportant in modeling, it probably isn't worth keeping, regardless of how many missing values it has.

For example, a market research company may collect data from a general questionnaire containing 50 questions. Two of the questions address age and political persuasion, information that many people are reluctant to give. In this case, `Age` and `Political_persuasion` have many missing values.

### Field measurement level

---

In determining which method to use, you should also consider the measurement level of fields with missing values.

**Numeric fields.** For numeric field types, such as `Continuous`, you should always eliminate any non-numeric values before building a model, because many models won't function if blanks are included in numeric fields.

**Categorical fields.** For categorical fields, such as `Nominal` and `Flag`, altering missing values isn't necessary but will increase the accuracy of the model. For example, a model that uses the field `Sex` will still function with meaningless values, such as `Y` and `Z`, but removing all values other than `M` and `F` will increase the accuracy of the model.

### Screening or removing fields

---

To screen out fields with too many missing values, you have several options:

- You can use a [Data Audit node](#) to filter fields based on quality
- You can use a [Feature Selection node](#) to screen out fields with more than a specified percentage of missing values and to rank fields based on importance relative to a specified target
- Instead of removing the fields, you can use a [Type node](#) to set the field role to `None`. This will keep the fields in the data set but exclude them from the modeling processes

**Parent topic:** [Handling missing values](#)

## Handling records with system missing values

---

### What are system missing values?

---

System missing values represent data values that aren't known or not applicable. In databases, these values are often referred to as `NULL` values. System missing values are different from blank values. Blank values are typically defined in the `Type` node as particular values, or ranges of values, that can be regarded as user-defined-missing. Blank values are handled differently in the context of modeling.

### Constructing system missing values

---

System missing values might be present in data that's read from a data source (for example, database tables might contain `NULL` values). System missing values can be constructed by using the value `undef` in expressions. For example, the following CLEM expression returns the `Age`, if less than or equal to 30, or a missing value if greater than 30:

```
if Age > 30 then undef else Age endif
```

Missing values can also be created when an outer join is carried out, when a number is divided by zero, when the square root of a negative number is computed, and in other situations.

## Displaying system missing values

System missing values are displayed in tables and other output as `$null$`.

## Testing for system missing values

Use the special function `@NULL` to return `true` if the argument value is a system missing value. For example:

```
if @NULL(MyFieldName) then 'It is null' else 'It is not null' endif
```

## System missing values passed to functions

System missing values that are passed to functions usually propagate missing values to the output. For example, if the value of field `f1` is a system missing value in a particular row, then the expression `log(f1)` also evaluates to a system missing value for that row. An exception is the `@NULL` function.

## System missing values in expressions that involve arithmetic operators

Applying arithmetic operators to values that include a system missing value results in a system missing value. For example, if the value of field `f1` is a system missing value in a particular row, then the expression `f1 + 10` also evaluates to a system missing value for that row.

## System missing values in expressions that involve logical operators

When you work with system missing values in expressions that involve logical operators, the rules of three-valued logic (*true*, *false*, and *missing*) apply and can be described in truth tables. The truth tables for the common logical operators of *not*, *and*, and *or* are shown in the following tables.

Table 1. Truth table  
for NOT

Operand	NOT Operand
true	false
false	true
missing	missing

Table 2. Truth table for AND

Operand1	Operand2	Operand1 AND Operand2
true	true	true
true	false	false
true	missing	missing
false	true	false
false	false	false
false	missing	false
missing	true	missing
missing	false	false
missing	missing	missing

Table 3. Truth table for OR

Operand1	Operand2	Operand1 OR Operand2
true	true	true
true	false	true
true	missing	true
false	true	true
false	false	false

Operand1	Operand2	Operand1 OR Operand2
false	missing	missing
missing	true	true
missing	false	missing
missing	missing	missing

## System missing values in expressions that involve comparison operators

When you compare a system missing value and a non-system-missing value, the outcome evaluates to a system missing value rather than a true or false result. System missing values can be compared with each other; two system missing values are considered to be equal.

## System missing values in if/then/else/endif expressions

When you use conditional expressions, and the conditional expression returns a system missing value, the value from the `else` clause is returned from the conditional expression.

## System missing values in the Select node

When, for a particular record, the selection expression evaluates to a missing value, the record is not output from the Select node (this action applies to both Include and Discard modes).

## System missing values in the Merge node

When you merge by using a key, any records that have system missing values in a key field are not merged.

## System missing values in aggregation

When aggregating data on columns, missing values are not included in the calculation. For example, in a column with three values { 1, 2, and `undef` }, the sum of the values in the column is computed as 3; the mean value is computed as 1.5.

**Parent topic:** [Handling missing values](#)

## Functions available for missing values

Different methods are available for dealing with missing values in your data. You may choose to use functionality available in Data Refinery or in SPSS Modeler nodes.

## Functions available in SPSS Modeler

In SPSS Modeler, there are several functions used to handle missing values. The following functions are often used in Select and Filler nodes to discard or fill missing values:

- `count_nulls (LIST`
- `@BLANK (FIELD`
- `@NULL (FIELD`
- `undef`

You can use @ functions in conjunction with the `@FIELD` function to identify the presence of blank or null values in one or more fields. Simply flag the fields when blank or null values are present, or fill them with replacement values or use them in a variety of other operations.

You can count nulls across a list of fields, as follows:

```
count_nulls(['cardtenure' 'card2tenure' 'card3tenure'])
```

When using any of the functions that accept a list of fields as input, you can use the special functions `@FIELDS_BETWEEN` and `@FIELDS_MATCHING`, as shown in the following example:

```
count_nulls(@FIELDS_MATCHING('card*'))
```

You can use the `undef` function to fill fields with the system-missing value, displayed as `$null$`. For example, to replace any numeric value, you can use a conditional statement, such as:

```
if not(Age > 17) or not(Age < 66) then undef else Age endif
```

This replaces anything that isn't in the range with a system-missing value, displayed as `$null$`. By using the `not()` function, you can catch all other numeric values, including any negatives.

Note on discarding records: When using a Select node to discard records, note that syntax uses three-valued logic and automatically includes null values in select statements. To exclude null values (system-missing) in a select expression, you must explicitly specify this by using `and not` in the expression. For example, to select and include all records where the type of prescription drug is `Drug C`, you would use the following select statement:

```
Drug = 'drugC' and not(@NULL(Drug))
```

## Functions available in Data Refinery

---

You can also use Data Refinery to handle missing values. See the following information.

- [GUI operations in Data Refinery](#)
- [Interactive code templates in Data Refinery](#)

**Parent topic:** [Missing data values](#)

## Saving and running models on Watson Machine Learning Server

---

With Watson Studio Desktop, you can run your models on a Watson Machine Learning Server for better performance.

### To save a model to the Watson Machine Learning Server

---

1. With your SPSS Modeler flow open, click the Save icon on the toolbar or right-click a node and select Save branch as model. If you haven't yet created or connected to a deployment space, you'll be prompted to do so. For more information about deployment spaces, contact your administrator and see [Deployment spaces](#).
2. Select the desired options and click Save to save the model to the deployment space associated with your flow. You can also save the model as PMML.

PMML, or predictive model markup language, is an XML format for describing data mining and statistical models, including inputs to the models, transformations used to prepare data for data mining, and the parameters that define the models themselves. You can save models as PMML, making it possible to share models with other applications that support this format. For more information about PMML, see the [Data Mining Group web site](#).

For more information about the Watson Machine Learning Server, contact your administrator and see [Connecting to IBM Watson Machine Learning Server](#).

### To run flows on the Watson Machine Learning Server

---

Use the drop-down on the toolbar to choose where to run your flow. By default, flows run on your local computer. If you need more processing power and you configured a connection to a Watson Machine Learning Server, you can run flows on a remote server instead. To add a new connection, go to Add-ons and services under the left navigation menu. The server connection(s) you add will then show up in the drop-down.

The run mode you select is used for all flows you run in your current Watson Studio session. For example, if you open flow ABC from project ABC and choose to run it on a Watson Machine Learning Server, then open flow XYZ from project XYZ, flow ABC will then also run on the server by default. Each time you restart the application, the run mode will be reset to run locally.

If the Watson Machine Learning Server you select isn't responding, there might be a problem with the server or the network connection. You can try again later, configure a new server connection, or revert to running on your local system. Note that if you get a `None.get` error, it usually means you've lost the connection with the server for some reason and you should restart Watson Studio or switch to a working server.

The following connections are supported when running SPSS Modeler flows on a Watson Machine Learning Server:

- Amazon Redshift
- Amazon S3
- Apache HDFS (formerly known as "Hortonworks HDFS")
- Apache Hive (*supports source connections only*)
- Analytics Engine HDFS (formerly known as "BigInsights HDFS")
- Cloud Object Storage
- Cloud Object Storage (infrastructure)
- Compose for MySQL
- Databases for PostgreSQL (formerly known as "Compose for PostgreSQL")
- Db2
- Db2 for z/OS
- Dropbox
- FTP (remote file system transfer)
- Informix
- Microsoft SQL Server
- MinIO
- Sybase (*supports source connections only*)
- Salesforce.com (*supports source connections only*)

## To deploy a model

---

In the left navigation menu, under Deployment Spaces, go to Assets and then select Deploy from the actions menu to deploy a model that you saved previously. Models you deploy will be listed under Assets, and you can copy them to other deployment spaces if desired. See [Deploying assets](#) for more information.

**Parent topic:** [Creating SPSS Modeler flows](#)

## Flow scripting

---

Scripts can be used to customize operations within a particular flow, and they are saved with that flow. For example, you might use a script to specify a particular run order for the terminal nodes within a flow. You use the flow properties panel to edit the script that is saved with the current flow.

To access scripting in a flow's properties:

1. Right-click in your flow's canvas and select Flow properties.
2. Click the Scripting tab to work with scripts for the current flow.

**Note:** By default, the Python scripting language is used. If you'd rather use a scripting language unique to old versions of SPSS Modeler, select Legacy.

You can specify whether or not the script runs when the flow runs. To run the script each time the flow runs, respecting the run order of the script, select Run the script. This setting provides automation at the flow level for quicker model building. Or, to ignore the script, you can select the option to only Run all terminal nodes when the flow runs.

The script editor includes the following features that help with script authoring:

- Syntax highlighting; keywords, literal values (such as strings and numbers), and comments are highlighted.
- Line numbering.
- Block matching; when the cursor is placed by the start of a program block, the corresponding end block is also highlighted.
- Suggested auto-completion.

A list of suggested syntax completions can be accessed by selecting Auto-Suggest from the context menu, or pressing Ctrl + Space. Use the cursor keys to move up and down the list, then press Enter to insert the selected text. To exit from auto-suggest mode without modifying the existing text, press Esc.

- **Flow scripting example**

You can use a flow to train a model when it runs. Normally, to test the model, you might run the modeling node to add the model to the flow, make the appropriate connections, and run an Analysis node.

- **Flow and SuperNode parameters**

You can define parameters for use in CLEM expressions and in scripting. They are, in effect, user-defined variables that are saved and persisted with the current flow or SuperNode and can be accessed from the user interface as well as through scripting.

**Parent topic:** [Creating SPSS Modeler flows](#)

## Flow scripting example

---

You can use a flow to train a model when it runs. Normally, to test the model, you might run the modeling node to add the model to the flow, make the appropriate connections, and run an Analysis node.

Using a script, you can automate the process of testing the model nugget after you create it. For example, you might use a script such as the following to train a neural network model:

```
stream = modeler.script.stream()
neuralnetnode = stream.findByType("neuralnetwork", None)
results = []
neuralnetnode.run(results)
appliernode = stream.createModelApplierAt(results[0], "Drug", 594, 187)
analysisnode = stream.createAt("analysis", "Drug", 688, 187)
typenode = stream.findByType("type", None)
stream.linkBetween(appliernode, typenode, analysisnode)
analysisnode.run([])
```

The following bullets describe each line in this script example.

- The first line defines a variable that points to the current flow.
- In line 2, the script finds the Neural Net builder node.
- In line 3, the script creates a list where the execution results can be stored.
- In line 4, the Neural Net model nugget is created. This is stored in the list defined on line 3.
- In line 5, a model apply node is created for the model nugget and placed on the flow canvas.
- In line 6, an analysis node called `Drug` is created.
- In line 7, the script finds the Type node.
- In line 8, the script connects the model apply node created in line 5 between the Type node and the Analysis node.
- Finally, the Analysis node runs to produce the Analysis report.

It's possible to use a script to build and run a flow from scratch, starting with a blank canvas.

**Parent topic:** [Flow scripting](#)

## Flow and SuperNode parameters

---

You can define parameters for use in CLEM expressions and in scripting. They are, in effect, user-defined variables that are saved and persisted with the current flow or SuperNode and can be accessed from the user interface as well as through scripting.

If you save a flow, for example, any parameters you set for that flow are also saved. (This distinguishes them from local script variables, which can be used only in the script in which they are declared.) Parameters are often used in scripting to control the behavior of the script, by providing information about fields and values that don't need to be hard coded in the script.

You can set flow parameters in a flow script or in a flow's properties (right-click the canvas in your flow and select Flow Properties), and they're available to all nodes in the flow. They're displayed in the Parameters list in the Expression Builder.

You can also set parameters for SuperNodes, in which case they're visible only to nodes encapsulated within that SuperNode.

## Using parameters in CLEM expressions

Parameters are represented in CLEM expressions by `$P-pname`, where `pname` is the name of the parameter. When used in CLEM expressions, parameters must be placed within single quotes; for example, `'$P-scale'`.

**Parent topic:** [Flow scripting](#)

## SQL optimization

You can push many data preparation and mining operations directly in your database to improve performance.

One of the most powerful capabilities of SPSS Modeler is the ability to perform many data preparation and mining operations directly in the database. By generating SQL code that can be pushed back to the database for execution, many operations, such as sampling, sorting, deriving new fields, and certain types of graphing, can be performed in the database rather than on the client or server computer. When you're working with large datasets, these *pushbacks* can dramatically enhance performance in several ways:

- By reducing the size of the result set to be transferred from the DBMS to Watson Studio. When large result sets are read through an ODBC driver, network I/O or driver inefficiencies may result. For this reason, the operations that benefit most from SQL optimization are row and column selection and aggregation (Select, Sample, Aggregate nodes), which typically reduce the size of the dataset to be transferred. Data can also be cached to a temporary table in the database at critical points in the flow (after a Merge or Select node, for example) to further improve performance.
- By making use of the performance and scalability of the database. Efficiency is increased because a DBMS can often take advantage of parallel processing, more powerful hardware, more sophisticated management of disk storage, and the presence of indexes.

Given these advantages, Watson Studio is designed to maximize the amount of SQL generated by each SPSS Modeler flow so that only those operations that can't be compiled to SQL are executed by Watson Studio. Because of limitations in what can be expressed in standard SQL (SQL-92), however, certain operations may not be supported.

The following databases are currently supported across various product offerings:

Table 1. Databases supporting SQL pushback on Windows

Database	Database version	SQL pushback in Watson Studio Cloud?	SQL pushback in Cloud Pak for Data?	SQL pushback in Watson Studio Desktop?	SQL pushback in Watson Machine Learning Server?
Amazon Redshift	All versions and future fix packs	Yes	Yes	Yes	Yes

Database	Database version	SQL pushback in Watson Studio Cloud?	SQL pushback in Cloud Pak for Data?	SQL pushback in Watson Studio Desktop?	SQL pushback in Watson Machine Learning Server?
Cloudera Impala	2.8	No	No	Yes	Yes
Databases for PostgreSQL	10.0 and future fix packs	Yes	Yes	Yes	Yes
Db2	11.1.0	Yes	Yes	Yes	Yes
Google BigQuery	Latest	No	No	Yes	No
Microsoft SQL Server	2016	Yes	Yes	Yes	Yes
Netezza	7.2	Yes	Yes	Yes	Yes
Oracle	12c	Yes	Yes	Yes	Yes
Teradata	16.2	No	No	Yes	No

For **Watson Studio Desktop**, you must install the SPSS Data Access Pack to enable SQL pushback support for the following databases. See the **SPSS Data Access Pack** section below for more information.

- Cloudera Impala
- Oracle
- Microsoft SQL Server
- Amazon Redshift

Also, to enable SQL pushback for the following databases on **Watson Studio Desktop**, you must install their native ODBC drivers:

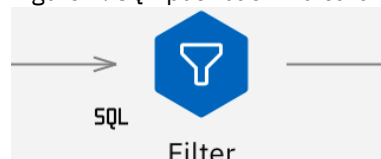
- Db2  
You must install the IBM Data Server Driver Package available [here](#).
- Netezza  
You must install the Netezza client, documented [here](#), which includes the Netezza ODBC driver.
- Teradata  
An additional ODBC driver is required; install the latest [Teradata Tools and Utilities](#).
- Google BigQuery  
You must install the required ODBC driver and then configure the JSON `KeyFilePath`. For Windows, you can install the latest ODBC driver from [cloud.google.com](#) (this version is free) or from [simba.com](#). For macOS, install the latest ODBC driver from [simba.com](#).  
Note that if you use the contents of a key file to define your BigQuery connection in the project, SQL pushback isn't supported. To enable SQL pushback, you must instead save the key file directly to your local computer and point to the absolute path in your connection definition.

macOS only supports Db2 and Google BigQuery (you must install the required native drivers).

Tips:

- When running a flow, nodes that push back to your database are highlighted with a small SQL icon beside the node. When you start making edits to a flow after running it, the icons will be removed until the next time you run the flow.

Figure 1. SQL pushback indicator





- If a node can't be pushed back, all subsequent nodes in the flow won't be pushed back either (pushback stops at that node). This may impact how you want to organize the order of nodes in your flow.
- You don't need to define an ODBC data source to use your drivers with Watson Studio Desktop. Instead, create a connection in your project. See [Adding connections to projects](#) for more info.

Notes: Keep the following information in mind regarding SQL:

- Because of minor differences in SQL implementation, flows that run in a database may return slightly different results when executed in Watson Studio. These differences may also vary depending on the database vendor, for similar reasons. For example, depending on the database configuration for case sensitivity in string comparison and string collation, SPSS Modeler flows that run using SQL pushback may produce different results from those that run without SQL pushback. Contact your database administrator for advice on configuring your database. To maximize compatibility with Watson Studio, database string comparisons should be case sensitive.
- When using Watson Studio to generate SQL, it's possible the result using SQL pushback is not consistent on some platforms (Linux, for example). This is because floating point is handled differently on different platforms.

## SPSS Data Access Pack

---

This section only applies to Watson Studio Desktop.

On **Windows**, you can obtain pushback to SQL Server and other non-Db2 databases if you install SPSS Data Access Pack version 7.1.2, which is available on [IBM Passport Advantage](#). The part number is CNKK9EN.

For SPSS Data Access Pack installation instructions, see [here](#). Note that some of the information in the PDF doesn't apply to Watson Studio.

- **How does SQL pushback work?**  
The initial fragments of a flow leading from the data import nodes are the main targets for SQL generation. When a node is encountered that can't be compiled to SQL, the data is extracted from the database and subsequent processing is performed.
- **Tips for maximizing SQL pushback**  
To get the best performance boost from SQL optimization, pay attention to the items in this section.
- **Nodes supporting SQL pushback**  
The tables in this section show nodes representing data-mining operations that support SQL pushback. If a node doesn't appear in these tables, it doesn't support SQL pushback.
- **CLEM expressions and operators supporting SQL pushback**  
The tables in this section list the mathematical operations and expressions that support SQL generation and are often used during data mining. Operations absent from these tables don't support SQL generation.
- **Generating SQL from model nuggets**  
When using data from a database, SQL code can be pushed back to the database for execution, providing superior performance for many operations. For some nodes, SQL for the model nugget can be generated, pushing back the model scoring stage to the database. This allows flows containing these nuggets to have their full SQL pushed back.

**Parent topic:** [Creating SPSS Modeler flows](#)

## How does SQL pushback work?

---

The initial fragments of a flow leading from the data import nodes are the main targets for SQL generation. When a node is encountered that can't be compiled to SQL, the data is extracted from the database and subsequent processing is performed.

During flow preparation and prior to running, the SQL generation process happens as follows:

- The software reorders flows to move downstream nodes into the "SQL zone" where it can be proven safe to do so.

- Working from the import nodes toward the terminal nodes, SQL expressions are constructed incrementally. This phase stops when a node is encountered that can't be converted to SQL or when the terminal node (for example, a Table node or a Graph node) is converted to SQL. At the end of this phase, each node is labeled with an SQL statement if the node and its predecessors have an SQL equivalent.
- Working from the nodes with the most complicated SQL equivalents back toward the import nodes, the SQL is checked for validity. The SQL that was successfully validated is chosen for execution.
- Nodes for which all operations have generated SQL are highlighted with a SQL icon beside the node on the flow canvas. Based on the results, you may want to further reorganize your flow where appropriate to take full advantage of database execution.

## Where do improvements occur?

---

SQL pushback improves performance in a number of data operations:

- Joins (merge by key). Join operations can increase optimization within databases.
- Aggregation. The Aggregate, Distribution, and Web nodes all use aggregation to produce their results. Summarized data uses considerably less bandwidth than the original data.
- Selection. Choosing records based on certain criteria reduces the quantity of records.
- Sorting. Sorting records is a resource-intensive activity that is performed more efficiently in a database.
- Field derivation. New fields are generated more efficiently in a database.
- Field projection. The software extracts only fields that are required for subsequent processing from the database, which minimizes bandwidth and memory requirements. The same is also true for superfluous fields in flat files: although the software must read the superfluous fields, it doesn't allocate any storage for them.
- Scoring. SQL can be generated from decision trees, rulesets, linear regression, and factor-generated models.

**Parent topic:** [SQL optimization](#)

## Tips for maximizing SQL pushback

---

To get the best performance boost from SQL optimization, pay attention to the items in this section.

**Flow order.** SQL generation may be halted when the function of the node has no semantic equivalent in SQL because SPSS Modeler's data-mining functionality is richer than the traditional data-processing operations supported by standard SQL. When this happens, SQL generation is also suppressed for any downstream nodes. Therefore, you may be able to significantly improve performance by reordering nodes to put operations that halt SQL as far downstream as possible. The SQL optimizer can do a certain amount of reordering automatically, but further improvements may be possible. A good candidate for this is the Select node, which can often be brought forward. See [Nodes supporting SQL pushback](#) for more information.

**CLEM expressions.** If a flow can't be reordered, you may be able to change node options or CLEM expressions or otherwise recast the way the operation is performed, so that it no longer inhibits SQL generation. Derive, Select, and similar nodes can commonly be rendered into SQL, provided that all of the CLEM expression operators have SQL equivalents. Most operators can be rendered, but there are a number of operators that inhibit SQL generation (in particular, the sequence functions ["@ functions"]). Sometimes generation is halted because the generated query has become too complex for the database to handle. See [CLEM expressions and operators supporting SQL pushback](#) for more information.

**Multiple input nodes.** Where a flow has multiple data import nodes, SQL generation is applied to each import branch independently. If generation is halted on one branch, it can continue on another. Where two branches merge (and both branches can be expressed in SQL up to the merge), the merge itself can often be replaced with a database join, and generation can be continued downstream.

**Scoring models.** In-database scoring is supported for some models by rendering the generated model into SQL. However, some models generate extremely complex SQL expressions that aren't always evaluated effectively within the database. For this reason, SQL generation must be enabled separately for each generated model nugget. If you find that a model nugget is inhibiting SQL generation, open the model nugget's settings and select Generate SQL for

this model (with some models, you may have additional options controlling generation). Run tests to confirm that the option is beneficial for your application. See [Nodes supporting SQL pushback](#) for more information.

When testing modeling nodes to see if SQL generation for models works effectively, we recommend first saving all flows from SPSS Modeler. Note that some database systems may hang while trying to process the (potentially complex) generated SQL.

Database caching. If you are using a node cache to save data at critical points in the flow (for example, following a Merge or Aggregate node), make sure that database caching is enabled along with SQL optimization. This will allow data to be cached to a temporary table in the database (rather than the file system) in most cases.

Vendor-specific SQL. Most of the generated SQL is standards-conforming (SQL-92), but some nonstandard, vendor-specific features are exploited where practical. The degree of SQL optimization can vary, depending on the database source.

**Parent topic:** [SQL optimization](#)

## Nodes supporting SQL pushback

The tables in this section show nodes representing data-mining operations that support SQL pushback. If a node doesn't appear in these tables, it doesn't support SQL pushback.

Table 1. Record Operations nodes

Nodes supporting SQL generation	Notes
Select	Supports generation only if SQL generation for the select expression itself is supported (see expressions below). If any fields have nulls, SQL generation does not give the same results for discard as are given in native SPSS Modeler.
Sample	Simple sampling supports SQL generation to varying degrees depending on the database.
Aggregate	SQL generation support for aggregation depends on the data storage type.
RFM Aggregate	Supports generation except if saving the date of the second or third most recent transactions, or if only including recent transactions. However, including recent transactions does work if the <code>datetime_date(YEAR, MONTH, DAY)</code> function is pushed back.
Sort	
Merge	<p>No SQL generated for merge by order.</p> <p>Merge by key with full or partial outer join is only supported if the database/driver supports it. Non-matching input fields can be renamed by means of a Filter node, or the Filter settings of an import node.</p> <p>Supports SQL generation for merge by condition.</p> <p>For all types of merge, <code>SQL_SP_EXISTS</code> is not supported if inputs originate in different databases.</p>
Append	Supports generation if inputs are unsorted. SQL optimization is only possible when your inputs have the same number of columns.
Distinct	A Distinct node with the (default) mode Create a composite record for each group selected doesn't support SQL optimization.

Table 2. SQL generation support in the Sample node for simple sampling

Mode	Sample	Max size	Seed	Db2 for z/OS	Db2 for OS/400	Db2 for Win/UNIX	Oracle	SQL Server	Teradata
Include	First	n/a		Y	Y	Y	Y	Y	Y
	1-in-n	off		Y	Y	Y	Y		Y
		max		Y	Y	Y	Y		Y
	Random %	off	off	Y		Y	Y		Y

Mode	Sample	Max size	Seed	Db2 for z/OS	Db2 for OS/400	Db2 for Win/UNIX	Oracle	SQL Server	Teradata
			on	Y		Y	Y		
		max	off	Y		Y	Y		Y
			on	Y		Y	Y		
Discard	First	off					Y		
		max					Y		
	1-in-n	off		Y	Y	Y	Y		Y
		max		Y	Y	Y	Y		Y
	Random %	off	off	Y		Y	Y		Y
			on	Y		Y	Y		
		max	off	Y		Y	Y		Y
			on	Y		Y	Y		

Table 3. SQL generation support in the Aggregate node

Storage	Sum	Mean	Min	Max	SDev	Median	Count	Variance	Percentile
Integer	Y	Y	Y	Y	Y	Y*	Y	Y	Y*
Real	Y	Y	Y	Y	Y	Y*	Y	Y	Y*
Date			Y	Y		Y*	Y		Y*
Time			Y	Y		Y*	Y		Y*
Timestamp			Y	Y		Y*	Y		Y*
String			Y	Y		Y*	Y		Y*

\* Median and Percentile are supported on Oracle.

Table 4. Field Operations nodes

Node s suppo rting SQL gener ation	Notes
Type	Supports SQL generation if the Type node is instantiated and no <b>ABORT</b> or <b>WARN</b> type checking is specified.
Filter	
Derive	Supports SQL generation if SQL generated for the derive expression is supported (see expressions below).
Ensemble	Supports SQL generation for Continuous targets. For other targets, supports generation only if the Highest confidence wins ensemble method is used.
Filler	Supports SQL generation if the SQL generated for the derive expression is supported (see expressions below).
Anonymize	Supports SQL generation for Continuous targets, and partial SQL generation for Nominal and Flag targets.
Reclassify	
Binning	Supports SQL generation if the Tiles (equal count) binning method is used and the Read from Bin Values tab if available option is selected. Due to differences in the way that bin boundaries are calculated (this is caused by the nature of the distribution of data in bin fields), you might see differences in the binning output when comparing normal flow execution results and SQL pushback results. To avoid this, use the Record count tiling method, and either Add to next or Keep in current tiles to obtain the closest match between the two methods of flow execution.
RFM Analysis	Supports SQL generation if the Read from Bin Values tab if available option is selected, but downstream nodes will not support it.

<b>Nodes supporting SQL generation</b>	<b>Notes</b>
Partition	Supports SQL generation to assign records to partitions.
Set To Flag	
Restructure	

Table 5. Graphs nodes

<b>Nodes supporting SQL generation</b>	<b>Notes</b>
Distribution	
Web	
Evaluation	

For some models, SQL for the model nugget can be generated, pushing back the model scoring stage to the database. The main use of this feature is not to improve performance, but to allow flows containing these nuggets to have their full SQL pushed back. See [Generating SQL from model nuggets](#) for more information.

Table 6. Model nuggets

<b>Model nuggets supporting SQL generation</b>	<b>Notes</b>
C&R Tree	Supports SQL generation for the single tree option, but not for the boosting, bagging, or large dataset options.
QUEST	
CHAID	
C5.0	
Decision List	
Linear	Supports SQL generation for the standard model option, but not for the boosting, bagging, or large dataset options.
Neural Net	Supports SQL generation for the standard model option (Multilayer Perceptron only), but not for the boosting, bagging, or large dataset options.
PCA/Factor	
Logistic	Supports SQL generation for Multinomial procedure but not Binomial. For Multinomial, generation isn't supported when confidences are selected, unless the target type is Flag.
Generated Rulesets	
Auto Classifier	If a User Defined Function (UDF) scoring adapter is enabled, these nuggets support SQL pushback. Also, if either SQL generation for Continuous targets, or the Highest confidence wins ensemble method are used, these nuggets support further pushback downstream.
Auto Numeric	

Table 7. Outputs nodes

<b>Nodes supporting SQL generation</b>	<b>Notes</b>
Table	Supports generation if SQL generation is supported for highlight expression (see expressions below).
Matrix	Supports generation except if All numerics is selected for the Fields option.
Analysis	Supports generation, depending on the options selected.

Nodes supporting SQL generation	Notes
Transform	
Statistics	Supports generation if the Correlate option isn't used.
Report	
Set Globals	

Parent topic: [SQL optimization](#)

## CLEM expressions and operators supporting SQL pushback

The tables in this section list the mathematical operations and expressions that support SQL generation and are often used during data mining. Operations absent from these tables don't support SQL generation.

Table 1. Operators

Operations supporting SQL generation	Notes
+	
-	
/	
*	
><	Used to concatenate strings.

Table 2. Relational operators

Operations supporting SQL generation	Notes
=	
/=	Used to specify "not equal."
>	
>=	
<	
<=	

Table 3. Functions

Operations supporting SQL generation	Notes
abs	
allbutfirst	
allbutlast	
and	
arccos	
arcsin	
arctan	
arctanh	
cos	
div	
exp	
fracof	
hasstartstring	
hassubstring	
integer	
intof	
isaplha code	
islowercode	
isnumbercode	

Operations supporting SQL generation	Notes
isstartstring	
issubstring	
isuppercode	
last	
length	
locchar	
log	
log10	
lowertoupper	
max	
member	
min	
negate	
not	
number	
or	
pi	
real	
rem	
round	
sign	
sin	
sqrt	
string	
strmember	
subscrs	
substring	
substring_between	
uppertolower	
to_string	

Table 4. Special functions

Operations supporting SQL generation	Notes
@NULL	
@GLOBAL_AVE	You can use the special global functions to retrieve global values computed by the Set Globals node.
@GLOBAL_SUM	
@GLOBAL_MAX	
@GLOBAL_MEAN	
@GLOBAL_MIN	
@GLOBALSDEV	

Table 5. Aggregate functions

Operations supporting SQL generation	Notes
Sum	
Mean	
Min	
Max	

Operations supporting SQL generation	Notes
Count	
SDev	

**Parent topic:** [SQL optimization](#)

## Generating SQL from model nuggets

When using data from a database, SQL code can be pushed back to the database for execution, providing superior performance for many operations. For some nodes, SQL for the model nugget can be generated, pushing back the model scoring stage to the database. This allows flows containing these nuggets to have their full SQL pushed back.

For a generated model nugget that supports SQL pushback:

1. Double-click the model nugget to open its settings.
2. Depending on the node type, one or more of the following options is available. Choose one of these options to specify how SQL generation is performed.

Generate SQL for this model

- Default: Score using Server Scoring Adapter (if installed) otherwise in process. This is the default option. If connected to a database with a scoring adapter installed, this option generates SQL using the scoring adapter and associated user defined functions (UDF) and scores your model within the database. When no scoring adapter is available, this option fetches your data back from the database and scores it in SPSS Modeler.
- Score by converting to native SQL without Missing Value Support. This option generates native SQL to score the model within the database, without the overhead of handling missing values. This option simply sets the prediction to null (\$null\$) when a missing value is encountered while scoring a case.
- Score by converting to native SQL with Missing Value Support. For CHAID, QUEST, and C&R Tree models, you can generate native SQL to score the model within the database with full missing value support. This means that SQL is generated so that missing values are handled as specified in the model. For example, C&R Trees use surrogate rules and biggest child fallback.
- Score outside of the Database. This option fetches your data back from the database and scores it in SPSS Modeler.

**Parent topic:** [SQL optimization](#)

## Disabling or caching nodes in a flow

You can disable a node so it's ignored when the flow runs. And you can set up a cache on a node.

- **Disabling nodes in a flow**  
You can disable process nodes that have a single input so that they're ignored when the flow runs. This saves you from having to remove or bypass the node and means you can leave it connected to the remaining nodes.
- **Caching options for nodes**  
To optimize the running of flows, you can set up a cache on any nonterminal node. When you set up a cache on a node, the cache is filled with the data that passes through the node the next time you run the data flow. From then on, the data is read from the cache (which is stored temporarily) rather than from the data source.

**Parent topic:** [Creating SPSS Modeler flows](#)

## Disabling nodes in a flow

You can disable process nodes that have a single input so that they're ignored when the flow runs. This saves you from having to remove or bypass the node and means you can leave it connected to the remaining nodes.



You can still open and edit the node settings; however, any changes will not take effect until you enable the node again.

For example, you might use a Filter node to filter several fields, and then build models based on the reduced data set. If you want to also build the same models *without* fields being filtered, to see if they improve the model results, you can disable the Filter node. When you disable the Filter node, the connections to the modeling nodes pass directly through from the Derive node to the Type node.

## To disable a node

---

1. In your flow, right-click the node you want to disable.
2. Select Disable node. The node changes to a dark color.

When you want to include the node back in the flow, select Enable node in the same way.

**Parent topic:** [Disabling or caching nodes in a flow](#)

## Caching options for nodes

---

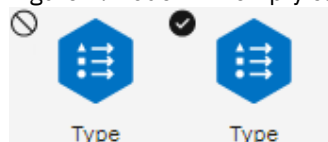
To optimize the running of flows, you can set up a cache on any nonterminal node. When you set up a cache on a node, the cache is filled with the data that passes through the node the next time you run the data flow. From then on, the data is read from the cache (which is stored temporarily) rather than from the data source.

Caching is most useful following a time-consuming operation such as a sort, merge, or aggregation. For example, suppose that you have an import node set to read sales data from a database and an Aggregate node that summarizes sales by location. You can set up a cache on the Aggregate node rather than on the import node because you want the cache to store the aggregated data rather than the entire data set.

Note: Caching at import nodes, which simply stores a copy of the original data as it is read into SPSS Modeler, won't improve performance in most circumstances.

Nodes with caching enabled are displayed with a special circle-backslash icon beside them. When the data is cached at the node, the icon changes to a check mark.

Figure 1. Node with empty cache vs. node with full cache



## To enable a cache

---

Right-click the node in your flow and select Cache > Enable. You can turn off the cache by right-clicking the node again and selecting Cache > Disable.

## Caching nodes in a database

---

For flows that run in a database, you can cache data mid-flow to a temporary table in the database rather than the file system. When combined with SQL optimization, this may result in significant gains in performance. For example, the output from a flow that merges multiple tables to create a data mining view may be cached and reused as needed. By automatically generating SQL for all downstream nodes, performance can be further improved.

To take advantage of database caching, both SQL optimization and database caching must be enabled.

With database caching enabled, simply right-click any nonterminal node to cache data at that point, and the cache will be created automatically directly in the database the next time the flow runs. If database caching or SQL optimization is not enabled, the cache will be written to the file system instead.

Note: The following databases support temporary tables for the purpose of caching: Db2, Oracle, SQL Server, and Teradata. Other databases, such as Netezza, will use a normal table for database caching.

## To flush a cache

---

A circle-backslash icon beside a node indicates that its cache is empty. When the cache is full, the icon becomes a check mark. If you want to replace the contents of the cache, you must first flush the cache and then re-run the data flow to refill it.

In your flow, right-click the node and select Cache > Flush.

**Parent topic:** [Disabling or caching nodes in a flow](#)

## Importing an SPSS Modeler stream

---

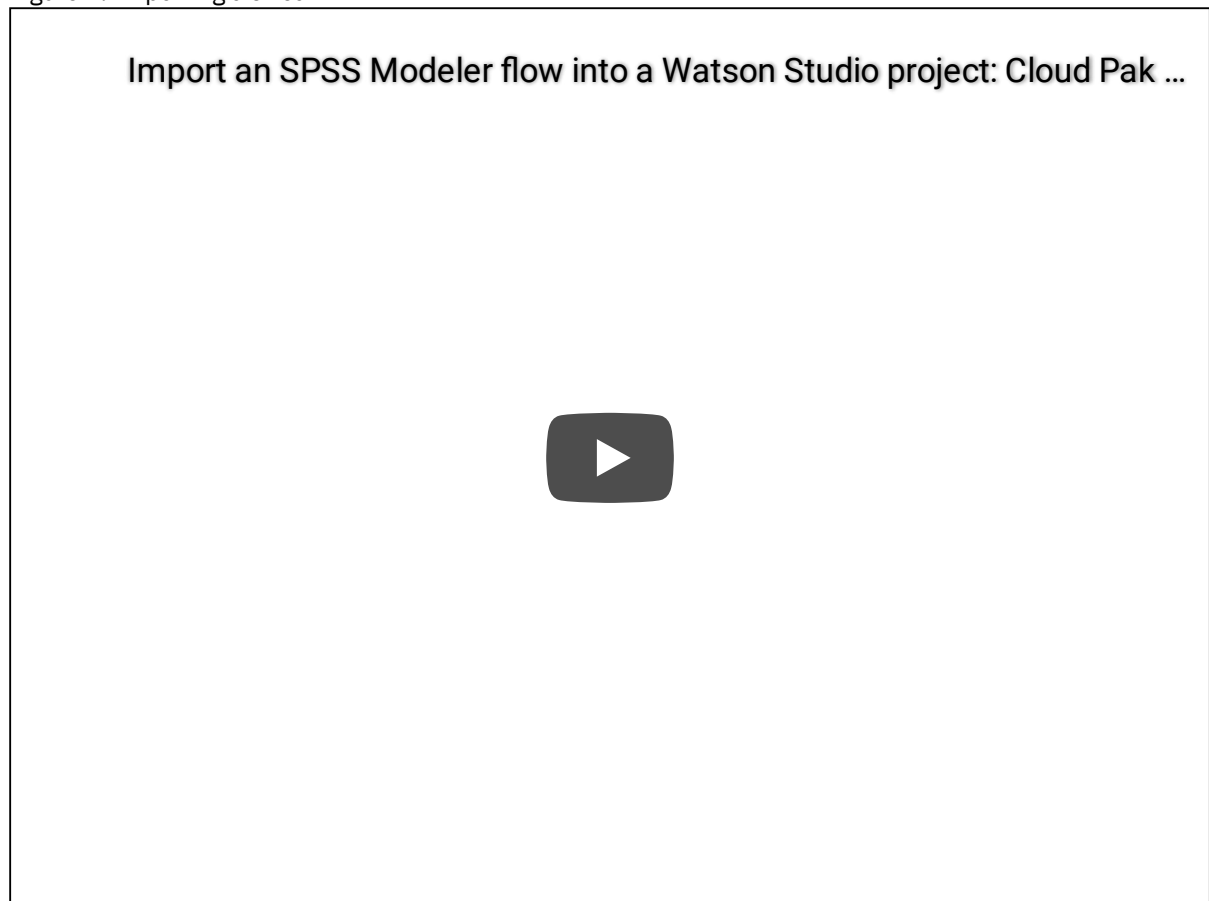
You can import a stream (.str) that was created in SPSS Modeler Subscription or SPSS Modeler client.

1. From your project, under Modeler flows, click New modeler flow.
2. Select From File, select the .str file you want to import, and click Create.

If the imported stream contains one or more source (import) or export nodes, you'll be prompted to convert the nodes. Watson Studio will walk you through the migration process.

Watch the following video for an example of this easy process:

Figure 1. Importing a stream



<https://youtube.com/embed/0GBMTaZYXWU>

If the stream contains multiple import nodes that use the same data file, then you must first add that file to your project as a data asset before migrating because the conversion can't upload the same file to more than one import node. After adding the data asset to your project, reopen the flow and proceed with the migration using the new data asset. Nodes with the same name will be automatically mapped to project assets.

Configure export nodes to export to your project or to a connection. The following export nodes are supported:

Table 1. Export nodes that can be migrated

Supported SPSS Modeler export nodes
Analytic Server
Database
Flat File
Statistics Export
Data Collection
Excel
IBM Cognos Analytics Export
TM1 Export
SAS
XML Export

Notes: Keep the following information in mind when migrating nodes.

- When migrating export nodes, you're converting node types that don't exist in Watson Studio. The nodes are converted to Data Asset export nodes or a connection. Due to a current limitation for automatically migrating nodes, only existing project assets or connections can be selected as export targets. These assets will be overwritten during export when the flow runs.
- To preserve any type or filter information, when an import node is replaced with Data Asset nodes, they're converted to a supernode.
- After migration, you can go back later and use the Convert button if you want to migrate a node that you skipped previously.
- If the stream you imported uses scripting, you may encounter an error when you run the flow even after completing a migration. This could be due to the flow script containing a reference to an unsupported import or export node. To avoid such errors, you must remove the scripting code that references the unsupported node.
- If the stream you're importing contains unsupported data file types, you need to convert them to a supported type (CSV, Excel, or SPSS Statistics .sav).

**Parent topic:** [Creating SPSS Modeler flows](#)

## Extension nodes

---

SPSS Modeler supports the languages R and Apache Spark (via Python).

To complement SPSS Modeler and its data mining abilities, several Extension nodes are available to enable expert users to input their own R scripts or Python for Spark scripts to carry out data processing, model building, and model scoring.

- The Extension Import node is available under Import on the Node Palette. See [Extension Import node](#).
- The Extension Transform node is available under Record Operations on the Node Palette. See [Extension Transform node](#).
- The Extension Model node is available under Modeling on the Node Palette. See [Extension Model node](#).
- The Extension Output node is available under Outputs on the Node Palette. See [Extension Output node](#).
- The Extension Export node is available under Export on the Node Palette. See [Extension Export node](#).
- **R scripts**  
SPSS Modeler supports R scripts.

- **Python for Spark scripts**  
SPSS Modeler supports Python scripts for Apache Spark.

**Parent topic:** [Creating SPSS Modeler flows](#)

## R scripts

---

SPSS Modeler supports R scripts.

### Allowable syntax

---

- In the syntax field of the properties for the various Extension nodes, only statements and functions that are recognized by R are allowed.
- For the Extension Transform node and the Extension model nugget, data passes through the R script (in batch). For this reason, R scripts for model scoring and process nodes should not include operations that span or combine rows in the data, such as sorting or aggregation. This limitation is imposed to ensure that data can be split up in a Hadoop environment, and during in-database mining. Extension Output and Extension model building nodes do not have this limitation.
- The addition of a non-batch data transfer mode, in both the Extension Transform node and the Extension model nugget, means that you can either span or combine rows in the data.
- All R nodes can be seen as independent global R environments. Therefore, using `library` functions within the two separate R nodes requires the loading of the R library in both R scripts.
- To display the value of an R object that's defined in your R script, you must include a call to a printing function. For example, to display the value of an R object that's called `data`, include the following line in your R script:

```
print(data)
```

- You can't include a call to the R `setwd` function in your R script because this function is used by SPSS Modeler to control the file path of the R scripts output file.
- Flow parameters that are defined for use in CLEM expressions and scripting are not recognized if used in R scripts.
- SPSS Modeler doesn't support the interactive plot in R

**Parent topic:** [Extension nodes](#)

## Python for Spark scripts

---

SPSS Modeler supports Python scripts for Apache Spark.

Note:

- Python nodes depend on the Spark environment.
- Python scripts must use the Spark API because data is presented in the form of a Spark DataFrame.
- When installing Python, make sure all users have permission to access the Python installation.
- If you want to use the Machine Learning Library (MLlib), you must install a version of Python that includes NumPy.
- **Scripting with Python for Spark**  
SPSS Modeler can run Python scripts using the Apache Spark framework to process data. This documentation provides the Python API description for the interfaces provided.

**Parent topic:** [Extension nodes](#)

## Scripting with Python for Spark

---

SPSS Modeler can run Python scripts using the Apache Spark framework to process data. This documentation provides the Python API description for the interfaces provided.

The SPSS Modeler installation includes a Spark distribution.

## Accessing data

---

Data is transferred between a Python/Spark script and the execution context in the form of a Spark SQL DataFrame. A script that consumes data (that is, any node except an Import node) must retrieve the data frame from the context:

```
inputData = asContext.getSparkInputData()
```

A script that produces data (that is, any node except a terminal node) must return a data frame to the context:

```
asContext.setSparkOutputData(outputData)
```

You can use the SQL context to create an output data frame from an RDD where required:

```
outputData = sqlContext.createDataFrame(rdd)
```

## Defining the data model

---

A node that produces data must also define a data model that describes the fields visible downstream of the node. In Spark SQL terminology, the data model is the schema.

A Python/Spark script defines its output data model in the form of a `pyspark.sql.types.StructType` object. A `StructType` describes a row in the output data frame and is constructed from a list of `StructField` objects. Each `StructField` describes a single field in the output data model.

You can obtain the data model for the input data using the `:schema` attribute of the input data frame:

```
inputSchema = inputData.schema
```

Fields that are passed through unchanged can be copied from the input data model to the output data model. Fields that are new or modified in the output data model can be created using the `StructField` constructor:

```
field = StructField(name, dataType, nullable=True, metadata=None)
```

Refer to your Spark documentation for information about the constructor.

You must provide at least the field name and its data type. Optionally, you can specify metadata to provide a measure, role, and description for the field (see [Data metadata](#)).

## DataModelOnly mode

---

SPSS Modeler needs to know the output data model for a node, before the node runs, to enable downstream editing. To obtain the output data model for a Python/Spark node, SPSS Modeler runs the script in a special "data model only" mode where there is no data available. The script can identify this mode using the `isComputeDataModelOnly` method on the Analytic Server context object.

The script for a transformation node can follow this general pattern:

```
if asContext.isComputeDataModelOnly():
    inputSchema = asContext.getSparkInputSchema()
    outputSchema = ... # construct the output data model
    asContext.setSparkOutputSchema(outputSchema)
else:
    inputData = asContext.getSparkInputData()
    outputData = ... # construct the output data frame
    asContext.setSparkOutputData(outputData)
```

## Building a model

---

A node that builds a model must return to the execution context some content that describes the model sufficiently that the node which applies the model can recreate it exactly at a later time.

Model content is defined in terms of key/value pairs where the meaning of the keys and the values is known only to the build and score nodes and is not interpreted by SPSS Modeler in any way. Optionally the node may assign a MIME type to a value with the intent that SPSS Modeler might display those values which have known types to the user in the model nugget.

A value in this context may be PMML, HTML, an image, etc. To add a value to the model content (in the build script):

```
asContext.setModelContentFromString(key, value, mimeType=None)
```

To retrieve a value from the model content (in the score script):

```
value = asContext.getModelContentToString(key)
```

As a shortcut, where a model or part of a model is stored to a file or folder in the file system, you can bundle all the content stored to that location in one call (in the build script):

```
asContext.setModelContentFromPath(key, path)
```

Note that in this case there is no option to specify a MIME type because the bundle may contain various content types.

If you need a temporary location to store the content while building the model you can obtain an appropriate location from the context:

```
path = asContext.createTemporaryFolder()
```

To retrieve existing content to a temporary location in the file system (in the score script):

```
path = asContext.getModelContentToPath(key)
```

## Error handling

---

To raise errors, throw an exception from the script and display it to the SPSS Modeler user. Some exceptions are predefined in the module `spss.pyspark.exceptions`. For example:

```
from spss.pyspark.exceptions import ASContextException
if ... some error condition ...:
    raise ASContextException("message to display to user")
```

- **Analytic Server Context**

The Context provides support for the Analytic Server Context interface for interaction with the SPSS Analytic Server.

- **Data metadata**

This section describes how to set up the data model attributes based on `pyspark.sql.StructField`.

- **Date, time, timestamp**

- **Exceptions**

This section describes possible exception instances. They are all a subclass of python exception.

- **Examples**

This section provides Python for Spark scripting examples.

**Parent topic:** [Python for Spark scripts](#)

## Analytic Server Context

---

The Context provides support for the Analytic Server Context interface for interaction with the SPSS Analytic Server.

## AnalyticServerContext Objects

---

`AnalyticServerContext` objects set up the context environment which provides several interfaces for interacting with SPSS Analytic Server. An application that wants to construct this context instance must do so using the `spss.pyspark.runtime.getContext()` interface rather than implementing the interface directly.

Returns the Pyspark python `SparkContext` instance:

```
cxt.getSparkContext() : SparkContext
```

Returns the Pyspark python `SQLContext` instance:

```
cxt.getSparkSQLContext() : SQLContext
```

Returns `True` to describe whether the execution is made only to compute the output data model. Otherwise returns `False`:

```
cxt.isComputeDataModelOnly() : Boolean
```

Returns `True` if the script is running in the Spark environment. Currently, it always returns `True`:

```
cxt.isSparkExecution() : Boolean
```

Loads input data from the upstream temporary file and generates the `pyspark.sql.DataFrame` instance:

```
cxt.getSparkInputData() : DataFrame
```

Returns a `pyspark.sql.StructType` instance generated from the input data model. Returns `None` if the input data model does not exist:

```
cxt.getSparkInputSchema() : StructType
```

Serializes the output data frame into Analytic Server context and returns the context:

```
cxt.setSparkOutputData(outDF) : AnalyticServerContext
```

Parameter:

- `outDF (DataFrame)` : The output data frame value

Exceptions:

- `DataOutputNotSupported` : If this interface is invoked in the function `pyspark:buildmodel`
- `ASContextException` : If the output data frame is `None`
- `InconsistentOutputDataModel` : The field names and storage type information common to both objects is inconsistent

Converts the `outSchema StructType` instance into a data model, serializes it into the Analytic Server context, and returns the context:

```
cxt.setSparkOutputSchema(outSchema) : AnalyticServerContext
```

Parameter:

- `outSchema (StructType)` : The output `StructType` object

Exceptions:

- `ASContextException` : If the output schema instance is `None`
- `InconsistentOutputDataModel` : The field names and storage type information common to both objects is inconsistent

Stores the location of model building output to the Analytic Server context and returns the context:

```
cxt.setModelContentFromPath(key, path, mimetype=None) : AnalyticServerContext
```

The path can be a directory path which should use the `cxt.createTemporaryFolder()` API to generate, when everything under the directory is packaged up as model content.

Parameters:

- `key (string)` : key string value
- `path (string)` : location of model building output string path
- `mimetype (string, optional)` : the MIME type of the content

Exceptions:

- `ModelOutputNotSupported` : When not invoking this API from the `pyspark:buildmodel` function
- `KeyError` : If the key attribute is `None` or the string is empty

Stores the model building content, metadata, or other attributes to the Analytic Server context and returns the context:

```
cxt.setModelContentFromString(key, value, mimetype=None) : AnalyticServerContext
```

Parameters:

- `key (string)` : key string value
- `value (string)` : the model metadata string value
- `mimetype (string, optional)` : the MIME type of the content

Exceptions:

- `ModelOutputNotSupported` : When not invoking this API from the `pyspark:buildmodel` function
- `KeyError` : If the key attribute is `None` or the string is empty

Returns the temporary folder location that is managed by Analytic Server; this can be used to store the model content:

```
cxt.createTemporaryFolder() : string
```

Exception:

- `ModelOutputNotSupported` : When not invoking this API from the `pyspark:buildmodel` function

Returns the location of the model which matches the input key:

```
cxt.getModelContentToPath(key) : string
```

Parameter:

- `key (string)` : key string value

Exceptions:

- `ModelInputNotSupported` : When not invoking this API from the `pyspark:applymodel` function
- `KeyError` : If the key attribute is `None` or the string is empty
- `IncompatibleModelContentType` : If the model content type is not a container

Returns the model content, metadata of the model, or other model attributes which match the input key:

```
cxt.getModelContentToString(key) : string
```

Parameter:

- `key (string)` : key string value

Exceptions:



- `ModelInputNotSupported` : When not invoking this API from the `pyspark:applymodel` function
- `KeyError` : If the key attribute is `None`, or the string is empty, or the key does not exist
- `IncompatibleModelContentType` : If the model content type is not consistent

Returns the mime-type assigned to the input key. It returns `None` if the specified content has no mime type:

```
cxt.getModelContentMimeType(key) : string
```

Parameter:

- `key (string)` : key string value

Exceptions:

- `ModelInputNotSupported` : When not invoking this API from the `pyspark:applymodel` function
- `KeyError` : If the key attribute is `None`, or the string is empty, or the key does not exist

**Parent topic:** [Scripting with Python for Spark](#)

## Data metadata

---

This section describes how to set up the data model attributes based on `pyspark.sql.StructField`.

### `spss.datamodel.Role` Objects

---

This class enumerates valid roles for each field in a data model.

**BOTH:** Indicates that this field can be either an antecedent or a consequent.

**FREQWEIGHT:** Indicates that this field is to be used as a frequency weight; this isn't displayed to the user.

**INPUT:** Indicates that this field is a predictor or an antecedent.

**NONE:** Indicates that this field is not used directly during modeling.

**TARGET:** Indicates that this field is predicted or a consequent.

**PARTITION:** Indicates that this field identifies the data partition.

**RECORDID:** Indicates that this field identifies the record id.

**SPLIT:** Indicates that this field splits the data.

### `spss.datamodel.Measure` Objects

---

This class enumerates measurement levels for fields in a data model.

**UNKNOWN:** Indicates that the measure type is unknown.

**CONTINUOUS:** Indicates that the measure type is continuous.

**NOMINAL:** Indicates that the measure type is nominal.

**FLAG:** Indicates that the field value is one of two values.

**DISCRETE:** Indicates that the field value should be interpreted as a collection of values.

**ORDINAL:** Indicates that the measure type is ordinal.

**TYPELESS:** Indicates that the field can have any value compatible with its storage.

## pyspark.sql.StructField Objects

---

Represents a field in a `StructType`. A `StructField` object comprises four fields:

- `name` (string): name of a `StructField`
- `dataType` (`pyspark.sql.DataType`): specific data type
- `nullable` (bool): if the values of a `StructField` can contain `None` values
- `metadata` (dictionary): a python dictionary that stores the option attributes

You can use the metadata dictionary instance to store the measure, role, or label attribute for the specific field. The key words for these attributes are:

- `measure`: the key word for measure attribute
- `role`: the key word for role attribute
- `displayLabel`: the key word for label attribute

Example:

```
from spss.datamodel.Role import Role
from spss.datamodel.Measure import Measure
_metadata = {}
_metadata['measure'] = Measure.TYPELESS
_metadata['role'] = Role.NONE
_metadata['displayLabel'] = "field label description"
StructField("userName", StringType(), nullable=False,
metadata=_metadata)
```

**Parent topic:** [Scripting with Python for Spark](#)

## Date, time, timestamp

---

For operations that use **date**, **time**, or **timestamp** type data, the value is converted to the real value based on the value 1970-01-01:00:00:00 (using Coordinated Universal Time).

For the **date**, the value represents the number of days, based on the value 1970-01-01 (using Coordinated Universal Time).

For the **time**, the value represents the number of seconds at 24 hours.

For the **timestamp**, the value represents the number of seconds based on the value 1970-01-01:00:00:00 (using Coordinated Universal Time).

**Parent topic:** [Scripting with Python for Spark](#)

## Exceptions

---

This section describes possible exception instances. They are all a subclass of python exception.

### MetadataException Objects

---

This exception is thrown if an error occurs during operation of the metadata object.

### UnsupportedOperationException Objects

---

This exception is thrown if the specific operation doesn't allow execution.

### InconsistentOutputDataModel Objects

---

This exception is thrown if both `setSparkOutputSchema` and `setSparkOutputData` are invoked but the field names and storage type information common to both objects are inconsistent.

## IncompatibleModelContentType Objects

---

This exception is thrown during the following scenarios:

- Using `setModelContentFormString` to set model but using `getModelContentToPath` to get value
- Using `setModelContentFormPath` to set model but using `getModelContentToString` to get value

## DataOutputNotSupported Objects

---

This exception is raised in `setSparkOutputData` in an execution handled by function `pyspark:buildmodel`.

## ModelInputNotSupported Objects

---

This exception is only raised if the script doesn't invoke the `getModelContentPathByKey` and `getModelContentToString` APIs in the `pyspark:applymodel` function.

## ModelOutputNotSupported Objects

---

This exception is only raised if the script doesn't invoke the `setModelContentFromPath` and `setModelContentFromString` APIs in the `pyspark:buildmodel` function.

## ASContextException Objects

---

This exception is thrown if an unexpected runtime exception occurs.

**Parent topic:** [Scripting with Python for Spark](#)

## Examples

---

This section provides Python for Spark scripting examples.

### Basic scripting example for processing data

---

```
import spss.pyspark.runtime
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()

if cxt.isComputeDataModelOnly():
    _schema = cxt.getSparkInputSchema()
    cxt.setSparkOutputSchema(_schema)
else:
    _structType = cxt.getSparkInputSchema()
    df = cxt.getSparkInputData()
    _newDF = df.sample(False, 0.01, 1)
    cxt.setSparkOutputData(_newDF)
```

### Example model building script, using the LinearRegressionWithSGD algorithm

---

```
from pyspark.context import SparkContext
from pyspark.sql.context import SQLContext
from pyspark.sql import Row
from pyspark.mllib.regression import
LabeledPoint, LinearRegressionWithSGD, LinearRegressionModel
from pyspark.mllib.linalg import DenseVector
```

```

import numpy
import json

import spss.pyspark.runtime
from spss.pyspark.exceptions import ASContextException

ascontext = spss.pyspark.runtime.getContext()
sc = ascontext.getSparkContext()
df = ascontext.getSparkInputData()

# field settings and algorithm parameters

target = '%target_field%'
predictors = [%predictor_fields%]
num_iterations=%num_iterations%
prediction_field = "$LR-" + target

# save linear regression model to a filesystem path

def save(model, sc, path):
    data =
sc.parallelize([json.dumps({"intercept":model.intercept,"weights":model.weights.tolist()})
])
    data.saveAsTextFile(path)

# print model details to stdout

def dump(model,predictors):
    print(prediction_field+" = " + str(model.intercept))
    weights = model.weights.tolist()
    for i in range(0,len(predictors)):
        print("\t+ "+predictors[i]+"*"+ str(weights[i]))

# check that required fields exist in the input data

input_field_names = [ty[0] for ty in df.dtypes[:]]
if target not in input_field_names:
    raise ASContextException("target field "+target+" not found")
for predictor in predictors:
    if predictor not in input_field_names:
        raise ASContextException("predictor field "+predictor+" not found")

# define map function to convert from dataframe Row objects to mllib LabeledPoint

def row2LabeledPoint(target,predictors,row):
    pvals = []
    for predictor in predictors:
        pval = getattr(row,predictor)
        pvals.append(float(pval))
    tval = getattr(row,target)
    return LabeledPoint(float(tval),DenseVector(pvals))

# convert dataframe to an RDD containing LabeledPoint

training_points = df.rdd.map(lambda row:
row2LabeledPoint(target,predictors,row))

# build the model

model = LinearRegressionWithSGD.train(training_points,num_iterations,intercept=True)

# write a text description of the model to stdout

dump(model,predictors)

# save the model to the filesystem and store into the output model content

modelpath = ascontext.createTemporaryFolder()

```

```
save(model,sc,modelpath)
ascontext.setModelContentFromPath("model",modelpath)
```

## Example model scoring script, using the LinearRegressionWithSGD algorithm

---

```
import json
import spss.pyspark.runtime
from pyspark.sql import Row
from pyspark.mllib.regression import
LabeledPoint, LinearRegressionWithSGD, LinearRegressionModel
from pyspark.mllib.linalg import DenseVector
from pyspark.sql.context import SQLContext
import numpy
from pyspark.sql.types import DoubleType, StructField

ascontext = spss.pyspark.runtime.getContext()
sc = ascontext.getSparkContext()

prediction_field = "$LR-" + '%target_field%'
predictors = [%predictor_fields%]

# compute the output schema by adding the prediction field
outputSchema = ascontext.getSparkInputSchema()
outputSchema.fields.append(StructField(prediction_field,
DoubleType(), nullable=True))

# make a prediction based on a regression model and Dataframe Row object
# return a list containing the input row values and the predicted value
def predict(row,model,predictors,infields,prediction_field_name):
    pvals = []
    rdict = row.asDict()
    for predictor in predictors:
        pvals.append(float(rdict[predictor]))
    estimate = float(model.predict(pvals))
    result = []
    for field in infields:
        result.append(rdict[field])
    result.append(estimate)
    return result

# load a serialized model from the filesystem

def load(sc, path):
    js = sc.textFile(path).take(1)[0]
    obj = json.loads(js)
    weights = numpy.array(obj["weights"])
    intercept = obj["intercept"]
    return LinearRegressionModel(weights,intercept)

ascontext.setSparkOutputSchema(outputSchema)

if not ascontext.isComputeDataModelOnly():
    # score the data in the input data frame
    indf = ascontext.getSparkInputData()

    model_path = ascontext.getModelContentToPath("model")
    model = load(sc,model_path)

    # compute the scores
    infield_names = [ty[0] for ty in indf.dtypes[:]]
    scores_rdd = indf.rdd.map(lambda
row:predict(row,model,predictors,infield_names,prediction_field))

    # create an output DataFrame containing the scores
    sqlCtx = SQLContext(sc)
    outdf = sqlCtx.createDataFrame(scores_rdd,schema=outputSchema)
```

```
# return the output DataFrame as the result
ascontext.setSparkOutputData(outdf)
```

**Parent topic:** [Scripting with Python for Spark](#)

## Setting properties for flows

---

You can specify a number of properties to apply to the current flow.

To set flow properties, click the Flow Properties icon:



The following properties are available.

### Options

---

#### General

- Limit members for nominal fields. Select this option and specify a maximum number of members for nominal (set) fields after which the data type of the field becomes Typeless. This option is useful when working with large nominal fields. But note that when the measurement level of a field is set to Typeless, its role is automatically set to None. This means that the fields aren't available for modeling.
- Refresh source nodes on execution. Select this option to automatically refresh all source (import) nodes when running the current flow. This action is analogous to clicking the Refresh button in an import node's properties, except that this option automatically refreshes all import nodes (except User Input nodes) for the current flow.

#### Date/Time

- Import date/time as. Select whether to use date/time storage for date/time fields or whether to import them as string variables.
- Date format. Select a date format to use for date storage fields or when strings are interpreted as dates by CLEM date functions.
- Time format. Select a time format to use for time storage fields or when strings are interpreted as times by CLEM time functions.
- Rollover days/mins. For time formats, select whether negative time differences are interpreted as referring to the previous day or hour.
- Date baseline (1st Jan). Select the baseline years (always 1 January) to be used by CLEM date functions that work with a single date.
- 2-digit dates start from. Specify the cutoff year to add century digits for years that are denoted with only 2 digits. For example, specifying 1930 as the cutoff year assumes that 05/11/02 is in the year 2002. The same setting will use the 20th century for dates after 30; thus 05/11/73 is assumed to be in 1973.
- Time zone. Select how the time zone is chosen for use with the `datetime_now` CLEM expression.
  - If you select Server, the time zone is used from where the SPSS Modeler run-time is running (in some cases this may be the same as the Client option). Or if your flow uses data from a database and the supported database uses SQL pushback, the `datetime_now` expression will use the time of the database.
  - If you select Client, the time zone is used from the machine where SPSS Modeler is installed.
  - Alternatively, you can select any of the Coordinated Universal Time values for the time zone.

#### Number Formats

For standard, scientific, and currency display formats, specify the number of decimal places to use when displaying real numbers.

#### Optimization

You can use these settings to optimize flow performance.

- Enable flow rewriting. Select this option to enable flow rewriting. Flow rewriting reorders the nodes in a flow behind the scenes for more efficient operation, without altering flow semantics.
- Optimize CLEM expressions. This option enables the optimizer to search for CLEM expressions that can be preprocessed before the flow runs, to increase the processing speed. As a simple example, if you have an expression such as `log(salary)`, the optimizer will calculate the actual salary value and pass that on for processing. This can be used to improve both SQL pushback and SPSS Modeler performance.
- Optimize syntax execution. This method of flow rewriting increases the efficiency of operations that incorporate more than one node containing SPSS Statistics syntax. Optimization is achieved by combining the syntax commands into a single operation, instead of running each as a separate operation.
- Optimize other execution. This method of flow rewriting increases the efficiency of operations that can't be delegated to the database. Optimization is achieved by reducing the amount of data in the flow as early as possible. While maintaining data integrity, the flow is rewritten to push operations closer to the data source, thus reducing data downstream for costly operations, such as joins.
- Enable parallel processing. When running on a computer with multiple processors, this option allows the system to balance the load across those processors, which may result in faster performance. Use of multiple nodes or use of the following individual nodes may benefit from parallel processing: C5.0, Merge (by key), Sort, Bin (rank and tile methods), and Aggregate (using one or more key fields).
- Database caching (SQL only). For flows that generate SQL to be run in the database, data can be cached mid flow to a temporary table in the database rather than to the file system. When combined with SQL optimization, this may result in significant gains in performance. For example, the output from a flow that merges multiple tables to create a data mining view may be cached and reused as needed. With database caching enabled, simply right-click any nonterminal node to cache data at that point, and the cache is automatically created directly in the database the next time the flow runs. This allows SQL to be generated for downstream nodes, further improving performance. Alternatively, this option can be disabled if needed, such as when policies or permissions preclude data being written to the database. If database caching or SQL optimization is not enabled, the cache will be written to the file system instead.
- Use relaxed conversion (SQL only). This option enables the conversion of data from either strings to numbers, or numbers to strings, if stored in a suitable format. For example, if the data is kept in the database as a string, but actually contains a meaningful number, the data can be converted for use when the pushback occurs.

## Logging

- Display SQL in the messages log at run time. Specifies whether SQL generated while running the flow is passed to the messages log.
- Display SQL generation in the message log during preparation. During flow preview, specifies whether a preview of the SQL that would be generated is passed to the messages log.
- SQL format Specifies whether any SQL that's displayed in the log should contain native SQL functions or standard ODBC functions of the form `{fn FUNC( . . . )}`, as generated by SPSS Modeler. The former relies on ODBC driver functionality that may not be implemented.
- Reformat SQL for improved readability. Specifies whether SQL displayed in the log should be formatted for readability.

## Scripting

---

Flow scripts are stored as a flow property here and are therefore saved and loaded with a specific flow. For example, you can write a flow script that automates the process of training and applying a model nugget. You can also specify that whenever a particular flow runs, the script should run instead of the flow's canvas content.

### Script language

Select whether to use Python scripting or the legacy SPSS Modeler scripting that was specific to old versions of SPSS Modeler.

### Script

Enter a script to customize operations within a flow. The script is saved with the flow. You can use a script to specify a particular execution order for the terminal nodes within a flow.

When the flow is run

Specify whether to run all terminal nodes or to run the script you provided whenever the flow runs.

## Parameters

---

You can define parameters for use in CLEM expressions and in scripting. They are, in effect, user-defined variables that are saved and persisted with the current flow, session, or SuperNode, and can be accessed from the user interface as well as through scripting. If you save a flow, for example, any parameters set for that flow are also saved. (This distinguishes them from local script variables, which can be used only in the script in which they are declared.) Parameters are often used in scripting to control the behavior of the script, by providing information about fields and values that don't need to be hard coded in the script.

If you set a parameter here in the flow properties, it's available to all nodes in the flow. Click Add Value and enter the following information.

### Name

Parameter names are listed here. For example, to create a parameter for the minimum temperature, you could type `minvalue`. Do not include the `$P-` prefix that denotes a parameter in CLEM expressions. This name is how the parameter is referenced in expressions.

### Label

Lists a descriptive name for each parameter created.

### Storage

Select a storage type from the list. Storage indicates how the data values are stored in the parameter. For example, when working with values containing leading zeros that you want to preserve (such as `008`), you should select String as the storage type. Otherwise, the zeros will be stripped from the value. Available storage types are string, integer, real, time, date, and timestamp. For date parameters, note that you must specify values using ISO standard notation as shown in the next paragraph.

### Value

Lists the current value for each parameter. Adjust the parameter as required. Note that for date parameters, values must be specified in ISO standard notation (that is, YYYY-MM-DD). Dates specified in other formats aren't accepted.

### Measure

Select the measurement level, which is used to describe characteristics of the parameter.

## Messages

---

In the Messages tab of the flow properties, you can easily view messages regarding flow operations, such as running, optimization, and time elapsed for model building and evaluation. Error messages are also reported in this table.

**Parent topic:** [Creating SPSS Modeler flows](#)

## Expression Builder

---

You can type CLEM expressions manually or use the Expression Builder, which displays a complete list of CLEM functions and operators as well as data fields from the current flow, allowing you to quickly build expressions without memorizing the exact names of fields or functions.

The Expression Builder controls automatically add the proper quotes for fields and values, making it easier to create syntactically correct expressions.

### Notes:


- The Expression Builder is not supported in parameter settings.



- If you want to change your datasource, before changing the source you should check that the Expression Builder can still support the functions you have selected. Because not all databases support all functions, you may encounter an error if you run against a new datasource.
- You can run an SPSS Modeler desktop stream file (.str) that contains database functions. But they aren't yet available in the Expression Builder user interface.

## Opening the Expression Builder

---

The Expression Builder is available in all nodes where CLEM expressions are used, such as Select, Balance, Derive, Filler, Analysis, Report, and Table nodes. You can open it by double-clicking the node to open its properties, and then clicking the calculator icon: 

## Creating expressions

---

The Expression Builder provides not only complete lists of fields, functions, and operators - but also access to data values if your data is instantiated.

To create an expression:

1. Type in the Expression box, using the function and field lists as references

*or*

Select the required fields and functions from the lists.

2. Double-click to add the field or function to the Expression field.
3. Use the operand buttons to insert the operations into the expression.

## Selecting functions

---

The function list displays all available CLEM functions and operators. Scroll to select a function from the list, or, for easier searching, use the drop-down list to display a subset of functions or operators. For details, see [Selecting functions](#).

## Selecting fields

---

The field list displays all fields available at this point in the data stream. Scroll to select a field from the list. Double-click the field to add it to your expression. You can also select from recently used fields.

## Viewing and selecting field values

---

You can view field values in the Expression Builder. Note that data must be fully instantiated in an import or Type node to use this feature, so that storage, types, and values are known.

To view values for a field from the Expression Builder, select the required field to list its values. You can then insert the value into the current expression or list.

For flag and nominal fields, all defined values are listed. For continuous (numeric range) fields, the minimum and maximum values are displayed.

- [Selecting functions](#)  
The function list displays all available CLEM functions and operators. Scroll to select a function from the list, or, for easier searching, use the drop-down list to display a subset of functions or operators.

**Parent topic:** [Creating SPSS Modeler flows](#)

## Selecting functions

---

The function list displays all available CLEM functions and operators. Scroll to select a function from the list, or, for easier searching, use the drop-down list to display a subset of functions or operators.

Available functions are grouped into categories for easier searching.

- General Functions contains a selection of some of the most commonly-used functions.
- Recently Used contains a list of CLEM functions used within the current session.
- @ Functions contains a list of all the special functions, which have their names preceded by an @ sign. Note that the @DIFF1 (FIELD1, FIELD2) and @DIFF2 (FIELD1, FIELD2) functions require that the two field types are the same (for example, both Integer or both Long or both Real).
- Database Window Aggregates. If the flow includes a database connection (by means of a Data Asset Import node), this selection lists the window aggregation options that you can use within that database. These options are available in the Expression Builder for Field Operations nodes only. Note that because SPSS Modeler obtains the Window Aggregate Functions from the Database System View, the available options are dependent on database behavior.

Although called "aggregates," these options aren't designed for use in the Aggregate node. They're more applicable to nodes such as Derive or Select. This is because their output is scalar instead of a true aggregate; that is, they don't reduce the amount of data shown in the output in the same way that the Aggregate node does. For example, you could use this sort of aggregation to provide a moving average down through rows of data, such as "average of the current row and all previous rows."

- Built-In Aggregates. Contains a list of the possible modes of aggregation you can use.
- Operators. Lists all the operators you can use when building expressions. Operators are also available from the buttons.
- All Functions. Contains a complete list of available CLEM functions.

After you select a group of functions, double-click to insert the functions into the Expression box at the point indicated by the position of the cursor.

## Database functions

---

You can run an SPSS Modeler desktop stream file (.str) that contains database functions. But they aren't yet available in the Expression Builder user interface.

**Parent topic:** [Expression Builder](#)

## Notebooks

---

A Jupyter notebook is a web-based environment for interactive computing. You can run small pieces of code that process your data, and you can immediately view the results of your computation. Notebooks include all of the building blocks you need to work with data:

- The data
- The code computations that process the data
- Visualizations of the results
- Text and rich media to enhance understanding

## Working with notebooks

---

With IBM Watson Studio, you can create Python notebooks to analyze your data.

### Required service

None

### Data format

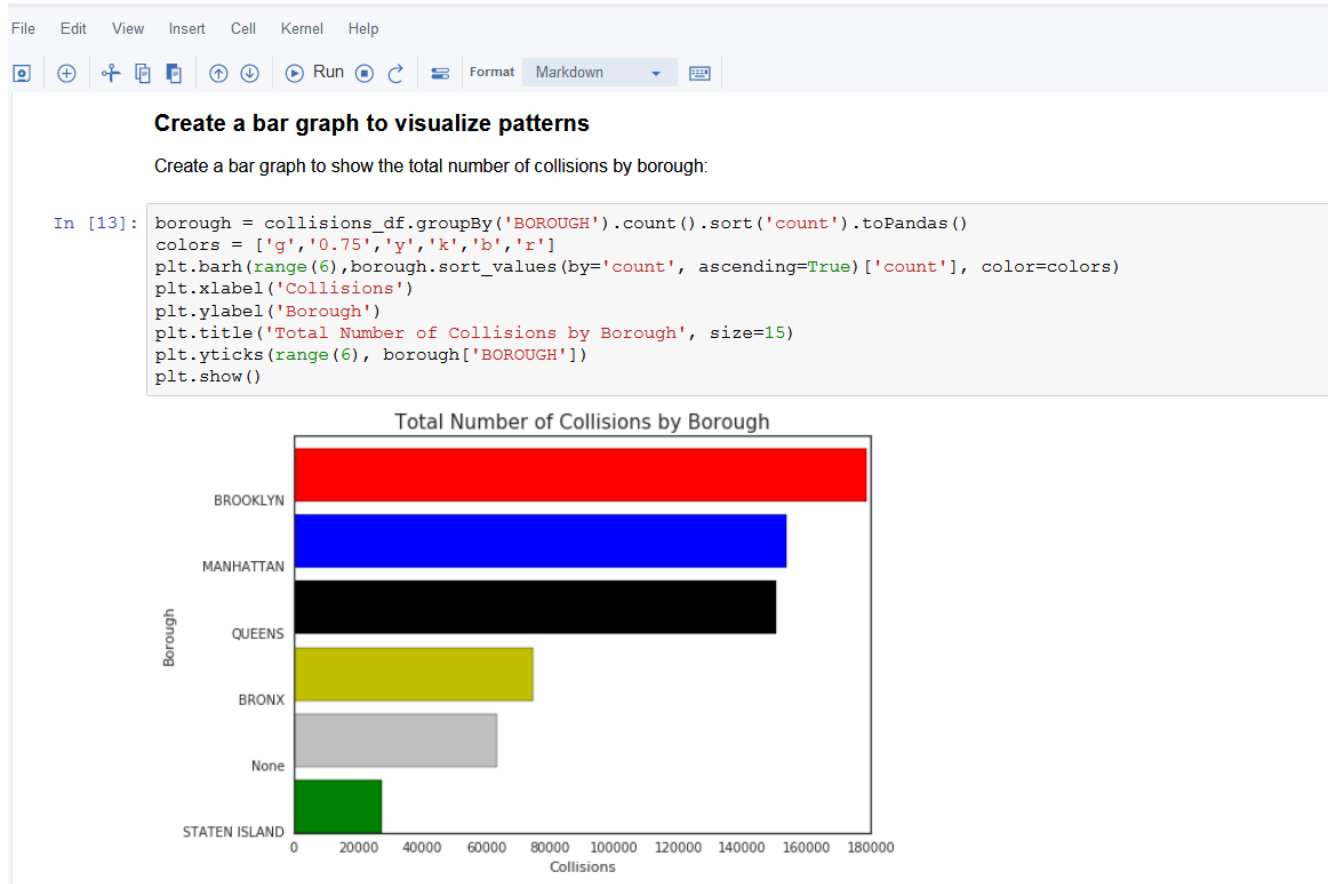
Files from your local file system

Tables in local relational databases

### Data size

5 GB. If your files are larger, you must load the data in multiple parts.

## Notebook UI



Code computations can build upon each other to quickly unlock key insights from your data. Notebooks record how you worked with data, so you can understand exactly what was done and reproduce computations reliably.

With IBM Watson Studio, you can create Python notebooks to analyze your data. If you want to work on more than one notebook at the same time, you can open multiple notebooks on separate browser tabs.

## Learn more

- [Create notebooks](#)
- [Libraries and packages](#)
- [Parts of a notebook](#)
- [Markdown for Jupyter notebooks cheatsheet](#)
- [Code and run notebooks](#)

## Creating notebooks

After you created a project, you can create a notebook file. When you create a notebook file the first time, you are prompted to install the notebook environments. See [Libraries and scripts for notebooks](#).

### Create a notebook file

To create a notebook file in the project:

1. From your project, click **Add to Project** and choose the Notebook asset type.
2. On the **New Notebook** page, specify the method to use to create your notebook. You can create a blank notebook, upload a notebook file from your file system, or upload a notebook file from a URL. The notebook you create or select must be a `.ipynb` file. Currently, you can only create Python notebooks.

Blank

From file

From URL

3. Click **Create Notebook**. The notebook opens in edit mode. The notebook is not locked when it is opened. To get familiar with the structure of a notebook, see [Parts of a notebook](#).

The notebook kernel remains active even if you leave the notebook or close the application window. When you reopen the same notebook, the notebook is connected to the same kernel.

## The parts of a notebook

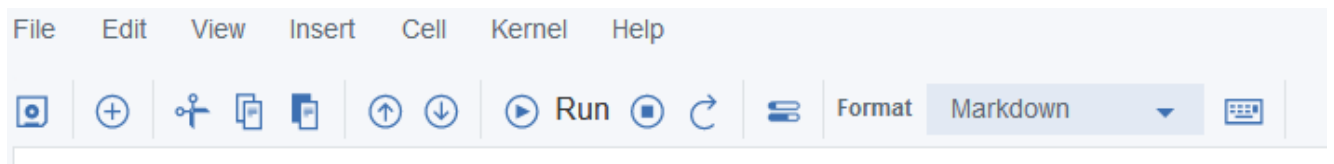
---

The notebook interface includes the following features:

- [Menu bar and toolbar](#)
- [Notebook action bar](#)
- [The cells in a Jupyter notebook](#)
  - [Jupyter Code cells](#)
  - [Jupyter markdown cells](#)
  - [Raw Jupyter NBConvert cells](#)

### Menu bar and toolbar

---



You can select notebook features that affect the way the notebook functions and perform the most-used operations within the notebook by clicking an icon.

### Notebook action bar

---

From the action bar, you can:

- View your notebook information. Change the name of your notebook by editing it in the **Name** field.

### The cells in a Jupyter notebook

---

A Jupyter notebook consists of a sequence of cells. The flow of a notebook is sequential. You enter code into an input cell, and when you run the cell, the notebook executes the code and prints the output of the computation to an output cell.

You can change the code in an input cell and re-run the cell as often as you like. In this way, the notebook follows a read-evaluate-print loop paradigm. You can choose to use tags to describe cells in a notebook.

The behavior of a cell is determined by a cell's type. The different types of cells include:

#### Jupyter code cells

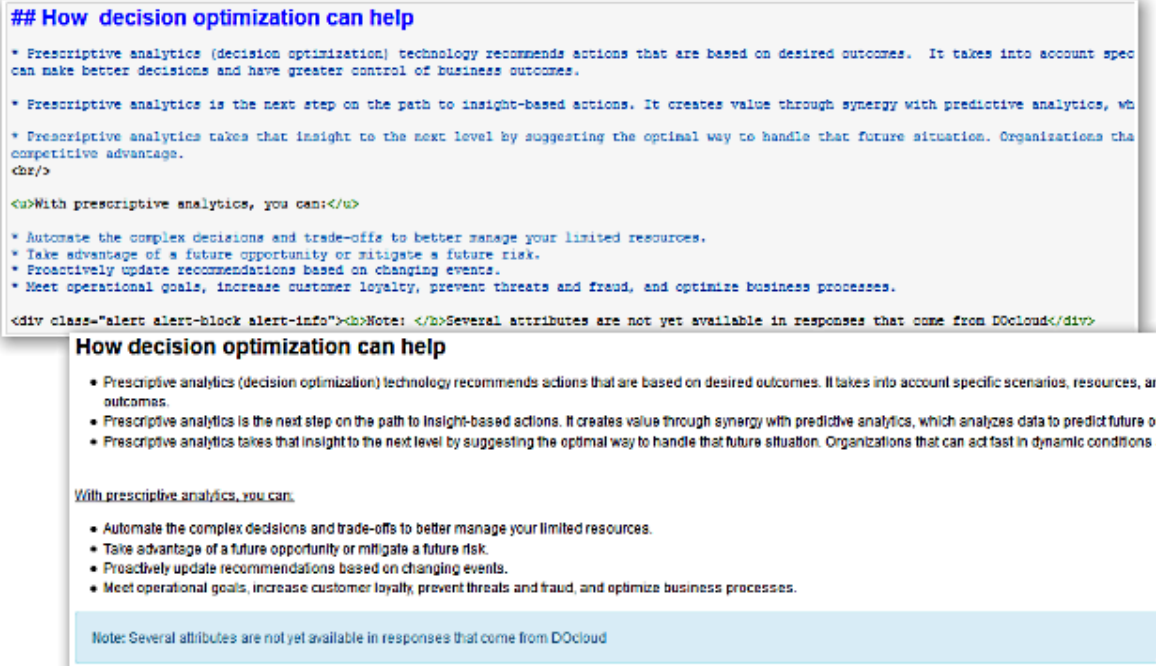
Where you can edit and write new code.

```
In [6]: import pandas as pd
gaspd = pd.DataFrame([(gas_names[i],int(gas_data[i][0]),int(gas_data[i][1]),int(gas_data[i][2]),int(gas_data[i][3]))
                    for i in range_gas])
oilpd = pd.DataFrame([(oil_names[i],int(oil_data[i][0]),int(oil_data[i][1]),int(oil_data[i][2]),oil_data[i][3])
                    for i in range_oil])
gaspd.columns = ['name','demand','price','octane','lead']
oilpd.columns= ['name','capacity','price','octane','lead']
```

## Jupyter markdown cells

Where you can document the computational process. You can input headings to structure your notebook hierarchically.

You can also add and edit image files as attachments to the notebook. The markdown code and images are rendered when the cell is run.



See [Markdown for Jupyter notebooks cheatsheet](#).

## Raw Jupyter NBConvert cells

Where you can write output directly or put code that you don't want to run. Raw cells are not evaluated by the notebook.

```
from docplex.cp.model import *
url = "https://api-oaas.doccloud.ibmcloud.com/job_manager/rest/v1"
key = "api_5bf69af0-ba5f-4236-b8eb-4c9a2f909436"
```

## Libraries and scripts for notebooks

When you select to install the environments for notebooks, the standard Python 3.6 libraries are installed to your computer.

### List installed libraries

To see the list of installed libraries, run the following command from a notebook cell:

```
!conda list
```

## Install additional libraries

---

If the library that you want to use in your notebook is not listed, use the Python conda package installer command to install the library to your environment. For example, run the following command in a code cell to install the spaCy library:

**Important:** On Windows, you must use "Run as Administrator" to install libraries or packages in your notebook.

```
!conda install spacy -y
```

## Import installed libraries

---

To import an installed library into your notebook, run the following command from a notebook cell with the library name:

```
import library_name
```

You can write a script that includes multiple classes and methods and then [import the script into your notebook](#).

## Importing scripts into a notebook

---

You might only want to include the necessary bits of code for your notebook that can be used in a presentation to communicate the results of your analysis. All helper functions, classes, and visualization code snippets for example, do not have to be included in the notebook. Instead you can add this code to a script which can be shared by all of the notebooks in your Watson Studio installation.

Any Python modules or packages on your local machine that have been added to the PYTHONPATH environment variable can be used from within the notebook by importing them.

To import the classes to access the methods in a script in your notebook, use the following command:

```
from <module> import <class name>
```

## Coding and running a notebook

---

After you create a notebook, you're ready to start writing and running code to analyze data.

A notebook runs in a Jupyter kernel in the notebook environments that you installed before you created the notebook. As only one conda Python 3.6 environment is installed to your computer, you cannot switch notebook kernels. Before you start coding, become familiar with the [notebook interface](#) and how to code in [Markdown](#) to annotate your code.

When you open a new notebook, the notebook is considered to be *untrusted* by the Jupyter service by default. When you run an untrusted notebook, content deemed untrusted will not be executed. Untrusted content includes any Javascript, or HTML or Javascript in Markdown cells or in any output cells that you did not generate.

To tell the service to trust your notebook content and execute all cells:

1. Click **Not Trusted** in the upper right corner of the notebook.
2. Click **Trust** to execute all cells.

To develop analytic applications in a notebook, follow these general steps:

1. [Import preinstalled libraries or install your own libraries](#).
2. Load and access data. See [Load and access data](#).
3. Prepare and analyze the data.
4. Save your changes to the notebook.

5. When you're not actively working on the notebook, click **File > Stop Kernel** to stop the notebook kernel and free up resources.
6. You can't download your notebook from the notebook editor. To retrieve the notebook, you can find the notebook in the following folder on your local machine:
  - **Windows:** `\Users\username\AppData\Roaming\IBM Watson Studio\projects\project-name\assets\notebook`
  - **macOS:** `/Users/username/Library/Application Support/IBM Watson Studio/projects/project-name/assets/notebook`

Alternatively, from the project page, you can click **View folder** under **LOCATION** in the left corner and navigate to the project assets.

## Loading and accessing data in a notebook

---

You can integrate data into notebooks by loading the data from your local file system or from a local database into a data structure or container in your notebook, for example, a `pandas.DataFrame`, or `numpy.array`.


### Load data from local files

---

To access data from a local file, you can load the file directly from within a notebook, or first load the file as a data asset into your project. From your notebook, you add automatically generated code to access the data by using the `Insert to code` function. The inserted code serves as a quick start to allow you to easily begin working with data sets.

The `Insert to code` function supports the file types CSV, JSON and XLSX only. For file types that are not supported, you can only select to insert the credentials. This returns the path to the file. With the file path information, you can write your own code to load the file data into a `DataFrame` or other data structure in a notebook cell. See the [project-lib library](#) for functions to help you read the data.

To add a file from your local system to your notebook:

1. In the notebook, click the **Find and Add Data** icon () , and then browse a data file or drag it into your notebook sidebar.
2. Click in an empty code cell in your notebook and then click the **Insert to code** link below the file and choose how to load the data.

Code is generated and added to your notebook for you. The generated code imports any required packages, accesses the data file, and loads the data into a `DataFrame` or `RDD`. For file type JSON, the generated code includes a description and the code is in comments in case you want to change the way the JSON data is accessed.

### Load data from a connection

---

To access data stored in a local database, or from a remote data source, you must connect to the data source and write your own code to access and load the data to a data structure in your notebook.

For sample code showing how to access data from a Planning Analytics connection, see [Adding data from a Planning Analytics connection](#).

## Adding data from a Planning Analytics connection

---

You need to write your own code if you want to access data from a Planning Analytics connection in a notebook. To do this you can use available open source libraries like for example `TM1Py`, a Python package that wraps the Planning

Analytics REST API in a library. After you install the open source library, you can use the library functions to connect to the Planning Analytics service and to access the data.

**NOTE:** The information about the third-party open source libraries is provided for your convenience. Although the code has been tested, it is not officially supported by IBM.

The following sample code show you how to read data from a Planning Analytics connection and load the data to dataframes in a notebook using the TM1Py open source library:

- See [TM1Py](#) for details on the available functions.
- See [TM1Py - Help](#) for getting started with the library.

1. In a notebook code cell, begin by installing the TM1Py library:

**Important:** On Windows, you must use "Run as Administrator" to install libraries or packages in your notebook.

```
# install TM1Py library
!pip install TM1py
```

2. Then use TM1Py library functions to connect to the Planning Analytics service.

- If the Planning Analytics service uses basic authentication, you need to specify the URL, username and password:

```
# Connect to your Planning Analytics service
# The base URL is the TM1 API endpoint excluding the /api/v1 suffix
from TM1py.Services import TM1Service
tm1 = TM1Service(base_url='http://<server.company.com:port>',
user='<username>', password='<password>', ssl=False)
```

- If the Planning Analytics uses CAM authentication, you need to specify not only the URL, username and password, but also the namespace:

```
# Connect to your Planning Analytics service
# The base URL is the TM1 API endpoint excluding the /api/v1 suffix
from TM1py.Services import TM1Service
tm1 = TM1Service(base_url='http://<server.company.com:port>',
user='<username>', password='<password>', namespace='<namespace>', ssl=False)
```

**Hint:** Set the `ssl` parameter to `True` if the connection to the Planning Analytics service is a secure HTTP connection.

3. There are different ways to use the TM1Py library with your Planning Analytics data. The following sample code shows you how to read all the cubes to retrieve the name of the cube with the data you want to analyze, and to use the views of the retrieved cube to browse and modify the data.

```
# Read all cubes and print cube names
tm1.cubes.get_all_names()
```

4. Read the views of a specific cube and print the view names:

```
# Read all views for one cube and print the view names
tm1.cubes.views.get_all_names(cube_name="<cube-name>")
```

5. Load the data from a public view to a pandas DataFrame and print the first entries. If you want to load data from a private view, you need to set the `private` parameter to `True`.

```
# Read a dataframe from a public view and print the entries:
df_1 = tm1.cubes.cells.execute_view_dataframe(cube_name="<cube-name>", view_name="
<view-name>", private=False)
df_1.head()
```

## Using project-lib for Python



The `project-lib` library for Python contains a set of functions that help you to interact with Watson Studio projects and project assets. You can think of the library as a programmatical interface to a project. Using the `project-lib` library, you can access project metadata and assets, including files and connections. The library also contains functions that simplify fetching files associated with the project.

**Note:**

- The `project-lib` functions do not encode or decode data when saving data to or getting data from a file.

## The `project-lib` functions

---

The instantiated project object that is created after you have imported the `project-lib` library exposes a set of functions that are grouped in the following way:

- [Fetch project information](#)
- [Fetch files](#)

### Fetch project information

You can use the following functions to fetch project-related information programmatically:

- `get_metadata()`

This function returns the project metadata. The following example shows you how to get the ID of the project in which your notebook is located:

```
import project_lib
project = project_lib.Project()
project.get_metadata()["metadata"]["guid"]
```

- `get_files()`

This function returns the list of the files in your project. Each element in the returned list contains the ID and the name of the file. The list of returned files is not sorted by any criterion and can change when you call the function again.

- `get_dataassets_folder_path()`

This function returns the folder name where all the project assets are stored.

### Fetch files

You can use the following function to fetch files associated with your project.

- `get_file(file_name)` where `file_name` is the name of the file you want to fetch.

This function fetches a file into the memory of the running kernel. The function returns a byte buffer which can be used to bind into kernel-specific data structures, for example, a pandas DataFrame. This method of fetching files is not recommended for very large files.

The following example shows you how to fetch a file and read the data into a pandas DataFrame:

```
# Import the lib
from project_lib import Project
project = Project.access()

# Fetch the file
my_file = project.get_file("myFile.csv")

# Read the CSV data file into a pandas DataFrame
my_file.seek(0)
```

```
import pandas as pd
pd.read_csv(my_file, nrows=10)
```

## Markdown cheatsheet

---

Here's how to format the project readme file or Markdown cells in Jupyter notebooks. The differences between Markdown in the readme files and in notebooks are noted.

**Headings:** Use #s followed by a blank space for notebook titles and section headings:

```
# title
## major headings
### subheadings
#### 4th level subheadings
```

**Emphasis:** Use this code: Bold: `__string__` or `**string**`, Italic: `_string_` or `*string*`, Strikethrough: `~~string~~`

**Mathematical symbols:** Use this code: `$ mathematical symbols $`

**Monospace font:** Surround text with a back single quotation mark (```). Use monospace for file path and file names and for text users enter or message text users see.

**Line breaks:** Sometimes Markdown doesn't make line breaks when you want them. Put two spaces at the end of the line, or use this code for a manual line break: `<br>`

**Indented quoting:** Use a greater-than sign (`>`) and then a space, then type the text. The text is indented and has a gray horizontal line to the left of it until the next carriage return.

**Bullets:** Use the dash sign (`-`) with a space after it or a space, a dash, and a space (`-` ), to create a circular bullet. To create a sub bullet, use a tab followed a dash and a space. You can also use an asterisk instead of a dash, and it works the same.

**Numbered lists:** Start with `1.` followed by a space, then your text. Hit return and numbering is automatic. Start each line with some number and a period, then a space. Tab to indent to get subnumbering.

**Checkboxes in readme files:** Use this code for an unchecked box: `- [ ]` Use this code for a checked box: `- [x]`

**Tables in readme files:** Use this code:

```
| Heading | Heading |
| ----| ----|
| text   | text   |
| text   | text   |
```

**Graphics in notebooks:** Drag and drop images to the Markdown cell to attach it to the notebook. To add images to other cell types, use graphics that are hosted on the web with this code, substituting `url/name` with the full URL and name of the image: ``

**Graphics in readme files:** Use this code: `![Alt text](url/filename.gif "Title text")`

**Geometric shapes:** Use this code with a decimal or hex reference number from here: [UTF-8 Geometric shapes](#) `&#reference_number;`

**Horizontal lines:** Use three asterisks: `***`

**Internal links:** To link to a section with the notebook, use this code: `[section title](#section-title)` For the text in the parentheses, replace spaces and special characters with a hyphen. Make sure to test all the links!

Alternatively, you can add an ID for a section right above the section title. Use this code: `<a id="_dcs_markdown_workspace_Transform_htmlout_0_ws_j_analyze-data_markd-jupyter_section_ID"></a>` Make sure that the section\_ID is unique within the notebook.

**External links:** Use this code and test all links! `[link text] (http://url)`

## AutoAI Overview (Version 2.0)

---

The AutoAI graphical tool in Watson Studio automatically analyzes your data and generates candidate model pipelines customized for your predictive modeling problem. These model pipelines are created iteratively as AutoAI analyzes your dataset and discovers data transformations, algorithms, and parameter settings that work best for your problem setting. Results are displayed on a leaderboard, showing the automatically generated model pipelines ranked according to your problem optimization objective.

### Required service

Watson Machine Learning service

### Data format

Tabular: CSV files, with comma (,) delimiter

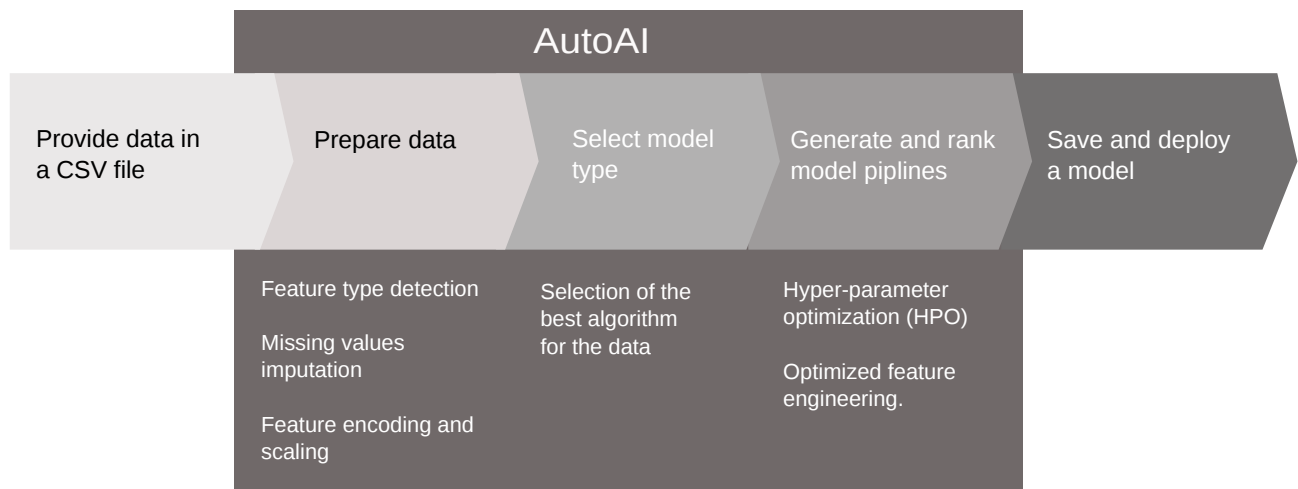
### Data size

Less than 1 GB

## AutoAI process

---

Using AutoAI, you can build and deploy a machine learning model with sophisticated training features and no coding. The tool does most of the work for you.



AutoAI automatically runs the following tasks to build and evaluate candidate model pipelines:

- [Data pre-processing](#)
- [Automated model selection](#)
- [Automated feature engineering](#)
- [Hyperparameter optimization](#)

### Data pre-processing

Most data sets contain different data formats and missing values, but standard machine learning algorithms work with numbers and no missing values. AutoAI applies various algorithms, or estimators, to analyze, clean, and prepare your raw data for machine learning. It automatically detects and categorizes features based on data type, such as

categorical or numerical. Depending on the categorization, it uses hyper-parameter optimization to determine the best combination of strategies for missing value imputation, feature encoding, and feature scaling for your data.

## Automated model selection

The next step is automated model selection that matches your data. AutoAI uses a novel approach that enables testing and ranking candidate algorithms against small subsets of the data, gradually increasing the size of the subset for the most promising algorithms to arrive at the best match. This approach saves time without sacrificing performance. It enables ranking a large number of candidate algorithms and selecting the best match for the data.

## Automated feature engineering

Feature engineering attempts to transform the raw data into the combination of features that best represents the problem to achieve the most accurate prediction. AutoAI uses a unique approach that explores various feature construction choices in a structured, non-exhaustive manner, while progressively maximizing model accuracy using reinforcement learning. This results in an optimized sequence of transformations for the data that best match the algorithms of the model selection step.

## Hyperparameter optimization

Finally, a hyper-parameter optimization step refines the best performing model pipelines. AutoAI uses a novel hyper-parameter optimization algorithm optimized for costly function evaluations such as model training and scoring that are typical in machine learning. This approach enables fast convergence to a good solution despite long evaluation times of each iteration.

## Next step

Use your own data to [build an AutoAI model](#).

# AutoAI tutorial: Build a binary classification model (Version 2.0)

---

This tutorial guides you through training a model to predict whether or not a customer is likely to subscribe to a bank promotion. In this tutorial, you will create an AutoAI experiment that analyzes your data and selects the best model type and algorithms to produce, train, and optimize pipelines, which are model candidates. After reviewing the pipelines, you will save one as a model, deploy it, then test it to get a prediction.



The steps are as follows:

1. Collect the data.
2. Create and run the experiment.
3. Save a pipeline as a model.
4. Deploy and test the model.

## Collecting the data

---

The data set is from a direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to train a model that can predict if a new client will subscribe (yes/no) a term deposit (variable y).

1. Right-click the Raw button to save the sample training data file to your local computer from here: [bank-full.csv](#) 
2. Right-click the Raw button to save the sample payload data file to your local computer from here: [bank-payload.csv](#) 

If you preview the sample data, you can see it is *structured* demographic data in rows and columns, and saved in a .csv file.

age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no

## Creating and training the AutoAI experiment

In this section, you will define and run the experiment on the banking data to generate pipelines, or model candidates.

### Define the experiment

1. From the **Assets** page of your project, click **Add to project**, then click **AutoAI experiment**.
2. Name your experiment and add an optional description.
3. Accept the default compute configuration and click **Create**.
4. You are prompted to add your data for the experiment. Browse for and upload the data file *bank-full.csv*.

### What do you want to predict?

After adding the data, you choose a prediction column, which represents the problem you are trying to solve with the experiment. For this experiment, we want to know if a new bank customer will subscribe to a bank promotion, represented by the column labeled **y**.

1. Select **y** as the column to predict. You can see that when you choose a column to predict, AutoAI selects a model type that matches the data. AutoAI analyzes your data and determines that the **y** column contains Yes/No information, making this data suitable for a binary classification model.
2. Click **Run experiment**.

### Select prediction column

DATA SOURCE bank-full.csv

Column name	Type
pdays	Integer
previous	Integer
poutcome	String
y	String

Prediction column: y

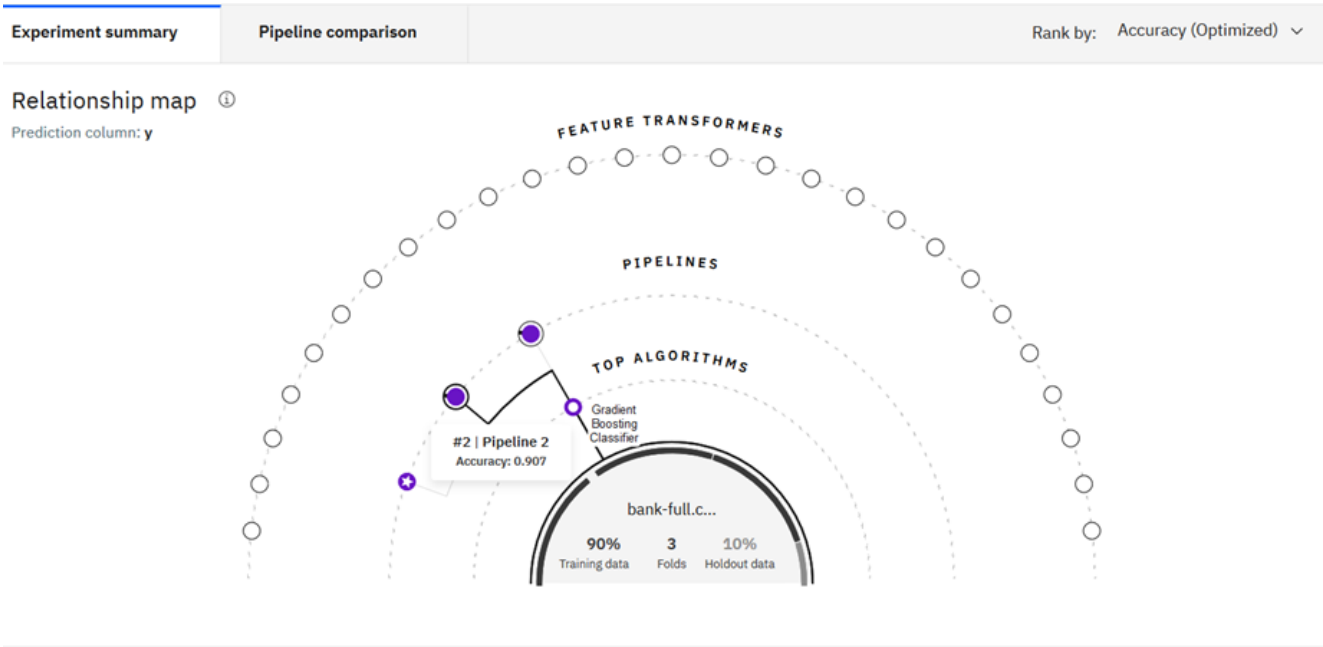
PREDICTION TYPE	POSITIVE CLASS	OPTIMIZED METRIC
Binary Classification ⓘ	Yes	Accuracy ⓘ

Experiment settings ⚙️
Run experiment ▶️

### Review the pipelines

As AutoAI runs the experiment, it generate a set of pipelines, ranked according to how they perform with certain metrics or optimizations.

As the experiment runs, an infographic shows the algorithms applied to build each pipeline. Hover over nodes in the infographic to get information about the pipeline.



For a list of algorithms, or estimators, available with each machine learning technique in AutoAI, see: [AutoAI implementation detail](#)

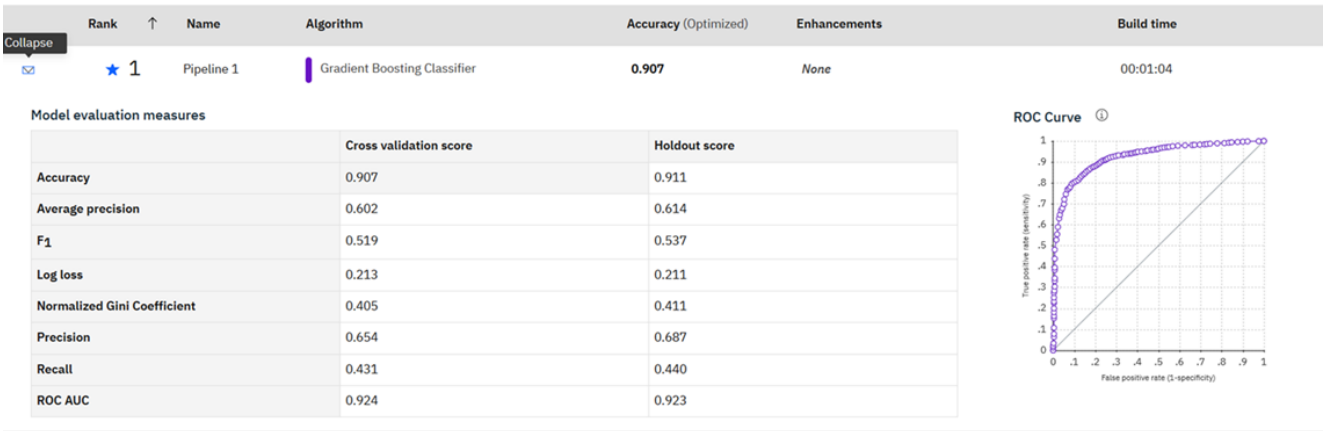
Choose a pipeline

Once the pipeline creation is complete, you can view and compare the ranked pipelines in a leaderboard.

Pipeline leaderboard

	Rank	↑	Name	Algorithm	Accuracy (Optimized)	Enhancements	Build time
>	★ 1		Pipeline 1	Gradient Boosting Classifier	0.907	None	00:01:04
>	2		Pipeline 2	Gradient Boosting Classifier	0.907	HPO-1	00:03:46
>	3		Pipeline 3	Gradient Boosting Classifier	0.906	HPO-1 FE	00:33:03

Expand a pipeline to view details about it.



You can also click **Pipeline comparison** to view differences between pipelines. When you are done reviewing the pipelines, choose one to save as a model.


1. Choose **Save as model** from the action menu for Pipeline 1. This saves the pipeline as a machine learning asset in your project.
2. Click **View in project** from the notification to open the project and see the details for the saved model.

## Deploy the trained model

From the model details page:

1. Click **Promote to deployment space**. If there is no deployment space associated with your project, you are prompted to create one now. Follow the steps to create the space, then promote the model again.
2. After you promote the model, click **go to deployment space** from the notification.

From the **Assets** page:

1. Click the deployment icon  next to the model name.
2. In the page that opens, fill in the fields:
  - Specify a name for the deployment.
  - Select "Online" as the **Deployment type**.
  - Click **Create**.

When the deployment status changes to *Deployed*, click on the deployment name to view the deployment details page.

## Step 3: Test the deployed model

You can test the deployed model from the deployment details page.

On the **Test** tab of the deployment details page, fill out the form with test values and click **Predict** to generate a prediction. For example:

Bank promotion deployment ✔ Deployed Online

API reference **Test**

Enter input data

age

37

job

management

marital

married

education

secondary

▼

Predict

Result

```
0 {
1   "predictions": [
2     {
3       "fields": [
4         "prediction",
5         "probability"
6       ],
7       "values": [
8         [
9           "no",
10          [
11            0.9507308204357073,
12            0.04926917956429266

```

The resulting prediction indicates that a customer with the attributes entered has a low probability of signing up for the bank promotion.

## Creating a batch deployment

To process a batch of inputs and have the output written to a file instead of displayed in real time, create a batch deployment.

## Step 1: Upload the input data

For a batch deployment, you provide input data, also known as the model *payload*, in a CSV file. The data should be structured like the training data, with the same column headers. The batch job will process each row of data and create a corresponding prediction.

For this tutorial, you will use the payload data *bank-payload.csv* that you downloaded as part of the tutorial setup. When you deploy a model, you can add the payload data to a project, upload it directly to a space, or link to the data in a storage repository such as a Cloud Object Storage bucket. In this case, you will upload the file directly to the deployment space.

From the **Assets** page of the deployment space:

1. Click **Add to space** then choose **Data**
2. Upload the file *bank-payload.csv* file that you saved locally.

## Step 2: Create the batch deployment

Now you can define the batch deployment.

1. Click the deployment icon next to the model name.
2. In the page that opens, fill in the fields:
  - Specify a name for the deployment.
  - Select "Batch" as the **Deployment type**.
  - Choose the smallest hardware specification.
  - Click **Create**.

## Step 3: Create the batch job:

The batch job executes the deployment. To create the job you specify the input data and the name for the output file. You can set up a job to run on a schedule, or run immediately.

1. Click **Create job**.
2. Specify the input file: *bank-payload.csv*.
3. Name the output file: *bank-tutorial-output*
4. Click **Create and run** to run the job immediately.

## Step 4: View the output

When the deployment status changes to *Deployed*, return to the **Assets** page for the deployment space. You will see that the file *bank-tutorial-output.csv* was created and added to your assets list.

Click the download icon next to the output file and open the file in an editor. You can review the prediction results for the customer information submitted for batch processing.

prediction	probability				
no	[0.978688135959038, 0.02131186404096194]				
no	[0.9676568367404358, 0.032343163259564246]				
no	[0.9215355424756787, 0.07846445752432127]				

For each case, the prediction returned indicates these customers are unlikely to subscribe to the bank promotion.



# Building an AutoAI model (Version 2.0)

---

AutoAI automatically prepares data, applies algorithms, and attempts to build model pipelines best suited for your data and use case. This topic describes how to generate the model pipelines.

Follow these steps to upload data and have AutoAI create the best model for your data and use case.

1. [Collect your input data](#)
2. [Open the AutoAI tool](#)
3. [Specify details of your model and training data and launch AutoAI](#)
4. [View the results](#)

## Collect your input data

---

Collect your input data in a CSV file. Where possible, AutoAI will transform the data and impute missing values.

**Note:** You can use the IBM Watson Studio Data Refinery tool to prepare and shape your data.

## Open the AutoAI tool

---

For your convenience, your AutoAI model creation uses the default storage associated with your project to store your data and to save model results, so you do not have to set up any separate repositories.

1. Open a project. The project must have an associated deployment space or you will not be able to add an AutoAI experiment. If it does not, create an associated space before continuing.
2. Click **Add to project**.
3. Click **AutoAI Experiment**.

**Note:** After you create an AutoAI asset it will display on the Assets page for your project in the **AutoAI experiment** section, so you can return to it.

## Specify details of your experiment

---

1. Specify a name and description for your experiment.
2. Select a compute configuration and click **Create**. The compute configuration specifies the computing resources to allocate to running the experiment. Larger sizes will improve training speed and might be required for larger data sources, but will cost more than smaller configurations.
3. Choose data from your project or upload it from your file system, then press **Continue**. Data must be in a CSV file. You can click the Preview icon after the data source name to review your data. Note that if your data source is larger than 1GB, AutoAI will automatically select a sample size of 1GB to use for training.
4. Choose the **Column to predict** for the data you want the experiment to predict.
  - Based on analyzing a subset of the data set, AutoAI chooses a default model type: binary classification, multiclass classification, or regression. Binary is selected if the target column has two possible values, multiclass if it has a discrete set of 3 or more values, and regression if the target column is a continuous numeric variable. You can override this selection.
  - AutoAI chooses a default metric for optimizing. For example, the default metric for a binary classification model is Accuracy.
  - By default, ten percent of the training data is held out to test the performance of the model.
5. (Optional) Click **Experiment settings** to view or customize options for your AutoAI run. To edit the settings for your experiment, click:
  - **Data source**, where you can adjust:
    - whether to subsample data. If you have a large data set, you can choose to train with a representative sample of the data to speed up pipeline creation. You can specify whether subsampling should be done by a percentage of the training data or by a specified number of rows.

- the percentage of training data vs holdout data. Training data is used to train the model, and holdout data is withheld from training the model and used to measure the performance of the model.
- columns to include. You can choose to include columns with data that supports the prediction column, and exclude irrelevant columns to speed up pipeline performance.
- **Prediction settings**, where you can:
  - change the model type. AutoAI selects a model type that best suits a sampling of the data, but you can override it. For example, if the sample data for the prediction column contains only two types of values, AutoAI will choose binary classification as the model type. If you know there are more than two values in the column, you can override the setting and choose multiclass classification instead. For binary classification models you can also edit the positive class.
  - change the metric to be optimised for the experiment. **Note:** For a binary classification experiment, if you change the metric to *Precision*, *Average Precision*, *Recall*, or *F1*, a Positive Class is required. Confirm that the Positive Class is correct or the experiment might generate inaccurate results.
  - optionally specify which algorithms AutoAI should consider for pipeline creation. Only checked algorithms will be considered during the model selection phase of the experiment.
  - change the number of algorithms to use to create pipelines. By default, AutoAI will choose the top two performing algorithms of the ones it considers, and use those algorithms to generate 8 pipelines that you can view and compare, but you can change the number from 1 to 4. For example, if you select 3 algorithms, AutoAI will identify the top three performing algorithms and use them to generate a total of 12 pipelines that you can view, compare, and save as models. Note that more pipelines will increase the training time for the experiment and use more resources.
- **Runtime settings**, where you can review experiment settings.

Click **Run Experiment** to begin model pipeline creation.

An infographic shows you the creation of pipelines for your data. The duration of this phase depends on the size of your data set. A notification message informs you if the processing time will be brief or require more time. You can work in other parts of the product while the pipelines build.



Hover over nodes in the infographic to explore the factors that pipelines share as well as their unique properties. You can see the factors that pipelines share as well as the properties that make a pipeline unique. For a guide to the data in the infographic, click the Legend tab in the information pane. Or, to see a different view of the pipeline creation, click the Experiment details tab of the notification pane, then click **Switch views** to view the progress map. In either view, click a pipeline node to view the associated pipeline in the leaderboard.

## View the results

---

When the pipeline generation process completes, you can view the leading model candidates and evaluate them before saving a pipeline as a model.

### Next step

Follow the steps in [Selecting an AutoAI model](#) for details on how to evaluate the pipelines as model candidates, then save a model.

## Saving an AutoAI generated notebook (Version 2.0)

---

If you want to view the code that created a particular model pipeline, or interact with the model programmatically, you can save a model pipeline as a notebook.

**Note:** this feature is offered as a tech preview and is subject to change.

To save a pipeline as a notebook:

1. Complete your AutoAI experiment.
2. Select the pipeline you want to save in the leaderboard, and choose **Save as notebook** from the action menu for the pipeline.
3. Name your notebook, add an optional description, and save it.

Your notebook is added to the containing project as a new notebook asset. You can run, deploy, and score the notebook as you would any notebook model.

**Attention:** If you encounter errors when running an AutoAI notebook that indicate you are missing a library (for example, `libomp`), follow the instructions at the top of the notebook for installing the missing library. You must be connected to the internet to install a library package.

## What is saved with the notebook

---

AutoAI saves a representation of the saved pipeline by replacing the optimization steps with the transformers that AutoAI has configured. The code is based on an scikit-learn library. In the notebook, you can view:

- Annotated code with comments highlighting the pipeline hierarchy and the transformations applied for each step.
- Holdout scoring and cross-validation of the training data, with the underlying changes to some transformers in AutoAI libraries
- Markdown cells that you can edit in the notebook editor.

For example, at the top of the notebook, you can review the libraries used to create the pipeline:

```
import sklearn
import xgboost
import lightgbm
from sklearn.cluster import FeatureAgglomeration
import numpy
from numpy import nan, dtype, mean
import autoai_libs
from autoai_libs.sklearn.custom_scorers import CustomScorers
import sklearn.ensemble
from autoai_libs.cognito.transforms.transform_utils import TExtras, FC
from autoai_libs.transformers.exportable import * # temporary
from autoai_libs.utils.exportable_utils import * # temporary
from sklearn.pipeline import Pipeline
```

After you save a pipeline as a notebook, you can review the steps used to compose the pipeline. For more information on the estimators, or algorithms, and transformers that are applied to your data to create the pipeline, refer to [Implementation details](#).

## Using autoai-lib for Python (Version 2.0)

The autoai-lib library for Python contains a set of functions that help you to interact with IBM Watson Machine Learning AutoAI experiments. Using the `autoai-lib` library, you can review and edit the data transformations that take place in the creation of the pipeline.

## Installing autoai-lib for Python

Follow instructions for installing a Python library to install `autoai-lib`.

### The autoai-lib functions

The instantiated project object that is created after you have imported the `autoai-lib` library exposes these functions:

#### `autoai_libs.transformers.exportable.NumpyColumnSelector()`

Selects a subset of columns of a numpy array

Usage:

```
autoai_libs.transformers.exportable.NumpyColumnSelector(columns=None)
```

Option	Description
columns	list of column indices to select

#### `autoai_libs.transformers.exportable.CompressStrings()`

Removes spaces and special characters from string columns of an input numpy array X.

Usage:

```
autoai_libs.transformers.exportable.CompressStrings(compress_type='string',
dtypes_list=None, misslist_list=None, missing_values_reference_list=None,
activate_flag=True)
```

Option	Description
compress_type	type of string compression. 'string' for removing spaces from a string and 'hash' for creating an int hash. Default is 'string'. 'hash' is used when there are columns with strings and <code>cat_imp_strategy='most_frequent'</code>
dtypes_list	list containing strings that denote the type of each column of the input numpy array X (strings are among 'char_str', 'int_str', 'float_str', 'float_num', 'float_int_num', 'int_num', 'boolean', 'Unknown'). If None, the column types are discovered. Default is None.
misslist_list	list containing lists of missing values of each column of the input numpy array X. If None, the missing values of each column are discovered. Default is None.

Option	Description
<code>missing_values_reference_list</code>	reference list of missing values in the input numpy array X
<code>activate_flag</code>	flag that indicates that this transformer will be active. If False, transform(X) outputs the input numpy array X unmodified.

## **autoai\_libs.transformers.exportable.NumpyReplaceMissingValues()**

Given a numpy array and a reference list of missing values for it, replaces missing values with a special value (typically a special missing value such as np.nan).

Usage:

```
autoai_libs.transformers.exportable.NumpyReplaceMissingValues(missing_values,
filling_values=np.nan)
```

Option	Description
<code>missing_values</code>	reference list of missing values
<code>filling_values</code>	special value assigned to unknown values

## **autoai\_libs.transformers.exportable.NumpyReplaceUnknownValues()**

Given a numpy array and a reference list of known values for each column, replaces values that are not part of a reference list with a special value (typically np.nan). This is typically used to remove labels for columns in a test dataset that have not been seen in the corresponding columns of the training dataset.

Usage:

```
autoai_libs.transformers.exportable.NumpyReplaceUnknownValues(known_values_list=None,
filling_values=None, missing_values_reference_list=None)
```

<code>known_values_list</code>	reference list of lists of known values for each column
<code>filling_values</code>	special value assigned to unknown values
<code>missing_values_reference_list</code>	reference list of missing values

t

## **autoai\_libs.transformers.exportable.boolean2float()**

Converts a 1-D numpy array of strings that represent booleans to floats and replaces missing values with np.nan. Also changes type of array from 'object' to 'float'.

Usage:

```
utoai_libs.tmsformers.exportble.boolean2flot(ctivte_flg=True)
```

Option	Description
<code>activate_flag</code>	flag that indicates that this transformer will be active. If False, transform(X) outputs the input numpy array X unmodified.

## autoai\_libs.transformers.exportable.CatImputer()

This is a wrapper for categorical imputer. Internally it currently uses sklearn [SimpleImputer](#)

Usage:

```
autoai_libs.transformers.exportable.CatImputer(strategy, missing_values,
sklearn_version_family=global_sklearn_version_family, activate_flag=True)
```

Option	Description
strategy	string, optional, default='mean'. The imputation strategy for missing values. -mean: replace using the mean along each column. Can only be used with numeric data. -median: replace using the median along each column. Can only be used with numeric data. -most_frequent: replace using most frequent value each column. Used with strings or numeric data. -constant: replace with fill_value. Can be used with strings or numeric data.
missing_values	number, string, np.nan (default) or None. The placeholder for the missing values. All occurrences of missing_values will be imputed.
sklearn_version_family	str indicating the sklearn version for backward compatibility with versions 019, and 020dev. Currently unused. Default is None.
activate_flag	flag that indicates that this transformer will be active. If False, transform(X) outputs the input numpy array X unmodified.

## autoai\_libs.transformers.exportable.CatEncoder()

This is a wrapper for categorical encoder. If encoding parameter is 'ordinal', internally it currently uses sklearn [OrdinalEncoder](#). If encoding parameter is 'onehot', or 'onehot-dense' internally it currently uses sklearn [OneHotEncoder](#)

Usage:

```
autoai_libs.transformers.exportable.CatEncoder(encoding, categories, dtype,
handle_unknown, sklearn_version_family=global_sklearn_version_family, activate_flag=True)
```

Option	Description
encoding	str, 'onehot', 'onehot-dense' or 'ordinal'. The type of encoding to use (default is 'ordinal') 'onehot': encode the features using a one-hot aka one-of-K scheme (or also called 'dummy' encoding). This creates a binary column for each category and returns a sparse matrix. 'onehot-dense': the same as 'onehot' but returns a dense array instead of a sparse matrix. 'ordinal': encode the features as ordinal integers. This results in a single column of integers (0 to n_categories - 1) per feature.
categories	'auto' or a list of lists/arrays of values. Categories (unique values) per feature: 'auto': Determine categories automatically from the training data. list: categories[i] holds the categories expected in the ith column. The passed categories must be sorted and should not mix strings and numeric values. The used categories can be found in the encoder.categories_ attribute.
dtype	number type, default np.float64 Desired dtype of output.
handle_unknown	'error' (default) or 'ignore'. Whether to raise an error or ignore if a unknown categorical feature is present during transform (default is to raise). When this parameter is set to "ignore" and an unknown category is encountered during transform, the resulting one-hot encoded columns for this feature will be all zeros. In the inverse transform, an unknown category will be denoted as None. Ignoring unknown categories is not supported for encoding='ordinal'.

Option	Description
sklearn_version_family	str indicating the sklearn version for backward compatibility with versions 019, and 020dev. Currently unused. Default is None.
activate_flag	flag that indicates that this transformer will be active. If False, transform(X) outputs the input numpy array X unmodified.

## autoai\_libs.transformers.exportable.float32\_transform()

Transforms a float64 numpy array to float32.

Usage:

```
autoai_libs.transformers.exportable.float32_transform(activate_flag=True)
```

Option	Description
activate_flag	flag that indicates that this transformer will be active. If False, transform(X) outputs the input numpy array X unmodified.

## autoai\_libs.transformers.exportable.FloatStr2Float()

Given numpy array X and dtypes\_list that denotes the types of its columns, it replaces columns of strings that represent floats (type 'float\_str' in dtypes\_list) to columns of floats and replaces their missing values with np.nan.

Usage:

```
autoai_libs.transformers.exportable.FloatStr2Float(dtypes_list,
missing_values_reference_list=None, activate_flag=True)
```

Option	Description
dtypes_list	list containing strings that denote the type of each column of the input numpy array X (strings are among 'char_str', 'int_str', 'float_str', 'float_num', 'float_int_num', 'int_num', 'boolean', 'Unknown').
missing_values_reference_list	reference list of missing values
activate_flag	flag that indicates that this transformer will be active. If False, transform(X) outputs the input numpy array X unmodified.

## autoai\_libs.transformers.exportable.NumImputer()

This is a wrapper for numerical imputer.

Usage:

```
autoai_libs.transformers.exportable.NumImputer(strategy, missing_values,
activate_flag=True)
```

Option	Description
strategy	num_imp_strategy : string, optional (default='mean'). The imputation strategy: - If 'mean', then replace missing values using the mean along the axis. - If 'median', then replace missing values using the median along the axis. - If 'most_frequent', then replace missing using the most frequent value along the axis.
missing_values	integer or 'NaN', optional (default='NaN'). The placeholder for the missing values. All occurrences of missing_values will be imputed: - For missing values encoded as np.nan, use the string value 'NaN'. - activate_flag: flag that indicates that this transformer will be active. If False, transform(X) outputs the input numpy array X unmodified.

## autoai\_libs.transformers.exportable.OptStandardScaler()

This is a wrapper for scaling of numerical variables. It currently uses sklearn [StandardScaler](#) internally.

Usage:

```
autoai_libs.transformers.exportable.OptStandardScaler(use_scaler_flag=True,
num_scaler_copy=True, num_scaler_with_mean=True, num_scaler_with_std=True)
```

Option	Description
num_scaler_copy	boolean, optional, default True. If False, try to avoid a copy and do in-place scaling instead. This is not guaranteed to always work. With in-place, for example, if the data is not a NumPy array or scipy.sparse CSR matrix, a copy may still be returned.
num_scaler_with_mean	boolean, True by default. If True, center the data before scaling. This does not work (and will raise an exception) when attempted on sparse matrices, because centering them entails building a dense matrix which in common use cases is likely to be too large to fit in memory.
num_scaler_with_std	boolean, True by default. If True, scale the data to unit variance (or equivalently, unit standard deviation).
use_scaler_flag	boolean, flag that indicates that this transformer will be active. If False, transform(X) outputs the input numpy array X unmodified. Default is True.

## autoai\_libs.transformers.exportable.NumpyPermuteArray()

Rearranges columns or rows of a numpy array based on a list of indices.

Usage:

```
autoai_libs.transformers.exportable.NumpyPermuteArray(permutation_indices=None, axis=None)
```

Option	Description
permutation_indices	list of indexes based on which columns will be rearranged
axis	0 permute along columns, 1, permute along rows

## Feature transformation

These methods apply to the feature transformations described in [AutoAI implementation details](#).



## **autoai\_libs.cognito.transforms.transform\_utils.TA1(fun, name=None, datatypes=None, feat\_constraints=None, tgraph=None, apply\_all=True, col\_names=None, col\_dtypes=None)**

For unary stateless functions, such as square, log, etc. use TA1.

Usage:

```
autoai_libs.cognito.transforms.transform_utils.TA1(fun, name=None, datatypes=None,
feat_constraints=None, tgraph=None, apply_all=True, col_names=None, col_dtypes=None)
```

Option	Description
fun	the function pointer
name	a string name that uniquely identifies this transformer from others
datatypes	a list of datatypes either of which are valid input to the transformer function (numeric, float, int, etc.)
feat_constraints	all constraints which must be satisfied by a column to be considered a valid input to this transform
tgraph	tgraph object should be the invoking TGraph() object. Note that this is optional and you may pass None, but that will result in some failure to detect some inefficiencies due to lack of caching
apply_all	only use applyAll = True. It means that the transformer will enumerate all features (or feature sets) that match the specified criteria and apply the provided function to each.
col_names	names of the feature columns in a list
col_dtypes	list of the datatypes of the feature columns

## **autoai\_libs.cognito.transforms.transform\_utils.TA2()**

For binary stateless functions, such as sum, product, use TA2.

Usage:

```
autoai_libs.cognito.transforms.transform_utils.TA2(fun, name, datatypes1,
feat_constraints1, datatypes2, feat_constraints2, tgraph=None, apply_all=True,
col_names=None, col_dtypes=None)
```

Option	Description
fun	the function pointer
name: a string name that uniquely identifies this transformer from others	
datatypes1	a list of datatypes either of which are valid inputs (first parameter) to the transformer function (numeric, float, int, etc.)
feat_constraints1	all constraints which must be satisfied by a column to be considered a valid input (first parameter) to this transform
datatypes2	a list of datatypes either of which are valid inputs (second parameter) to the transformer function (numeric, float, int, etc.)
feat_constraints2	all constraints which must be satisfied by a column to be considered a valid input (second parameter) to this transform
tgraph	tgraph object should be the invoking TGraph() object. Note that this is optional and you may pass None, but that will result in some missing out on inefficiencies due to lack of caching

Option	Description
apply_all	only use applyAll = True. It means that the transformer will enumerate all features (or feature sets) that match the specified criteria and apply the provided function to each.
col_names	names of the feature columns in a list
col_dtypes	list of the datatypes of the feature columns

## autoai\_libs.cognito.transforms.transform\_utils.TB1()

For unary state-based transformations (with fit/transform) use, such as frequent count.

Usage:

```
autoai_libs.cognito.transforms.transform_utils.TB1(tans_class, name, datatypes,
feat_constraints, tgraph=None, apply_all=True, col_names=None, col_dtypes=None)
```

Option	Description
tans_class	a class that implements fit() and transform() in accordance with the transformation function definition
name	a string name that uniquely identifies this transformer from others
datatypes	list of datatypes either of which are valid input to the transformer function (numeric, float, int, etc.)
feat_constraints	all constraints which must be satisfied by a column to be considered a valid input to this transform
tgraph	tgraph object should be the invoking TGraph() object. Note that this is optional and you may pass None, but that will result in some missing out on inefficiencies due to lack of caching
apply_all	only use applyAll = True. It means that the transformer will enumerate all features (or feature sets) that match the specified criteria and apply the provided function to each.
col_names	names of the feature columns in a list.
col_dtypes	list of the datatypes of the feature columns.

## autoai\_libs.cognito.transforms.transform\_utils.TB2()

For binary state-based transformations (with fit/transform) use, such as group-by.

Usage:

```
autoai_libs.cognito.transforms.transform_utils.TB2(tans_class, name, datatypes1,
feat_constraints1, datatypes2, feat_constraints2, tgraph=None, apply_all=True)
```

Option	Description
tans_class	a class that implements fit() and transform() in accordance with the transformation function definition
name	a string name that uniquely identifies this transformer from others
datatypes1	a list of datatypes either of which are valid inputs (first parameter) to the transformer function (numeric, float, int, etc.)
feat_constraints1	all constraints which must be satisfied by a column to be considered a valid input (first parameter) to this transform
datatypes2	a list of datatypes either of which are valid inputs (second parameter) to the transformer function (numeric, float, int, etc.)
feat_constraints2	

Option	Description
feat_constraints2	all constraints which must be satisfied by a column to be considered a valid input (second parameter) to this transform
tgraph	tgraph object should be the invoking TGraph() object. Note that this is optional and you may pass None, but that will result in some missing out on inefficiencies due to lack of caching
apply_all	only use applyAll = True. It means that the transformer will enumerate all features (or feature sets) that match the specified criteria and apply the provided function to each.

## autoai\_libs.cognito.transforms.transform\_utils.TAM()

For a transform that applies at the data level, such as PCA, use TAM.

Usage:

```
autoai_libs.cognito.transforms.transform_utils.TAM(tans_class, name, tgraph=None,
apply_all=True, col_names=None, col_dtypes=None)
```

Option	Description
tans_class	a class that implements fit() and transform() in accordance with the transformation function definition
name	a string name that uniquely identifies this transformer from others
tgraph	tgraph object should be the invoking TGraph() object. Note that this is optional and you may pass None, but that will result in some missing out on inefficiencies due to lack of caching
apply_all	only use applyAll = True. It means that the transformer will enumerate all features (or feature sets) that match the specified criteria and apply the provided function to each.
col_names	names of the feature columns in a list
col_dtypes	list of the datatypes of the feature columns

## autoai\_libs.cognito.transforms.transform\_utils.TGen()

TGen is a general wrapper and can be used for most functions (may not be most efficient though).

Usage:

```
autoai_libs.cognito.transforms.transform_utils.TGen(fun, name, arg_count, datatypes_list,
feat_constraints_list, tgraph=None, apply_all=True, col_names=None, col_dtypes=None)
```

Option	Description
fun	the function pointer
name	a string name that uniquely identifies this transformer from others
arg_count	number of inputs to the function, in this example it is 1, for binary, it will be 2, and so on
datatypes_list	a list of arg_count lists that correspond to the acceptable input data types for each parameter. In the above example, since arg_count=1, there is one list within the outer list, and it contains a single type called 'numeric'. In another case, it could be a specific case 'int' or even more specific 'int64', multiple of those
feat_constraints_list	a list of arg_count lists that correspond to some constraints that should be imposed on selection of the input features

Option	Description
tgraph	tgraph object should be the invoking TGraph() object. Note that this is optional and you may pass None, but that will result in some missing out on inefficiencies due to lack of caching
apply_all	only use applyAll = True. It means that the transformer will enumerate all features (or feature sets) that match the specified criteria and apply the provided function to each.
col_names	names of the feature columns in a list
col_datatypes	list of the datatypes of the feature columns

## autoai\_libs.cognito.transforms.transform\_utils.FS1()

Feature selection, type 1 (using pairwise correlation between each feature and target.)

Usage:

```
autoai_libs.cognito.transforms.transform_utils.FS1(cols_ids_must_keep,
additional_col_count_to_keep, ptype)
```

Option	Description
cols_ids_must_keep	serial numbers of the columns that must be kept irrespective of their feature importance
additional_col_count_to_keep	how many columns need to be retained
ptype	classification or regression

## autoai\_libs.cognito.transforms.transform\_utils.FS2()

Feature selection, type 2.

Usage:

```
autoai_libs.cognito.transforms.transform_utils.FS2(cols_ids_must_keep,
additional_col_count_to_keep, ptype, eval_algo)
```

Option	Description
cols_ids_must_keep	serial numbers of the columns that must be kept irrespective of their feature importance
additional_col_count_to_keep	how many columns need to be retained
ptype	classification or regression

## Selecting an AutoAI model (Version 2.0)

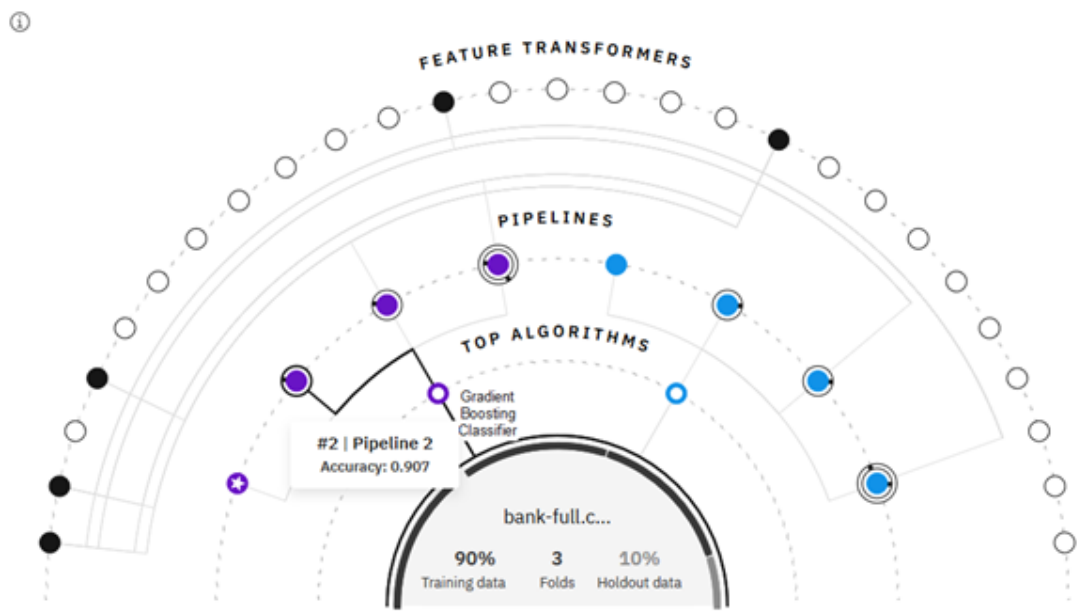
AutoAI automatically prepares data, applies algorithms, and attempts to build model pipelines best suited for your data and use case. This topic describes how to evaluate the model pipelines.

During AutoAI training, your data set is split to a training part and a hold-out part. The training part is used by the AutoAI training stages to generate the AutoAI model pipelines and cross-validation scores used to rank them. After AutoAI training, the hold-out part is used for the resulting pipeline model evaluation and computation of performance information such as ROC curves and confusion matrices, shown in the leaderboard. The training/hold-out split ratio is 90/10.

As the training progresses, you are presented with a dynamic infographic and leaderboard. Hover over nodes in the infographic to explore the factors that pipelines share as well as their unique properties. You can see the factors that pipelines share as well as the properties that make a pipeline unique. For a guide to the data in the infographic, click the Legend tab in the information pane. Or, to see a different view of the pipeline creation, click the Experiment details tab of the notification pane, then click **Switch views** to view the progress map. In either view, click a pipeline node to view the associated pipeline in the leaderboard. The leaderboard contains model pipelines ranked by cross-validation scores.

## View the pipeline transformations

Hover over a node in the infographic to view the transformations for a pipeline. The sequence of data transformations consists of a pre-processing transformer and a sequence of data transformers, if feature engineering was performed for the pipeline. The algorithm is determined by model selection and optimization steps during AutoAI training.



See [Implementation details](#) to review the technical details for creating the pipelines.

## View the leaderboard

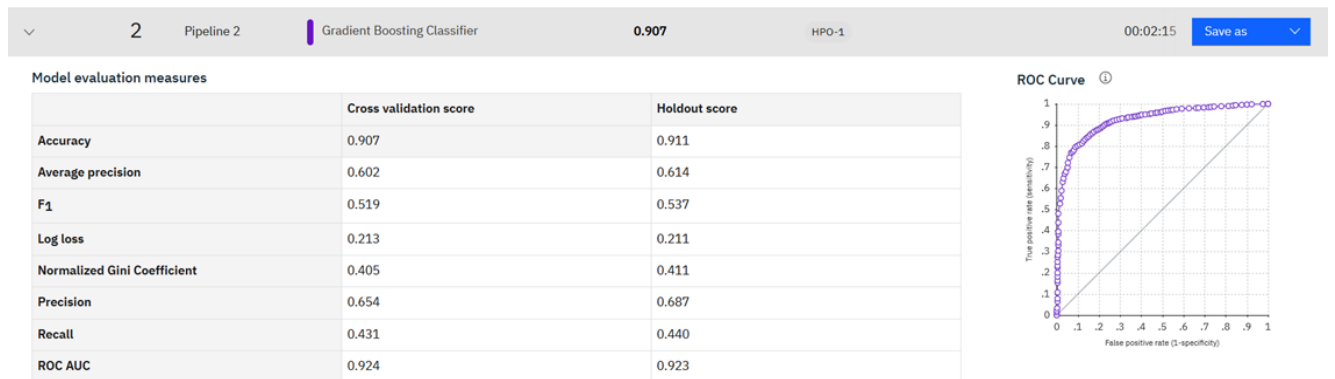
Each model pipeline is scored for a variety of metrics and then ranked. The default ranking metric for binary classification models is the area under the ROC curve, for multi-class classification models is accuracy, and for regression models is the root mean-squared error (RMSE). The highest-ranked pipelines are displayed in a leaderboard, so you can view more information about them. The leaderboard also provides the option to save select model pipelines after reviewing them.

Pipeline leaderboard

	Rank	↑	Name	Algorithm	Accuracy (Optimized)	Enhancements	Build time
>	★ 1		Pipeline 1	Gradient Boosting Classifier	0.907	None	00:00:33
>	2		Pipeline 2	Gradient Boosting Classifier	0.907	HPO-1	00:02:15
>	3		Pipeline 3	Gradient Boosting Classifier	0.906	HPO-1 FE	00:16:01
>	4		Pipeline 4	Gradient Boosting Classifier	0.906	HPO-1 FE HPO-2	00:10:14

You can evaluate the pipelines as follows:

- Click a pipeline in the leaderboard to view more detail about the metrics and performance.
- Click Compare to view how the top pipelines compare.
- Sort the leaderboard by a different metric.



## Viewing the confusion matrix

One of the details you can view for a pipeline for a binary classification experiment is a *Confusion matrix*.

The confusion matrix is based on the holdout data, which is the portion of the training dataset not used for training the model pipeline but only used to measure its performance on data that was not seen during training.

In a binary classification problem with a positive class and a negative class, the confusion matrix summarizes the pipeline model's positive and negative predictions in four quadrants depending on their correctness with respect to the positive or negative class labels of the holdout dataset.

For example, the Bank sample experiment seeks to identify customers that will take promotions offered to them. The confusion matrix for the top-ranked pipeline is:

### Confusion Matrix

TARGET : Y

Observed	Predicted		
	no	yes	Percent Correct
no	3,887	106	97.3%
yes	296	233	44.0%
Percent Correct	92.9%	68.7%	91.1%

The positive class is 'yes' (meaning a user will take the promotion), so you can see that the measurement of true negatives, that is, customers the model predicted correctly they would refuse their promotions, is fairly high.

Click the items in the navigation menu to view other details about the selected pipeline. For example, **Feature importance** shows which data features contribute most to your prediction output.

## Save a pipeline as a model

---

When you are satisfied with a pipeline, click **Save model** to save the candidate as a model to your project so you can test and deploy it.

## Next steps

---

Promote the trained model to a deployment space so that you can test it with new data and generate predictions.

## AutoAI implementation details (Version 2.0)

---

AutoAI automatically prepares data, applies algorithms, or estimators, and builds model pipelines best suited for your data and use case.

This topic describes some of these technical details that go into generating the pipelines:

- [Preparing and pre-processing the data](#)
- [Algorithms used for classification models](#)
- [Algorithms used for regression models](#)
- [Metrics by model type](#)
- [Data transformations](#)
- [AutoAI FAQ](#)

## Preparing the data for training

---

During automatic data preparation, AutoAI analyzes the training data and prepares it for model selection and pipeline generation. Data preparation involves these steps:

- [Feature column classification](#)
- [Feature engineering](#)
- [Pre-processing \(data imputation and encoding\)](#)

### Feature column classification

- Detects the types of feature columns and classifies them as categorical or numerical class
- Detects various types of missing values (default, user-provided, outliers)

### Feature engineering

- Handles rows for which target values are missing (drop (default) or target imputation)
- Drops unique value columns (except datetime and timestamps)
- Drops constant value columns

### Pre-processing (data imputation and encoding)

- Applies Sklearn imputation/encoding/scaling strategies (separately on each feature class)
- Handles labels of test set that were not seen in training set

## Algorithms used for classification models

---

These algorithms are the default algorithms used for automatic model selection for classification problems.

Algorithm	Description
Decision Tree Classifier	Maps observations about an item (represented in branches) to conclusions about the item's target value (represented in leaves). Supports both binary and multiclass labels, as well as both continuous and categorical features.

Algorithm	Description
Extra Trees Classifier	An averaging algorithm based on randomized decision trees.
Gradient Boosted Tree Classifier	Produces a classification prediction model in the form of an ensemble of decision trees. It only supports binary labels, as well as both continuous and categorical features.
LGBM Classifier	Gradient boosting framework that uses leaf-wise (horizontal) tree-based learning algorithm.
Logistic Regression	Analyzes a data set in which there are one or more independent variables that determine one of two outcomes. Only binary logistic regression is supported
Random Forest Classifier	Constructs multiple decision trees to produce the label that is a mode of each decision tree. It supports both binary and multiclass labels, as well as both continuous and categorical features.
XGBoost Classifier	Accurate sure procedure that can be used for classification problems. XGBoost models are used in a variety of areas including Web search ranking and ecology.

## Algorithms used for regression models

These algorithms are the default algorithms used for automatic model selection for regression problems.

Algorithm	Description
Decision Tree Regression	Maps observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It supports both continuous and categorical features.
Extra Trees Regression	An averaging algorithm based on randomized decision trees.
Gradient Boosting Regression	Produces a regression prediction model in the form of an ensemble of decision trees. It supports both continuous and categorical features.
LGBM Regression	Gradient boosting framework that uses tree-based learning algorithms.
Linear Regression	Models the linear relationship between a scalar-dependent variable y and one or more explanatory variables (or independent variables) x.
Random Forest Regression	Constructs multiple decision trees to produce the mean prediction of each decision tree. It supports both continuous and categorical features.
Ridge	Ridge regression is similar to Ordinary Least Squares but imposes a penalty on the size of coefficients.
XGBoost Regression	GBRT is an accurate and effective off-the-shelf procedure that can be used for regression problems. Gradient Tree Boosting models are used in a variety of areas including Web search ranking and ecology.

## Metrics by model type

The following metrics are available for measuring the accuracy of pipelines during training and when scoring data.

### Binary classification metrics

- Accuracy (default for ranking the pipelines)
- Roc auc
- Average precision
- F
- Negative log loss
- Precision



- Recall

## Multi-class classification metrics

Metrics for multi-class models can be adjusted to account for imbalances in labels. for example:

- Metrics with the *micro* qualifier calculate metrics globally by counting the total true positives, false negatives and false positives.
- Metrics with the *macro* qualifier calculates metrics for each label, and finds their unweighted mean. This does not take label imbalance into account.
- Metrics with the *weighted* qualifier calculate metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters *macro* to account for label imbalance; it can result in an F-score that is not between precision and recall.

These are the multi-class classification metrics:

- Accuracy (default for ranking the pipelines)
- F1
- F1 Micro
- F1 Macro
- F1 Weighted
- Precision
- Precision Micro
- Precision Macro
- Precision Weighted
- Recall
- Recall Micro
- Recall Macro
- Recall Weighted

## Regression metrics

- Negative root mean squared error (default for ranking the pipeline)
- Negative mean absolute error
- Negative root mean squared log error
- Explained variance
- Negative mean squared error
- Negative mean squared log error
- Negative median absolute error
- R2

## Metrics used for feature importance

Feature importance is calculated from the average of nine measures applied to the training data:

- Linear Correlation (f\_regression) metric
- Maximal Information Coefficient (MIC) metric
- Linear Regression (LR) metric
- L1 regularization metric (Lasso)
- Ridge metric
- RF metric
- Stability Selection
- Recursive Feature Elimination (RFE)
- Recursive Feature Elimination plus selection of best number of features

## Data transformations

---

For feature engineering, AutoAI uses a novel approach that explores various feature construction choices in a structured, non-exhaustive manner, while progressively maximizing model accuracy using reinforcement learning. This results in an optimized sequence of transformations for the data that best match the algorithms, or estimators, of the model selection step. This table lists some of the transformations used and some well-known conditions under which they are useful. This is not an exhaustive list of scenarios where the transformation is useful, as that can be complex and hard to interpret. Finally, the listed scenarios are not an explanation of how the transformations are selected. The selection of which transforms to apply is done in a trial and error, performance-oriented manner.

Name	Code	Function
Principal Component Analysis	pca	Reduce dimensions of data and realign across a more suitable coordinate system. Helps tackle the "curse of dimensionality" in linearly correlated data. It eliminates redundancy and separates significant signals in data.
Standard Scaler	stdscaler	Scales data features to a standard range. This helps the efficacy and efficiency of certain learning algorithms as well as other transformations such as PCA.
Logarithm	log	Reduces right skewness in features and make them more symmetric. Resulting symmetry in features helps algorithms understand the data better. Even scaling based on mean and variance is more meaningful on symmetrical data. Additionally, it can capture specific physical relationships between feature and target best described through a logarithm.
Cube Root	cbqrt	Reduces right skewness in data like logarithm, but is weaker than log in its impact, which might be more suitable in some cases. It is also applicable to negative or zero values to which log doesn't apply. Cube root can also change units such as reducing volume to length.
Square root	sqrtr	Reduces mild right skewness in data. It is weaker than log or cube root. It works with zeroes and reduces spatial dimensions such as area to length.
Square	squarer	Reduces left skewness to a moderate extent to make such distributions more symmetric. It can also be helpful in capturing certain phenomena such as super-linear growth.
Product	product	A product of two features can expose a non-linear relationship to better predict the target value than the individual values alone. For example, item cost into number of items sold is a better indication of the size of a business than any of those alone.
Numerical XOR	nxor	This transform helps capture "exclusive disjunction" type of relationships between variables, similar to a bitwise XOR, but in a general numerical context.
Sum	sum	Sometimes the sum of two features is better correlated to the prediction target than the features alone. For instance, loans from different sources, when summed up, provide a better idea of a credit applicant's total indebtedness.
Divide	divide	Division is a fundamental operand used to express quantities such as gross GDP over population (per capita GDP), representing a country's average lifespan better than either GDP alone or population alone.
Maximum	max	Take the higher of two values.
Rounding	round	This transformation can be seen as perturbation or adding some noise to reduce overfitting that might have been a result of inaccurate observations.
Absolute Value	abs	Consider only the magnitude and not the sign of observation. Sometimes, the direction or sign of an observation doesn't matter so much as the magnitude of it, such as physical displacement, while considering fuel or time spent in the actual movement.
Hyperbolic tangent	tanh	Non-linear activation function can improve prediction accuracy, similar to that of neural network activation functions.
Sine	sin	Can reorient data to discover periodic trends such as simple harmonic motions.
Cosine	cos	Can reorient data to discover periodic trends such as simple harmonic motions.
Tangent	tan	Trigonometric tangent transform is usually helpful in combination with other transforms.

Name	Code	Function
Feature Agglomeration	featurereagglomeration	Clustering different features into groups, based upon distance or affinity, provides ease of classification for the learning algorithm.
Sigmoid	sigmoid	Non-linear activation function can improve prediction accuracy, similar to that of neural network activation functions.
Isolation Forest	isolationforest	Performs clustering by using an Isolation Forest to create a new feature containing an anomaly score for each sample.

## AutoAI FAQs

The following are commonly asked questions about creating an AutoAI experiment.

### How many pipelines are created?

Two AutoAI parameters determine the number of pipelines:

- **max\_num\_daub\_ensembles:** Maximum number (top-K ranked by DAUB model selection) of the selected algorithm, or estimator types, for example LGBMClassifierEstimator, XGBoostClassifierEstimator, or LogisticRegressionEstimator to use in pipeline composition. The default is 1, where only the highest ranked by model selection algorithm type is used.
- **num\_folds:** Number of subsets of the full dataset to train pipelines in addition to the full dataset. The default is 1 for training the full data set. For each fold and algorithm type, AutoAI creates 4 pipelines of increased refinement, corresponding to:

1. Pipeline with default sklearn parameters for this algorithm type
2. Pipeline with optimized algorithm using HPO
3. Pipeline with optimized feature engineering
4. Pipeline with optimized feature engineering and optimized algorithm using HPO

The total number of pipelines generated is :

```
TotalPipelines= max_num_daub_ensembles * 4, if num_folds = 1:
```

```
TotalPipelines= (num_folds+1) * max_num_daub_ensembles * 4, if num_folds > 1 :
```

### What hyperparameter optimization is applied to my model?

AutoAI uses a model-based, derivative-free global search algorithm, called RBfOpt, which is tailored for the costly machine learning model training and scoring evaluations required by hyperparameter optimization (HPO). In contrast to Bayesian optimization, which fits a Gaussian model to the unknown objective function, RBfOpt fits a radial basis function mode to accelerate the discovery of hyper-parameter configurations that maximize the objective function of the machine learning problem at hand. This acceleration is achieved by minimizing the number of expensive training and scoring machine learning models evaluations and by eliminating the need to compute partial derivatives.

For each fold and algorithm type, AutoAI creates two pipelines that use HPO to optimize for the algorithm type.

- The first is based on optimizing this algorithm type based on the preprocessed (imputed/encoded/scaled) dataset (pipeline 2) above).
- The second is based on optimizing the algorithm type based on optimized feature engineering of the preprocessed (imputed/encoded/scaled) data set.

The parameter values of the algorithms of all pipelines generated by AutoAI is published in status messages.

For more details regarding the RbfOpt algorithm, see:

- [RbfOpt: A blackbox optimization library in Python](#)
- [An effective algorithm for hyperparameter optimization of neural networks. IBM Journal of Research and Development, 61\(4-5\), 2017](#)

## Deploying assets (Version 2.0)

---

Deployment is the final stage of the lifecycle of a model or script, where you run your models and code. Watson Machine Learning provides the tools you need to deploy an asset, such as an SPSS modeler flow, or a machine learning model depending on what tools are configured for your system.

To deploy an asset, you must have a *deployment space* where you can organize the assets you need to create and monitor deployments. A space contains an overview of deployment status, the deployable assets, deployments, associated input and output data, and the associated environments. A deployment makes a copy of a model or script available to test and use. For example, you can create a deployment for a machine learning model so you can submit new data to a model and get a score, or prediction back.



When you promote a model or script to a space, components required for a successful deployment, such as a training library, model definition, or environment definition are automatically promoted as well. After promoting a model and data assets to a deployment space, you can create a deployment in the space.

You can also create a deployment programmatically, in the model or function code, and view the deployment in the space.

### Next steps

---

- Find out how to [view and manage assets on deployment spaces](#).
- Find out how to [deploy a model from a deployment space](#).
- Find out how to [deploy a model from a notebook](#) using the Python client.

## Deployment spaces (Version 2.0)

---

You can use deployment spaces to deploy models and manage your deployments.

Deployment spaces allow you to create deployments for saved models and view and manage all of the activity and assets for the deployments, including data connections and connected data assets. You can:

- [View deployed assets](#)
- [Create a deployment space](#)
- [Promote assets to a space](#)
- [Import a PMML model to a space](#)
- [Create deployments](#)
- [Export a deployment space](#)

## Viewing spaces

---

You configure and manage the deployment of a set of related assets in a space. A space contains an overview of deployment status, the deployable assets, deployments, associated input and output data, and the associated environments.

- To view all deployment spaces that you can access, click **Deployment Spaces** on the Watson Studio navigation menu.

## Creating a deployment space

---

A deployment space can only be associated with one project. If you have not already associated a deployment space with a project, when you attempt to promote an asset from a project, you are prompted to create a new space or choose an existing space to be associated with your project.

You can create a space from the **Overview** or **Settings** page of a project, or by following these steps:

1. Click **Deployment Spaces** on the Watson Studio navigation menu.
2. Click **Create new deployment space**
3. Choose whether to create a new space or import an existing one.
  - Choose **Empty Space** to create a new space.
  - Choose **Import space** to import a space that was created on IBM Watson Studio Desktop and saved as a .zip file. You can add the file from your file system. **Tip:** If you get an error importing a space file, try clearing your browser cookies then try again.
4. Enter the details for the space, then click **Create**.

View details about the space, including the space ID, from the **Settings** tab.

## Promoting assets to a deployment space

---

The following assets can be promoted to a deployment space:

- Saved models
- Data assets for use in deployments
- Connections defined in your project
- Functions
- Scripts

Promote or add assets to a deployment space in the following ways:

- From the space, choose **Add to space** and choose an asset type, such as data or machine learning model. Follow the prompt to upload or add the asset.
- From the project **Assets** page, choose **Promote** from the action menu for the asset to promote it to a deployment space.

For details on adding data to a space, see [Adding data sources to a space](#).

## Importing a model into the space

---

If you have a trained model saved in Predictive Model Markup Language (PMML) format in an .xml file, you can import that model directly into a deployment space and create a deployment for the model.

1. From the **Assets** tab of your deployment space, click **Add to space** and choose Watson Machine Learning model.
2. In the **Import model** dialog that displays, enter a name and optional description for the model.
3. Drop or upload the PMML file in the **Model content** box, then click **Import**.
4. Create a deployment for the model.

## Notes and restrictions

- Online is the only supported deployment type for PMML models.
- PMML models cannot be used in an SPSS stream flow.
- The PMML file must not contain a prolog. Depending on the library you are using when you save your model, a prolog might be added to the top of the file by default. For example, if your file contains a prolog string such as `spark-mllib-lr-model-pmml.xml`, remove the string before you import the PMML file to the deployment space.

## Creating a new deployment

---

When you promote a model to a space, components required for a successful deployment, such as a training library, model definition, or pipeline definition are automatically promoted as well. After promoting a model and data assets to a deployment space, you can create a deployment in the space.

1. Click the name of the saved model in the deployment space.
2. Click the Deployments tab.
3. Click **New Deployment** to create a deployment.
4. Choose the deployment type and fill out the specifics for the deployment. For details, see [deploying from a space](#).

## Exporting a deployment space

---

You can export a deployment space so that you can share the space with others or reuse the assets in another space.

To export a space:

1. From the space, click the export space icon.
2. Click **New export file**, specify a file name and optional description.
3. Select the assets you want to export with the space.
4. Click **Create** to create the export file.
5. Click **Download** to save the file.

You can reuse this space by choosing **Create a space from a file** when you create a new space.

## Adding data sources to a space (Version 2.0)

---

Add data sources to a deployment space to use with batch deployment jobs. Data can be:

- A data file such as a .csv file
- A connection to data that resides in a repository such as a database.
- Connected data that resides in a storage bucket, such as a data file that in a Cloud Object Storage bucket.

**Note:** although you can promote any kind of data connection to a space, where you use the connection is governed by factors such as model and deployment type. For example, you can access any of the connected data via a script, but in batch deployments you are limited to particular types of data, as listed in [Batch deployment details by framework](#). If you are connected to Watson Machine Learning Server 2.0, you can promote the connection asset from a project to a deployment space. The Watson Machine Learning Server needs to have the same access as the Watson Studio Desktop computer. For example, if the Watson Studio Desktop computer is behind a firewall, the Watson Machine Learning Server needs to be behind the firewall as well.

**Restriction:** From a space, you can only download connected data from Cloud Object Storage (COS) with HMAC credentials supplied. Downloading unsupported types of connected data can result in an error.

Data added to a space is managed in a similar way to data added to a project. For example:

- Adding data to a space creates a new copy of the asset and its attachments within the space, maintaining a reference back to the project asset. If an asset such as a data connection requires access credentials, they persist and are the same whether you are accessing the data from a project or from a space.
- Details about the data connection that you can edit from the project you can also edit from the space.
- Data assets are stored in a space in the same way they are stored in a project, using the same file structure for the space as what is used for the project.

You can add data to a space in one of these ways:

- Promote a data source, such as a file or connection, from an associated project.
- Add a data file, connection, or connected data directly to a space
- Save a data asset to a space programmatically

## Promoting data sources from a project

---

To promote data from a project:

1. Save a data source, data connection, or connected data to a project.
2. From the project **Assets** page, choose **Promote** from the action item for the data asset to promote it to a deployment space.

The data asset displays as an asset in the space and is available for use as an input data source in a batch deployment job.

## Adding data to a space

---

To add data directly to a space:

1. From the Assets page of the deployment space, click **Add to space**.
2. Choose the type of data asset to add:
  - **Data** to specify a file to upload.
  - **Connection** to specify a connection to a data repository such as DB2
  - **Connected data** to connect to data in a storage object such as a Cloud Object Storage bucket.
3. Complete the steps to add the data.

The data asset displays as an asset in the space and is available for use as an input data source in a batch deployment job.

## Collaborator permissions for spaces (Version 2.0)

---

When you add a collaborator to a deployment space, you specify which actions the user can do by assigning an access level. If you have the Admin role, you can change the access level of an existing collaborator on the **Access Control** page of the space.

To add one or more collaborators to a deployment space:

1. Open the deployment space and click the **Access Control** tab.
2. Click **Add collaborators**. From the panel that opens you can:
  - Search for existing Watson Studio users and choose a role. They are added immediately.
  - Enter email address of new users and choose a role.
  - Click **Select project collaborators** to add the collaborators from the associated project to the deployment space.

These roles provide these permissions for deployment spaces:

Enabled permission	Viewer	Editor	Admin
View assets and deployments	✓	✓	✓
Comment	✓	✓	✓
Monitor	✓	✓	✓
Test model deployment API	✓	✓	✓
Find implementation details	✓	✓	✓
Configure deployments		✓	✓
Batch score		✓	✓
Update assets		✓	✓
Import assets		✓	✓
Deploy assets		✓	✓
Remove assets		✓	✓
Remove deployments			✓
View spaces/members	✓	✓	✓
Associate space with project		✓	✓
Delete space			✓

## Creating deployments from a space (Version 2.0)

---

Use the tools available from a deployment space to deploy and run models and scripts. The types of deployment available for a model and the type of input supported depends on the model framework.

The types of deployments are:

- [Web service](#) Also called Online, this deployment loads the model or Python code when the deployment is created to generate predictions online, in real time.
- [Batch](#) to process batches of input submitted from a data file, a connection to a data repository such as a database, or connected data in a storage repository such as a Cloud Object Storage bucket. Batch deployments can be run on demand or on a schedule.
- [Core ML](#) downloads the code required to deploy on an iOS device.

## Creating an online deployment (Version 2.0)

---

Create an online , or Web service deployment to load a model or Python code when the deployment is created to generate predictions online, in real time.

For example, if you create a classification model to test whether a new customer is likely to participate in a sales promotion, you can create an online deployment for the model, and enter the new customer data to get an immediate prediction.

### Create the deployment

---

1. From the deployment space, click the name of the saved model you want to deploy. The model detail page opens.
2. From the **Deployments** tab, click **Add new deployment**.
3. Choose **Online** as the deployment type.
4. Provide a name and optional description for the deployment, then click **Create** to create the deployment.
5. When the deployment status shows **Deployed**, click the name of the deployment to view the deployment details.



## Working with an online deployment

---

There are two tabs for an online deployment:

- **API Reference** displays the API endpoint you will need to access the deployment programmatically. You will need the endpoint to use the deployment in an application. There are also code snippets in a variety of programming languages that illustrate how to access the deployment.
- **Test** provides a place where you can enter data and get a prediction back from the deployed model. If your model has a defined schema, you will see a form where you can submit data that matches the schema and get a score, or prediction, back. Otherwise, you can submit input data in JSON format and get a score back.

### Sample deployment code

When you submit JSON code as the payload, or input data, for a deployment, your input data must match the schema of the model. The "fields" must match the column headers for the data, and the "values" must contain the data, in the same order. Use this format:

```
{"input_data":[{"fields": [<field1>, <field2>, ...],
"values": [[<value1>, <value2>, ...]]
}]}
```

For example:

```
{"input_data":[{"fields": [
"PassengerId", "Pclass", "Name", "Sex", "Age", "SibSp", "Parch", "Ticket", "Fare", "Cabin", "Embarked"],
"values": [[1,3,"Braund, Mr. Owen Harris",0,22,1,0,"A/5 21171",7.25,null,"S"]]
}]}
```

#### Notes:

- Note that all strings are enclosed in double-quotes. The Python notation for dictionaries looks very similar, but Python strings in single-quotes are not accepted in the JSON data.
- Missing values can be indicated with `null`.

### Creating multiple copies of an online deployment

When you deploy a model from a deployment space or programmatically, a single copy of the model is deployed by default. Select the number of replicas that should be created for the deployment. Additional replicas allow for a larger volume of scoring requests.

To view or run a working sample of scaling a deployment programmatically, you can download a [notebook example](#).

The following example uses the Python client API to set the number of nodes to 3.

```
change_meta = {
    client.deployments.ConfigurationMetaNames.HARDWARE_SPEC: {
        "name": "S",
        "num_nodes": 3
    }
}

client.deployments.update(<deployment_id>, change_meta)
```

Note that the `HARDWARE_SPEC` value includes a name because the API requires a name or an id to be provided. However, this argument is actually disregarded for online deployments.

## Updating a deployed asset

---

After you create an online deployment, you can update the deployed asset from the same endpoint. For example, if you have a better performing model, you can replace the deployed model with the improved version. Once the update

is complete, the new model will be available from the REST API endpoint.

Before you update an asset, make sure these conditions are true:

- The framework of the new model is compatible with the existing deployed model
- The input schema exists and matches for the new and deployed model

**Caution:** Failure to follow these conditions can result in a failed deployment.

## Update an asset from the deployment space

1. From the **Deployments** tab of your deployment space, click the action menu for the deployment and choose **Update asset**.
2. Drag and drop or upload a new asset to replace the deployed asset.
3. Wait for the notification message that confirms a successful update, then test the new deployment.

## Update an asset using the Patch API command

Use the `Patch` command to update any supported asset except for RShiny scripts.

This sample shows how to patch a model for an online deployment with id: 6f01d512-fe0f-41cd-9a52-1e200c525c84 where the input payload is:

```
[
  {
    "op": "replace",
    "path": "/asset",
    "value": {
      "id": "6f01d512-fe0f-41cd-9a52-1e200c525c84",
      "rev": "1"
    }
  }
]
```

A successful output response will look like this:

```
{
  "entity": {
    "asset": {
      "href": "/v4/models/6f01d512-fe0f-41cd-9a52-1e200c525c84?space_id=f2ddb8ce-7b10-4846-9ab0-62454a449802",
      "id": "6f01d512-fe0f-41cd-9a52-1e200c525c84"
    },
    "custom": {
    },
    "description": "Test V4 deployments",
    "name": "test_v4_dep_online_space_hardware_spec",
    "online": {
    },
    "space": {
      "href": "/v4/spaces/f2ddb8ce-7b10-4846-9ab0-62454a449802",
      "id": "f2ddb8ce-7b10-4846-9ab0-62454a449802"
    },
    "space_id": "f2ddb8ce-7b10-4846-9ab0-62454a449802",
    "status": {
      "online_url": {
        "url": "https://example.com/v4/deployments/349dc1f7-9452-491b-8aa4-0777f784bd83/predictions"
      },
      "state": "updating"
    }
  },
  "metadata": {
    "created_at": "2020-06-08T16:51:08.315Z",
    "description": "Test V4 deployments",
  }
```

```

    "guid": "349dc1f7-9452-491b-8aa4-0777f784bd83",
    "href": "/v4/deployments/349dc1f7-9452-491b-8aa4-0777f784bd83",
    "id": "349dc1f7-9452-491b-8aa4-0777f784bd83",
    "modified_at": "2020-06-08T16:55:28.348Z",
    "name": "test_v4_dep_online_space_hardware_spec",
    "parent": {
      "href": ""
    },
    "space_id": "f2ddb8ce-7b10-4846-9ab0-62454a449802"
  }
}

```

#### Notes:

- The initial state for the PATCH API output is "updating." Keep polling the status until it changes to "ready," then retrieve the deployment meta.
- Only the `ASSET` attribute can be specified for the asset patch. Changing any other attribute will result in an error.
- The schema of the current model and the model being patched is compared to the deployed asset and a warning message is returned in the output of the Patch request API if the two don't match. For example, if a mismatch is detected, you will see this in the output response:

```

"status": {
  "message": {
    "text": "The input schema of the asset being patched does not match with the currently
deployed asset. Please ensure that the score payloads are up to date as per the asset
being patched."
  },

```

## Creating a batch deployment (Version 2.0)

---

A batch deployment processes input data from a file, data connection, or connected data in a storage bucket, and writes the output to a file.

### Before you begin

---

1. Save a model to a deployment space.
2. Promote or add the input file for the batch deployment to the space. For details on promoting an asset to a space, see [Deployment spaces](#).

### Structuring the input data

---

How you structure the input data, also known as the payload, for the batch job depends on the framework for the asset you are deploying. For supported input type by framework, see [Batch deployment details](#).

A .csv input file or other structured data formats should be formatted to match the schema of the asset. List the column names (fields) in the first row and values to be scored in subsequent rows. For example:

```

PassengerId, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked
1,3,"Braund, Mr. Owen Harris",0,22,1,0,A/5 21171,7.25,,S
4,1,"Winslet, Mr. Leo Brown",1,65,1,0,B/5 200763,7.50,,S

```

A JSON input file should provide the same information on fields and values, using this format:

```

{"input_data": [{
  "fields": [<field1>, <field2>, ...],
  "values": [[<value1>, <value2>, ...]]
}]}

```

For example:

```
{
  "input_data": [{
    "fields": [
      "PassengerId", "Pclass", "Name", "Sex", "Age", "SibSp", "Parch", "Ticket", "Fare", "Cabin", "Embarked"
    ],
    "values": [
      [1, 3, "Braund, Mr. Owen Harris", 0, 22, 1, 0, "A/5 21171", 7.25, null, "S"],
      [4, 1, "Winselt, Mr. Leo Brown", 1, 65, 1, 0, "B/5 200763", 7.50, null, "S"]
    ]
  }]
}
```

## Creating a batch deployment job

1. From the deployment space, click the name of the saved model you want to deploy. The model detail page opens.
2. Click **Create deployment**.
3. Choose **Batch** as the deployment type and enter a name for your deployment.
4. Choose a hardware definition based on the CPU and RAM that should be allocated for this deployment.
5. Click **Create** to create the deployment.
6. When the status changes to *Deployed*, click the deployment name, then click **Create job** to configure how to run the deployment.
7. Define the details for the job, such as name and an optional description for the job.
8. (Optional) Schedule when the batch job should run. Scheduled jobs display on the **Jobs** tab of the deployment space. You can edit the schedule and other options from the Jobs tab.
9. Specify the input data source or sources. Input data depends on what you are deploying:
  - Choose **Inline data** to enter the payload in JSON format.
  - Choose **Data asset** to specify an input data source. The source can be a data source file you promoted to the space, a connection to a data source, or connected data in a storage bucket.
  - Choose multiple input data sources to match a model, such as an SPSS modeler flow or an AutoAI data join experiment, which has multiple inputs.
10. If you specify a data asset, provide a name and optional description for the output file that will contain the results or choose a connected data asset where you want to write the results.
11. (Optional) If you are deploying a Python script, you can enter environment variables to pass parameters to the job.
12. Click **Create** to create the job, or **Create and run** to create the job and run it immediately. Results of the run are written to the specified output file and saved as a space asset.

**Note:** If you select to schedule a job to run every day of the week excluding given days, you might notice that the scheduled job does not run as you would expect. The reason is due to a discrepancy between the timezone of the user who creates the schedule, and the timezone of the master node where the job runs. This issue only exists if you exclude days of a week when you schedule to run a job.

## Batch scoring with connected data

When you create a batch deployment job programmatically, you can specify a direct connection to a data input and output source such as a DB2 database. When you create a batch deployment job from a space, however, you cannot access direct connections but you can connect to data in a storage repository such as a Cloud Object Storage bucket.

## Using connected data for an SPSS modeler flow job

An SPSS modeler flow can have a number of input and output data nodes. When connecting to DB2 or DashDB as an input and output data source, note that the connection details are selected from the input and output data reference, but the input and output table names are selected from the SPSS model stream file.

To perform batch deployment of SPSS model using a DB2 or DB2 Warehouse connection, the modeler stream Input and Output nodes should both be Data Asset nodes. In SPSS Modeler, configure the Data Asset nodes must with the table names that will be used later for job predictions. This must be done before saving the model to Watson Machine Learning.

When creating the deployment job for the SPSS model, make sure the type of data sources are the same for input and output. The configured table names from the model stream will be passed to the batch deployment and the input/output table names provided in the connected data will be ignored.

To perform batch deployment of SPSS model using a Cloud Object Storage (COS) connection, make sure the SPSS model stream has single input and output data asset nodes.

## Batch deployment details (Version 2.0)

---

You can create a batch deployment using any of these interfaces:

- Watson Studio user interface, from an Analytics deployment space
- Watson Machine Learning Python Client
- Watson Machine Learning REST APIs

### Data sources

---

The input data sources for a batch deployment job differ by framework. Input data can be supplied to a batch job as:

- **Inline data** - In this method, the input data for batch processing is specified in the batch deployment job's payload, for example, as a value for parameter `scoring.input_data`. Once the batch deployment job is completed, the output of the batch deployment job is written to the corresponding job's metadata parameter `scoring.predictions`
- **Data reference** - in this method, the input and output data for batch processing can be stored in a remote data source like a Cloud Object Storage bucket, an SQL/no-SQL database, or as a local or managed data asset in a deployment space. Details for data references include:
  - `input_data_references.type` and `output_data_reference.type` must be `data_asset`
  - The references to input data must be specified as a `/v2/assets` href in the `input_data_references.location.href` parameter in the deployment job's payload. The data asset specified here can be a reference to a local or connected data asset.
  - If the batch deployment job's output data has to be persisted in a remote data source, the references to output data must be specified as a `/v2/assets` href in `output_data_reference.location.href` parameter in the deployment job's payload.
  - If the batch deployment job's output data has to be persisted in a deployment space as a local asset, `output_data_reference.location.name` must be specified. Once the batch deployment job is completed successfully, the asset with the specified name will be created in the space.
  - If the output data references where the data asset is in a remote database, you can specify if the batch output should be appended to the table or if the table is to be truncated and output data updated. Use the `output_data_references.location.write_mode` parameter to specify the values `truncate` or `append`. Note the following:
    - Specifying `truncate` as value truncates the table and inserts the batch output data.
    - Specifying `append` as value appends the batch output data to the remote database table.
    - `write_mode` is applicable only for `output_data_references` parameter.
    - `write_mode` is applicable only for remote database related data assets. This parameter will not be applicable for a local data asset or a COS-based data asset.
  - Any input and output data asset references must be in the same space id as the batch deployment.
  - If the connected data asset references a Cloud Object Storage instance as source, for example, a file in a Cloud Object Storage bucket, you must supply the HMAC credentials for the COS bucket. Include an

Access Key and a Secret Key to your IBM Cloud Object Storage connection to enable access to the stored files.

## Specifying the compute requirements for the batch deployment job

---

The compute configuration for a batch deployment refers to the CPU and memory size allocated for a job. This information must be specified in the `hardware_spec` API parameter of either of these:

- deployments payload
- deployment jobs payload.

In the case of a batch deployment of an AutoAI model, the compute configuration must be specified in `hybrid_pipeline_hardware_specs` instead of `hardware_spec` parameter.

The compute configurations must be a reference to a predefined hardware specification. You can specify a hardware specification by name or id, using the id or name of the hardware specification with `hardware_spec` or `hybrid_pipeline_hardware_specs` (for AutoAI). The list and details about the predefined hardware specifications can be accessed through the Watson Machine Learning Python client or the Watson Machine Learning REST APIs.

### Predefined hardware specifications

These are the predefined hardware specifications available by model type.

### Watson Machine Learning models

Size	Hardware definition
XS	1 CPU and 4 GB RAM
S	2 CPU and 8 GB RAM
M	4 CPU and 16 GB RAM
ML	4 CPU and 32 GB RAM
L	8 CPU and 32 GB RAM
XL	8 CPU and 64 GB RAM

## Steps for submitting a batch deployment job (overview)

---

1. Create a deployment of type batch.
2. Submit a deployment job with reference to the batch deployment.
3. Poll for the status of the deployment job by querying the details of the corresponding deployment job via Watson Machine Learning Python client, REST APIs, or via the deployment space user interface.

## Queuing and concurrent job executions

---

The maximum number of concurrent jobs that can be run for each deployment is handled internally by the deployment service. A maximum of two jobs per batch deployment can be executed concurrently. Any deployment job requests for a specific batch deployment that already has two jobs under running state will be placed into a queue for execution at a later point of time. Once any of the running jobs are completed, the next job in the queue will be picked up for execution. There is no upper limit on the queue size.

## Retention of deployment job metadata

---

The job-related metadata will be persisted and can be accessed as long as the job and its deployment are not deleted.

## Input details by framework

---

Refer to your model type for details on what types of data are supported as input for a batch job.

- [SPSS](#)
- [AutoAI](#)
- [Scikit-Learn & XGBoost](#)
- [Keras](#)
- [Caffe](#)
- [Pytorch](#)
- [Python function](#)
- [Python Scripts](#)

## SPSS

**Type:** inline and data references

**Data Sources:** Data reference type must be `data_asset` for following assets

- Local/managed assets from the space
- Connected(remote) assets with source such as Cloud Object Storage, DB2 Warehouse, or DB2.

**File Formats:** csv,xls,sas,sav

### Notes:

- To create a local or managed asset as an output data reference, the `name` field should be specified for `output_data_reference` so that a data asset will be created with the specified name. Specifying an `href` that refers to an existing local data asset is not supported. Note that connected data assets referring to databases Db2 or Db2 Warehouse can be created in the `output_data_references` only when the `input_data_references` also refers to one of these sources.
- The `environment_variables` parameter of deployment jobs is not applicable
- If SPSS model stream has single input and single output data asset nodes, then the following data sources can be used: Local/managed assets from the deployment space, or connected(remote) assets with source such as Cloud Object Storage, Db2 Warehouse, or Db2.
- If an SPSS model stream has multiple input and output data asset nodes, only Db2 and Db2 Warehouse data sources are supported. The input data nodes can refer to multiple connections, however the output data nodes should refer to a single connection.
- Currently, the UI allows you to create a job for multiple input data references only if the model is saved with `input_data_schema`. To workaround this limitation, download the existing model from the UI flow and create a new model using the Python client or REST API by providing the model `input_data_schema`. For example, you can pass a list to the `INPUT_DATA_SCHEMA` data type as follows :

```
# The "id" name provided below is the "connection name" referred in data asset
nodes of SPSS model stream
input_schema = [ { "id": "DB2 WarehouseConn", "fields": [] },
                  { "id": "DB2Connection", "fields": [] }
                ]
meta_props = {
    wml_client.repository.ModelMetaNames.NAME: "Credit_Card_Complaints_Model",
    wml_client.repository.ModelMetaNames.RUNTIME_UID: "<any model runtime
id>",
    wml_client.repository.ModelMetaNames.INPUT_DATA_SCHEMA: [input_schema],
    wml_client.repository.ModelMetaNames.TYPE: "any model type"
}
```

Once the model is created using the Python client, you can continue to create the jobs from the UI flow by providing multiple input data references.

- If you are creating job via the Python client, you must provide the connection name referred in data nodes of SPSS model stream in the `"id"` field, and the data asset href in `"location.href"` for input/output data references of the deployment jobs payload. For example, you can construct the jobs payload like this:

```

job_payload_ref = {
  client.deployments.ScoringMetaNames.INPUT_DATA_REFERENCES: [{
    "id": "DB2Connection",
    "name": "drug_ref_input1",
    "type": "data_asset",
    "connection": {},
    "location": {
      "href": input_asset_href1
    }
  }, {
    "id": "Db2 WarehouseConn",
    "name": "drug_ref_input2",
    "type": "data_asset",
    "connection": {},
    "location": {
      "href": input_asset_href2
    }
  }],
  client.deployments.ScoringMetaNames.OUTPUT_DATA_REFERENCE: {
    "type": "data_asset",
    "connection": {},
    "location": {
      "href": output_asset_href
    }
  }
}

```

## AutoAI

**Type:** inline and data references

**Data Sources:** Data reference type must be data\_asset for following assets

- Local/managed assets from the space
- Connected(remote) assets with source such as Cloud Object Storage.

**File Formats:** csv

**Notes:** `environment_variables` parameter of deployment jobs is not applicable

## Scikit-Learn & XGBoost

**Type:** inline and data references

**Data Sources:** Data reference type must be data\_asset for following assets

- Local/managed assets from the space
- Connected(remote) assets with source such as Cloud Object Storage.

**File Formats:** csv, ZIP containing .csv files

**Notes:** `environment_variables` parameter of deployment jobs is not applicable

## Tensorflow

**Type:** inline and data references

**Data Sources:** Data reference type must be data\_asset for following assets

- Local/managed assets from the space
- Connected(remote) assets with source such as Cloud Object Storage.

**File Formats:** ZIP containing JSON files



**Notes:** `environment_variables` parameter of deployment jobs is not applicable

## Keras

**Type:** inline and data references

**Data Sources:** Data reference type must be `data_asset` for following assets

- Local/managed assets from the space
- Connected(remote) assets with source such as Cloud Object Storage.

**File Formats:** ZIP containing JSON files

**Notes:** `environment_variables` parameter of deployment jobs is not applicable

## Caffe

**Type:** inline and data references

**Data Sources:** Data reference type must be `data_asset` for following assets

- Local/managed assets from the space
- Connected(remote) assets with source such as Cloud Object Storage.

**File Formats:** ZIP containing JSON files

**Notes:** `environment_variables` parameter of deployment jobs is not applicable

## Pytorch

**Type:** inline and data references

**Data Sources:** Data reference type must be `data_asset` for following assets

- Local/managed assets from the space
- Connected(remote) assets with source such as Cloud Object Storage.

**File Formats:** ZIP containing JSON files

**Notes:** `environment_variables` parameter of deployment jobs is not applicable

## Python function

**Type:** inline

**Notes:** `environment_variables` parameter of deployment jobs is not applicable

## Python Scripts

**Type:** inline and data references

**Data Sources:** Data reference type must be `data_asset` for following assets

- Local/managed assets from the space
- Connected(remote) assets with source such as Cloud Object Storage.

**File Formats:** any

**Notes:**

- Environment variables required for executing the Python Script can be specified as key - value pairs in `scoring.environment_variables` parameters in deployment jobs payload. `Key` must be the name of

environment variable. `Value` must be value of the corresponding environment variable.

- The payload of deployment jobs payload will be saved as a JSON file in the deployment container where Python script will be executed. Python script can access the full path filename of the JSON file using `JOBS_PAYLOAD_FILE` environment variable.
- If input data is referenced as a local or managed data asset, deployment service will download the input data and place it in the deployment container where Python script will be executed. The location(path) of the downloaded input data can be accessed through `BATCH_INPUT_DIR` environment variable.
- If the input data is a connected data asset, downloading of the data must be handled by the Python script. If connected data asset reference is present in the deployment jobs payload, it can be accessed using `JOBS_PAYLOAD_FILE` which contains full path to deployment jobs payload saved as a JSON file.
- If output data must be persisted as a local or managed data asset in a space, users can specify the name of the asset to be created in `scoring.output_data_reference.location.name`. As part of Python script, output data can be placed in the path specified by environment variable `BATCH_OUTPUT_DIR`. Deployment service will compress in ZIP format and upload the data in `BATCH_OUTPUT_DIR`.
- If output data must be saved in a remote data store, users must specify the reference of the output data asset(for example, a connected data asset) in `output_data_reference.location.href`. Python script must take care of uploading the output data to the remote data source. If connected data asset reference is present in the deployment jobs payload, it can be accessed using `JOBS_PAYLOAD_FILE`, which contains full path to deployment jobs payload saved as a JSON file.
- If the Python script does not require any input or output data references to be specified in deployment job's payload, empty object as `{ }` can be specified for `input_data_references` and empty `{ }` object for `output_data_references` can be specified in deployment jobs payload.

## Managing deployment jobs (Version 2.0)

---

A job is a way of running a batch deployment or a script in Watson Machine Learning. You can choose to run a job manually or on a schedule you specify. After you create one or more jobs, you can view and manage them from the **Jobs** tab of your deployment space.

From the **Jobs** tab of your space, you can:

- See the list of the jobs in your space
- View the details of each job. You can change the schedule settings of a job and pick a different environment definition.
- Monitor job runs
- Create jobs
- Delete jobs

You can also create jobs when you create a batch deployment.

- [Creating jobs from the Jobs tab](#)
- [Scheduling a batch deployment](#)
- [Viewing jobs in a space](#)

## Creating jobs from the Jobs tab

---

To create a job:

1. From your deployment space, click the **Jobs** tab and then click **New job**.
2. Enter a job name and description.
3. Select the deployment you want to run.
4. Select the environment runtime for your job.
5. Optional: Select to schedule the job. Select the start date for this schedule. Click the calendar to select a date. Specify the start time, the time zone, and the repeat settings which depend on the frequency you selected.
6. Create the job. You can create and run the job immediately, for example if you didn't specify a schedule, or you can create the job and run it later, either manually or as specified in the schedule.

## Scheduling a batch deployment

---

When you create a batch deployment, you can optionally schedule a job for the deployment. Scheduling a deployment creates a job that will display on the Jobs page of the deployment space.

Note that if you exclude certain week days, the job might not run as you would expect. The reason is due to a discrepancy between the timezone of the user who creates the schedule, and the timezone of the master node where the job runs.

## Viewing jobs in a space

---

You can view all of the jobs that exist for your deployment space from the Jobs page. You can delete a job from this page.

To view the details of a specific job, click the job. From the job's details page, you can:

- View the runs for that job and the status of each run. If a run failed, you can select the run and view the log tail or download the entire log file to help you troubleshoot the run. A failed run might be related to a temporary connection or environment problem. Try running the job again. If the job still fails, you can send the log to Customer Support.
- Edit schedule settings or pick another environment definition.
- Run the job manually by clicking the run icon from the job action bar. You must deselect the schedule to run the job manually.

## Creating a Core ML deployment (Version 2.0)

---

In a Core ML deployment, you download a machine learning model for use with iOS apps. This deployment type is only supported on certain frameworks, such as TensorFlow.

To create a Core ML deployment:

1. From the deployment space, click the name of the saved model you want to deploy. The model detail page opens.
2. From the **Deployments** tab, click **Add new deployment**.
3. Choose **Core ML for iOS** as the deployment type.
4. Provide a name and adjust any optional settings for the deployment, then click **Create** to create the deployment.
5. When the deployment is complete, click the deployment name to view the details and to get the URL for the Core ML deployment.
6. Download the compressed file, uncompress it, and follow Core ML instructions for integrating with an iOS app.

You can review a sample notebook that shows how to use a [Core ML model to predict house price](#).

## Deploying using the Python client (Version 2.0)

---

Starting with a trained machine learning model, save the model to IBM Watson Machine Learning using the Python client, then deploy and score it.

- [Python client library reference](#)
- [Persisting a model to Watson Machine Learning](#)
- [Creating and scoring an online deployment](#)
- [Creating and scoring a batch deployment](#)

You will also learn how to list all of the deployments for given space:

- [Listing deployment jobs for given space](#)

**Note:** This topic provides excerpts from a model to demonstrate techniques, but it is not intended as a working tutorial. You can view and download complete notebook samples from this [repository](#).

## Prerequisites

---

To deploy a model to IBM Watson Machine Learning, you need:

- A trained model. Note that the model name cannot contain characters such as [ ] { } | \ " % ~ # < > that conflict with forming a valid HTTP request.
- The [credentials](#) to connect to the server.

## Watson Machine Learning Python client library reference

---

You can access a reference to all of the Python commands for Watson Machine Learning here: [Watson Machine Learning Python client library](#)

Before you can deploy, you must save the model programmatically to a deployment space on Watson Machine Learning.

**Note:** After you save a model to a deployment space, you can view it from the space and create a deployment in the user interface. For details, see [Creating a deployment](#).

This topic steps you through the process of saving, then deploying, a sample model.

## Save a model to the repository

---

1. Install the watson-machine-learning-client package to store your model in the Watson Machine Learning repository. For example:

**Important:** On Windows, if you have a "Per-machine (all users)" installation, you must use "Run as Administrator" to install libraries or packages in your notebook.

```
!pip install --upgrade watson-machine-learning-client-V4
```

2. Next, authenticate with the server. This sample shows authentication from a notebook. Note that fictional credentials are used in this example:

```
wml_credentials = {
    "instance_id": "wml_local",
    "url" : "https://<WML Server Hostname>:31843",
    "username": "<USER NAME>",
    "password": "<PASSWORD>",
    "version": "2.0"
}
```

3. Initialize the client with the credentials:

```
from ibm_watson_machine_learning import APIClient
wml_client = APIClient(wml_credentials)
```

4. (Optional) Create a new deployment space. To use an existing deployment space, skip this step and enter the name of the space in the next step, entering the credentials for your [Cloud Object Storage](#).

```
metadata = {
    client.spaces.ConfigurationMetaNames.NAME: 'YOUR DEPLOYMENT SPACE NAME',
    client.spaces.ConfigurationMetaNames.DESRIPTION: 'description',
    client.spaces.ConfigurationMetaNames.STORAGE: {
        "type": "bmc_os_object_storage",
        "resource_crn": 'PROVIDE COS RESOURCE CRN '
    },
    client.spaces.ConfigurationMetaNames.COMPUTE: {
        "name": 'INSTANCE NAME',
```

```

        "crn": 'PROVIDE THE INSTANCE CRN'
    }
}

space_details = client.spaces.store(meta_props=metadata)

```

5. Get the ID for the deployment space:

```

def guid_from_space_name(client, space_name):
    instance_details = client.service_instance.get_details()
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']['name'] ==
space_name) ['metadata'] ['guid'])

```

6. Enter the details for the deployment space, putting the name of your deployment space in place of "YOUR DEPLOYMENT SPACE".

```

space_uid = guid_from_space_name(client, 'YOUR DEPLOYMENT SPACE')
print("Space UID = " + space_uid)

```

**Out: Space UID = b8eb6ec0-dcc7-425c-8280-30a1d7a9c58a**

7. Set the default deployment space to work.

```

client.set.default_space(space_uid)

```

## Get the software specification

1. To view the list of predefined specifications:

```

client.software_specifications.list()

```

2. Find the id of the software specification environment that the function will be using :

```

software_spec_id = client.software_specifications.get_id_by_name('spss-
modeler_18.1')
print(software_spec_id)

```

## Store the model

1. Store the trained model to the repository and get the model ID. To do so, enter the absolute path of the trained model file, as well as the model name, model type and model runtime. Note that the model name cannot contain characters such as [ ] { } | \ " % ~ # < > that conflict with forming a valid HTTP request.

```

model_details = client.repository.store_model(model="<Trained Model
file>",meta_props={
    client.repository.ModelMetaNames.NAME:"<Model Name>",
    client.repository.ModelMetaNames.TYPE:"<model type>",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_id }
)
model_id = client.repository.get_model_uid(model_details)

```

For example, a trained SPSS model might have metadata like this:

```

model_details =
client.repository.store_model(model="example.com/my_spss_model",meta_props={
    client.repository.ModelMetaNames.NAME:"my_spss_model",
    client.repository.ModelMetaNames.TYPE:"spss-modeler_18.1",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_id }
)
model_id = client.repository.get_model_id(model_details)

```

2. Print the model ID:

```

print(model_id)

```

## Create an online deployment

---

Follow these steps to create an online deployment where you can get realtime scores for your model.

1. To select the hardware runtime environment to deploy the function, first view available hardware configurations:

```
client.hardware_specifications.list()
```

2. Select a hardware configuration:

```
hardware_spec_id = client.hardware_specifications.get_id_by_name('NAME OF THE  
HARDWARE SPECIFICATION')
```

for example :

```
#hard_ware_spec_id = client.hardware_specifications.get_id_by_name('M')}
```

3. Create and name the online deployment for the model you persisted.

```
dep_details = client.deployments.create(artifact_uid=model_id,meta_props={  
client.deployments.ConfigurationMetaNames.NAME:"<ONLINE_DEPLOYMENT_NAME>",  
client.deployments.ConfigurationMetaNames.HARDWARE_SPEC: { "id":  
hardware_spec_id},  
client.deployments.ConfigurationMetaNames.ONLINE:{}})
```

**Out: Successfully finished deployment creation, deployment\_uid='a095a83-3c91-4d10-8a5c-f967a7702902'**

4. Get the deployment ID.

```
dep_id = client.deployments.get_uid(dep_details)
```

5. To score the deployed model construct the model payload, following the schema of the model.

```
fields = ["Age", "Sex", "BP", "Cholesterol", "Na", "K"]  
values = [[23, "F", "HIGH", "HIGH", 0.792535, 0.031258]]  
  
scoring_payload = {  
client.deployments.ScoringMetaNames.INPUT_DATA: [{  
    "fields": fields,  
    "values": values  
}]  
}
```

6. Run the score function to generate the prediction.

```
client.deployments.score(deployment_id=dep_id,meta_props=scoring_payload)
```

```
Out[13]: {'predictions': [{'fields': ['Age',  
    'Sex',  
    'BP',  
    'Cholesterol',  
    'Na',  
    'K',  
    '$N-Drug',  
    '$NC-Drug'],  
    'values': [[23,  
    'F',  
    'HIGH',  
    'HIGH',  
    0.792535,  
    0.031258,  
    'drugY',  
    0.9999929901781122]]}]}
```

7. Finally, delete the deployment.

```
client.deployments.delete(dep_id)
```

## Create a batch deployment

---

Create a batch deployment to submit a data asset containing multiple payloads, and write the resulting predictions to an output file.

1. Create and name the batch deployment.

```
dep_details = client.deployments.create(artifact_uid=model_id,meta_props={
    client.deployments.ConfigurationMetaNames.NAME:"<BATCH_DEPLOYMENT_NAME>",
    client.deployments.ConfigurationMetaNames.HARDWARE_SPEC: { "id": hardware_spec_id},
    client.deployments.ConfigurationMetaNames.BATCH:{},
    client.deployments.ConfigurationMetaNames.COMPUTE:{"name":"S","nodes":1}
})
```

**Out: Successfully finished deployment creation, deployment\_uid="e6936222-4efb-4f03-997e-e1cf0ce29991"**

2. Get the deployment ID.

```
dep_id = client.deployments.get_uid(dep_details)
print(dep_id)
```

3. Create the data asset for the batch deployment, entering the input data asset file name and input data asset name. Make sure you use a supported data asset file formats for the runtime type.

```
asset_details = client.data_assets.create(name="
<INPUT_DATA_ASSET_NAME>",file_path="<Input Data Asset File>")
```

4. Get the asset ID.

```
asset_id = client.data_assets.get_uid(asset_details)
```

5. Get the Href for the input data asset.

```
asset_href = client.data_assets.get_href(asset_details)
```

6. Print the asset ID and asset Href.

```
print(asset_id)
print(asset_href)
```

**Out: 8c11502f-e047-4822-bc6e-7d6c93a24336 /v2/assets/8c11502f-e047-4822-bc6e-7d6c93a24336?space\_id=62748c9a-5353-46e7-8fa2-c3ec971eccf2**

## Create and run the batch deployment job

1. Construct the deployment job payload. Enter the input data asset Href and name, and the output data asset name and description.

```
job_payload_ref = {
    client.deployments.ScoringMetaNames.INPUT_DATA_REFERENCES: [{
        "id": "input_data",
        "name": "<INPUT_DATA_ASSET_NAME>",
        "type": "data_asset",
        "connection": {},
        "location": {
            "href": "<INPUT_DATA_ASSET_HREF>"
        },
    }],
    client.deployments.ScoringMetaNames.OUTPUT_DATA_REFERENCE: {
        "type": "data_asset",
```

```

        "connection": {},
        "location": {
            "name": "<OUTPUT_DATA_ASSET_NAME>_{}.csv".format(dep_id),
            "description": "<DESCRIPTION>"
        }
    }
}

```

2. Enter the batch job and supply the deployment ID.

```

job =
client.deployments.create_job(deployment_id=dep_id,meta_props=job_payload_ref)

```

3. Get the job ID.

```

job_id = client.deployments.get_job_uid(job)

```

4. Execute the job until the state of Get Job Status changes to "completed", "failed" or "cancelled"

```

client.deployments.get_job_status(job_id)

```

**Out: {'state': 'completed', 'running\_at': '2019-12-06T13:26:03.699Z', 'completed\_at': '2019-12-06T13:26:18.515Z'}**

5. Get the job details by ID.

```

client.deployments.get_job_details(job_id)

```

Results in:

```

Out[32]: {'metadata': {'guid': '1e7113d2-5b5b-47d4-9420-b7a3a1f7e946',
  'href': '/v4/deployment_jobs/1e7113d2-5b5b-47d4-9420-b7a3a1f7e946',
  'created_at': '2019-12-06T13:25:49.927Z',
  'parent': {'href': ''}},
  'entity': {'deployment': {'href': '/v4/deployments/e6936222-4efb-4f03-997e-e1cf0ce29991'},
  'scoring': {'input_data_references': [{'name': 'drug_ref_input',
    'location': {'href': '/v2/assets/8c11502f-e047-4822-bc6e-7d6c93a24336?space_id=62748c9a-5353-46e7-8fa2-c3ec971eccf2'},
    'id': 'input_data',
    'schema': {},
    'connection': {},
    'type': 'data_asset'}]},
  'output_data_reference': {'type': 'data_asset',
  'connection': {},
  'location': {'name': 'drug_predictions_e6936222-4efb-4f03-997e-e1cf0ce29991.csv',
  'description': 'testing csv file',
  'href': '/v2/assets/7249af68-626c-48ce-9b9c-2a7d4b9e0270?space_id=62748c9a-5353-46e7-8fa2-c3ec971eccf2'}},
  'status': {'state': 'completed',
  'running_at': '2019-12-06T13:26:03.699Z',
  'completed_at': '2019-12-06T13:26:18.515Z'}}}

```

6. Download the prediction results to the output file.

```

client.data_assets.download(<OUTPUT_DATA_ASSET_ID>, <PREDICTION_RESULT_FILENAME>)

```

7. Finally, clean up the deployment by deleting the batch deployment, the saved model, and the input data asset.

```

client.deployments.delete(dep_id)
client.repository.delete(model_id)
client.data_assets.delete(asset_id)

```

## Listing deployment jobs for given space

To list all of the deployment jobs currently available for a site, use the `list_jobs` method with a specified space id:

```

client.deployment.list_jobs(<deployment_id>)

```

Note that only deployment jobs are listed. Deleting a deployment job means it will no longer be listed for the space.

## Deploying Python functions (Version 2.0)



This topic describes how to deploy Python functions.

You can deploy Python functions in Watson Machine Learning the same way that you can deploy models. Your tools and apps can use the Watson Machine Learning Python client or REST API to send data to your deployed functions the same way that they send data to deployed models. Deploying functions gives you the ability to hide details (such as credentials), preprocess data before passing it to models, perform error handling, and include calls to multiple models, all within the deployed function instead of in your application.

**Note:** This topic provides excerpts from a model to demonstrate techniques, but it is not intended as a working tutorial. You can view and download samples from this [repository](#).

## Deploying a function from a deployment space

---

This topic describes how to deploy a function using the Python client, but you can also deploy a function from a deployment space via the user interface. For details on creating and deploying from a deployment space, see [Deployment spaces](#).

### Watson Machine Learning Python client library reference

[Watson Machine Learning Python client library](#) 

There are six basic steps for creating and deploying functions in Watson Machine Learning:

1. [Define the function](#)
2. [Authenticate and define a space](#)
3. [Store the function in the repository](#)
4. [Get the software specification](#)
5. [Deploy the stored function](#)
6. [Send data to the function for processing](#)

### Step 1: Define the function

To define a function, create a *Python closure* with a nested function named 'score'.

#### Example Python code

```
#wml_python_function
def my_deployable_function():

    def score( payload ):

        message_from_input_payload = payload.get("input_data")[0].get("values")[0][0]
        response_message = "Recieved message - {0}".format(message_from_input_payload)

        # Score using the pre-defined model
        score_response = {
            'predictions': [{ 'fields': ['Response_message_field'],
                               'values': [[response_message]]
                             }]
        }
        return score_response

    return score
```

You could test your function like this:

```
input_data = { "input_data": [{ "fields": [ "message" ],
                                "values": [[ "Hello world!" ]]
                              }
                ]
}
```

```

    }
function_result = my_deployable_function()( input_data )
print( function_result )

```

It will return the message 'Hello world!'.

## Python closures

To learn more about closures, see:

- [Python Closures](#)
- [Closures](#)
- [Nested function, Scope of variable & closures in Python](#)

## Requirements for the nested, 'score' function

The following are requirements and usage notes for the nested function for online deployments:

- `score()` must accept a single, JSON input parameter
- The scoring input payload will be passed as value for input parameter for `score()`. Therefore, the value of `score()` input parameter must be handled accordingly inside the `score()`.
- The scoring input payload must match the input requirements for the concerned Python Function.
- Additionally, the scoring input payload must include an array with the name `values` as shown in this example schema. Note that the `input_data` parameter is mandatory in the payload.

```

{ "input_data": [{
                                "values": [[ "Hello world!" ]]
                                }
]}

```

- The output payload expected as output of `score()` must include the schema of the "score\_response" variable described in the "Step 1: Define the function" section for status code 200. Note that the `prediction` parameter, which has an array of JSON objects as its value, is mandatory in the `score()` output.
- The `score` function must return a JSON-serializable object (for example: dictionaries or lists).
- When a Python function is saved using the Python client where a reference to the outer function is specified, only the code in the scope of the outer function (including its nested functions) are saved. Therefore, the code outside the outer function's scope will not be saved and thus will not be available when you deploy the function.

This topic assumes you have a Python function. For a sample of how to create one, download a [notebook demonstrating how to scrape a Web page](#) from the collection of sample notebooks.

## Step 2: Authenticate with the Python client

1. Add a notebook to your project by clicking **Add to project** and selecting **Notebook**.
2. Authenticate with the Python client, following the client instructions for authentication.
3. Initialize the client with the credentials:

```

from ibm_watson_machine_learning import APIClient
wml_client = APIClient(wml_credentials)

```

4. (Optional) Create a new deployment space. To use an existing deployment space, skip this step and enter the name of the space in the next step, entering the credentials for your [Cloud Object Storage](#).

```

metadata = {
    client.spaces.ConfigurationMetaNames.NAME: 'YOUR DEPLOYMENT SPACE NAME',

```

```

client.spaces.ConfigurationMetaNames.DESRIPTION:  description',
client.spaces.ConfigurationMetaNames.STORAGE: {
    "type": "bmcos_object_storage",
    "resource_crn": 'PROVIDE COS RESOURCE CRN '
},
client.spaces.ConfigurationMetaNames.COMPUTE: {
    "name": 'INSTANCE NAME,
    "crn": 'PROVIDE THE INSTANCE CRN'
}
}

```

```
space_details = client.spaces.store(meta_props=metadata)
```

#### 5. Get the ID for the deployment space:

```

def guid_from_space_name(client, space_name):
    instance_details = client.service_instance.get_details()
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']['name'] ==
space_name) ['metadata'] ['guid'])

```

#### 6. Enter the details for the deployment space, putting the name of your deployment space in place of 'YOUR DEPLOYMENT SPACE'.

```

space_uid = guid_from_space_name(client, 'YOUR DEPLOYMENT SPACE')
print("Space UID = " + space_uid)

```

**Out: Space UID = b8eb6ec0-dcc7-425c-8280-30a1d7a9c58a**

#### 7. Set the default deployment space to work.

```
client.set.default_space(space_uid)
```

### Step 4: Get the software specification

Your function requires a software specification to run.

#### 1. To view the list of predefined specifications:

```
client.software_specifications.list()
```

#### 2. Find the id of the software specification environment that the function will be using :

```

software_spec_id = client.software_specifications.get_id_by_name('ai-function_0.1-
py3.6')
print(software_spec_id)

```

### Step 4: Save the function

#### 1. Create the function metadata.

```

function_meta_props = {
    client.repository.FunctionMetaNames.NAME: 'sample_function_with_sw',
    client.repository.FunctionMetaNames.SOFTWARE_SPEC_ID: software_spec_id
}

```

#### 2. Extract the function UID from the details.

```

function_artifact =
client.repository.store_function(meta_props=function_meta_props,
function=my_deployable_function)
function_uid = client.repository.get_function_id(function_artifact)
print("Function UID = " + function_id)

```

**Function UID = 0f263463-21ec-4d2f-a277-2a7525f64b4e**

3. Get the saved function metadata from Watson Machine Learning using the function UID.

```
function_details = client.repository.get_details(function_uid)
from pprint import pprint
pprint(function_details)
```

4. To confirm the function was saved, list all of the stored functions using the `list_functions` method.

```
client.repository.list_functions()
```

## Step 5: Deploy the stored function

1. To select the hardware runtime environment to deploy the function, first view available hardware configurations:

```
client.hardware_specifications.list()
```

2. Select a hardware configuration:

```
hardware_spec_id = client.hardware_specifications.get_id_by_name('NAME OF THE
HARDWARE SPECIFICATION')
```

for example :

```
#hard_ware_spec_id = client.hardware_specifications.get_id_by_name('M')}
```

3. Deploy the Python function to the deployment space by creating deployment metadata and using the function UID obtained in the previous section.

```
deploy_meta = {
    client.deployments.ConfigurationMetaNames.NAME: "Web scraping python function
deployment",
    client.deployments.ConfigurationMetaNames.ONLINE: {},
    client.deployments.ConfigurationMetaNames.HARDWARE_SPEC: { "id":
hardware_spec_id}
}
```

4. Create the deployment.

```
deployment_details = client.deployments.create(function_uid,
meta_props=deploy_meta)
```

```
#####
Synchronous deployment creation for uid: '0f263463-21ec-4d2f-a277-2a7525f64b4e' started
#####
```

```
initializing..
ready
```

```
-----
Successfully finished deployment creation, deployment_uid='f451cc11-e1a4-4b74-a361-cca0ab457298'
-----
```

5. View the deployment details.

```
deployment_details
```

```
{'metadata': {'parent': {'href': ''},
  'guid': 'f451cc11-e1a4-4b74-a361-cca0ab457298',
  'modified_at': '',
  'created_at': '2020-01-14T00:06:22.171Z',
  'href': '/v4/deployments/f451cc11-e1a4-4b74-a361-cca0ab457298'},
  'entity': {'name': 'Web scraping python function deployment',
  'custom': {},
  'online': {},
  'description': '',
  'space': {'href': '/v4/spaces/b8eb6ec0-dcc7-425c-8280-30a1d7a9c58a'},
  'status': {'state': 'ready',
  'online_url': {'url': 'https://wmlserver-svt.ml.test.cloud.ibm.com:31843/v4/deployments/f451cc11-e1a4-4b74-a361-cca0ab457298/predictions'}},
  'asset': {'href': '/v4/functions/0f263463-21ec-4d2f-a277-2a7525f64b4e?space_id=b8eb6ec0-dcc7-425c-8280-30a1d7a9c58a'},
  'auto_redeploy': False}}
```

6. To confirm that the deployment was created successfully, list all deployments.

```
client.deployments.list()
```

GUID	NAME	STATE	CREATED	ARTIFACT_TYPE
f451cc11-e1a4-4b74-a361-cca0ab457298	Web scraping python function deployment	ready	2020-01-14T00:06:22.171Z	function

## Step 6: Send data to the function for processing

Follow these steps to score the function and return a prediction.

1. List the function you plan to score.

```
client.repository.list_functions()
```

2. Prepare the scoring payload, matching the schema of the function.

```
job_payload = {
  client.deployments.ScoringMetaNames.INPUT_DATA: [{
#    "input_data": [{
      'fields': ['url'],
      'values': [
        'https://www.ibm.com/cloud/machine-learning'
      ]
    }
  ]
}
pprint(job_payload)
```

```
{'input_data': [{'fields': ['url'], 'values': ['https://www.ibm.com/
cloud/machine-learning']}]}
```

3. Generate the prediction and display the results:

```
job_details = client.deployments.score(deployment_uid, job_payload)
pprint(job_details['predictions'][0]['values'][0][:10])
```

```
['02', '2018', '2019', '459', '49', '575', 'about', 'accelerate', 'accelerates', 'accelerator']
```

## Deploying a script using the Python client

You can create and deploy a script using the Python client for Watson Machine Learning. Access a reference to all of the Python commands for Watson Machine Learning here: [Watson Machine Learning Python client library](#)

For additional details on supported input and output types and setting environment variables, see [Batch deployment details](#).

### Create the script asset

After initializing the Watson Machine Learning client and creating and setting your default space, view available meta names for scripts.

```
print('Available script configuration:', wml_client.script.ConfigurationMetaNames.get())
print('Available deployments configuration:',
wml_client.deployments.ConfigurationMetaNames.get())
wml_client.software_specifications.list()
```

## Specify the software for running the script

---

Specify the software and version you will use to run the script. Then, create the metadata for the script and store the script in the repository.

**Note:** Software specifications for a script must be referenced by ID, not name.

```
softwareSpecId = wml_client.software_specifications.get_uid_by_name('ai-function_0.1-
py3.6')

# create the metadata for the script
meta_props = {
    wml_client.script.ConfigurationMetaNames.NAME: 'My_Script',
    wml_client.script.ConfigurationMetaNames.DESRIPTION: 'My_Script description', #
optional
    wml_client.script.ConfigurationMetaNames.SOFTWARE_SPEC_UID: softwareSpecId
}
# create the asset for the script
script_details = wml_client.script.store(meta_props, '../target/zips/script.py.zip')
print('Script:', script_details)
script_id = wml_client.script.get_uid(script_details)
print('Script:', wml_client.script.get_details(script_id))
```

## Notes about specifying the software

How you specify the software for executing a script depends on whether it is a default or customized environment.

- If you have created the Python script in Watson Studio JupyterLab with Default Python 3.6 environment, save the model using the software specification name `default_py3.6` or `ai-function_0.1-py3.6`. For example: `softwareSpecId = wml_client.software_specifications.get_uid_by_name('default_py3.6')`
- You can create a custom environment in Watson Studio if you want to install 3rd party libraries required to execute your Python scripts. You can use a custom software specification with `ai-function_0.1-py3.6` and `default_py3.6` software specifications as base with Python scripts.
- If you create your Python script in Watson Studio JupyterLab with a customized environment, reference the custom software specification by name when you save the model. The name of the custom software specification is the same as the name of the custom environment. For example, if you named your custom environment `my_cust_model1_env`: `softwareSpecId = wml_client.software_specifications.get_uid_by_name('my_cust_model1_env')`

## Create the deployment

---

This sample defines a batch deployment job for a script and creates a data asset (a .csv file) as the input data source for the deployment.

```
import pprint

pp = pprint.PrettyPrinter(indent=2)

deployment_meta_props = {
    wml_client.deployments.ConfigurationMetaNames.NAME: 'My_Script_deployment',
    wml_client.deployments.ConfigurationMetaNames.DESRIPTION: 'My_Script_deployment
description',
    wml_client.deployments.ConfigurationMetaNames.BATCH: {},
    wml_client.deployments.ConfigurationMetaNames.HARDWARE_SPEC: {'name': 'S'}
}
```

```
deployment = wml_client.deployments.create(script_id, deployment_meta_props)
deployment_id = wml_client.deployments.get_uid(deployment)
print('Created batch script deployment\n', pp.pformat(deployment))
```

## Notes about deploying a script

- Only batch deployment is supported for python scripts. For additional information about creating and executing a batch deployment job for python scripts, refer to the "Python Scripts" section the [Batch details](#) topic.
- If you want to pass any variable values to python scripts specific to a job, you can use the `environment_variables` parameter in the jobs creation payload to specify the requirements as key-value pairs. The keys will be environment variable name and corresponding value specified in the payload will be value for that environment variable. These are some pre-defined environment variables:

Variable	Description
JOB_PAYLOAD_FILE	Use this variable to specify the full path of the JSON payload file for the deployment when creating a batch deployment job. Information related to input data, input and output data references that was specified in Jobs payload can be accessed from this file.
WORKDIR	Contains the full path to the python script's .py file.
BATCH_INPUT_DIR1	When the input data reference specified in the jobs payload is a managed or reference data asset, the deployment service will download the corresponding data to the directory path specified in the <code>BATCH_INPUT_DIR1</code> environment variable.
BATCH_OUTPUT_DIR	When the batch job's output data has to be saved as a new data asset, users can place their batch output files in directory specified in <code>BATCH_OUTPUT_DIR</code> . The files in <code>BATCH_OUTPUT_DIR</code> will be compressed using ZIP utility and uploaded to the Catalog Asset Management Service. The name of the asset will be the value of <code>location.name</code> parameter payload. If <code>output_data_reference.location.name</code> is not present in payload but <code>output_data_references</code> is present in the payload with value as <code>{"type" : "data_asset"}</code> , the name of the newly created data asset will be <code>py-script asset</code> .

- Although you can install the required packages from within Python script using the `subprocess.check_output()` API call, the best practice is to use package extensions and custom software specifications for this purpose.
- You can use the directory specified environment variable `WORKDIR` or `/tmp` for any intermediate files-related operations.

## Specify the input data

Next, create the data asset that will be the input data for the job. The data can reside in a file, or be a reference to data stored in a database or in a storage system, such as a Cloud Object Storage bucket. For details on specifying input

data, see [Data Reference](#)

```
# create a data asset which will be an input to the batch scoring job of the script
input_data_asset = wml_client.data_assets.create('input_file',
file_path='resources/script_input_data.csv')
input_data_asset_href = wml_client.data_assets.get_href(input_data_asset)
```

## Define the job

---

When you specify the job to run the script, you specify the location of the input data and the name and location for the output from the job.

```
job_payload_ref = {
wml_client.deployments.ScoringMetaNames.INPUT_DATA_REFERENCES: [{
    'name': 'test_ref_input',
    'type': 'data_asset',
    'connection': {},
    'location': {
        'href': input_data_asset_href
    }
}],
wml_client.deployments.ScoringMetaNames.OUTPUT_DATA_REFERENCE: {
    'type': 'data_asset',
    'connection': {},
    'location': {
        'name': 'scripts_result_{}.csv'.format(deployment_id),
        'description': 'testing zip results'
    }
}
}
job = wml_client.deployments.create_job(deployment_id, meta_props=job_payload_ref)
job_id = wml_client.deployments.get_job_uid(job)
```

## Run the job

---

When the job runs you can poll for the status and print the result. The job is complete when the status changes to *completed* or *failed*.

```
# wait for the job to complete
status = wml_client.deployments.get_job_status(job_id)
loop = 0
while status['state'] != 'completed':
    if status['state'] == 'failed':
        break
    print('Job status', status['state'])
    loop += 1
    if loop > 30:
        break
    time.sleep(1)
    status = wml_client.deployments.get_job_status(job_id)

if status['state'] == 'completed':
    print('Job completed', pp.pformat(status))
else:
    print('Job failed', pp.pformat(status))
```

## Delete the deployment

---

Remove the deployment and delete the script asset when done.

```
# delete the deployment
if deployment_id != 'undefined':
    wml_client.deployments.delete(deployment_id)
# delete the script
if script_id != 'undefined':
```



```
wml_client.script.delete(script_id)
# delete the space
wml_client.spaces.delete(space_uid)
```

## Deploying models to Watson Machine Learning Server

After you create and train models in a notebook or using SPSS Modeler, you can deploy to Watson Machine Learning Server if you are set up to connect to the server.

To deploy an asset, you must have a *deployment space* associated with the project that contains the asset. A deployment space is where you will save the model, create the deployment, and find the information you need to score the model and get a prediction back, or embed the deployment in an app so you can interact with it programmatically.



Deployment is part of the lifecycle of a model. To build, train, and deploy a model from Watson Studio Desktop, follow these steps:

1. **Connect to Watson Machine Learning Server.** Work with your system administrator to determine whether you can connect to Watson Machine Learning Server and to get the credentials you will need to authenticate with the server.
2. **Create a deployment space.** Create a deployment space that is associated with your project.
3. **Promote the model.** When you feel the model is ready to test, promote a saved model from the project to the associated deployment space.
4. **Deploy the model.** When you deploy the model, you are making it available to process new data and return confidence scores about how well the new data aligns with the rules the model established during processing. For example, if you are trying to determine whether a customer with a particular profile is likely to buy a red or a blue car, you submit all of the profile data for the customer and get back a score, or prediction, of their likely purchase based on all of the related data used to train the model.

## Deployment types

You can create the following deployment types on Watson Machine Learning Server:

- **Online:** Submit payload data for processing and receive a score, or prediction, back in real-time. This is the only type of deployment supported in the deployment spaces interface.
- **Batch:** Perform bulk analysis on a batch of data.
- **Virtual:** Download a model for use in a machine learning application. This deployment type is only supported on certain frameworks.

You can view and download notebook samples of Batch and Virtual deployments from this repository:  
<https://github.com/IBMDaScience/sample-notebooks/tree/master/Desktop/notebooks>

## Next steps

---

Find out more about how to:

- [Connect to Watson Machine Learning Server](#)
- [View and manage assets on deployment spaces.](#)
- [Deploy a model from a deployment space.](#)
- [Deploy a model from a notebook](#) using the Python client.

## Deployment spaces (Version 1.1)

---

You can use deployment spaces to deploy models and manage your deployments.

Deployment spaces allow you to create deployments for saved models and view and manage all of the activity and assets for the deployments. You can:

- [View deployed assets](#)
- [Create a deployment space](#)
- [Promote assets to a space](#)
- [Create deployments](#)
- [View deployment details](#)

### Viewing deployed assets

---

You configure and manage the deployment of a set of related assets in a space. A space contains an overview of deployment status, the deployable assets, deployments, associated input and output data, and the associated environments.

- To view all deployment spaces that you can access, click **Deployment Spaces** on the Watson Studio navigation menu.

### Creating a deployment space

---

A deployment space can only be associated with one project. If you have not already associated a deployment space with a project, when you attempt to promote an asset from a project, you are prompted to create a new space or choose an existing space to be associated with your project.

You can create a space from the **Overview** or **Settings** page of a project, or by following these steps:

### Promoting assets to a deployment space

---

Promote assets to a deployment space in the following ways:

- From the project **Assets** page, choose **Promote** from the action item for the asset to promote it to a deployment space.
- Using the Python client, save an asset to a project, then promote it to an associated deployment space.
- If you are creating a batch deployment, you can either promote the data assets you need for the deployment from the project or you can upload the data asset from the space.

### Creating a new deployment

---

When you promote a model to a space, components required for a successful deployment, such as a training library, model definition, or pipeline definition are automatically promoted as well. After promoting a model and data assets to a deployment space, you can create a deployment in the space.

1. Click the name of the saved model in the deployment space.
2. Click the Deployments tab.
3. Click **New Deployment** to create a deployment.
4. Choose the deployment type and fill out the specifics for the deployment. For details, see [deploying from a space](#).

## Viewing deployment details

---

On the Assets page, you can view the assets in the deployment space. You can see how many deployments each asset has as well as configuration details.

- Click an asset name to view details. The **Overview** page shows details such as the runtime environment and the training date.
- Click **Deployments** to view details for the deployments such as status and deployment ID.
- Click **Schema** to view the structure of your model.

## Deploying models (Version 1.1)

---

Use the tools available from a deployment space to deploy and run models and scripts. The types of deployment available for a model and the type of input supported depends on the model framework.

The types of deployments are:

- Online - to create a REST API endpoint for generating predictions or output in real time. You can also use the endpoint to make the model available for wider use in applications.
- Batch - to process batches of input submitted from a data file, a connection to a data repository such as a database, or connected data in a storage repository such as a Cloud Object Storage bucket. Batch deployments can be run on demand or on a schedule.
- Core ML - to download a deployment in a format for use build iOS apps.

## Creating an online deployment

---

Create an online deployment to create a REST API endpoint. You can submit new data to the endpoint and get back a realtime result or use the endpoint in an application.

For example, if you create a classification model to test whether a new customer is likely to participate in a sales promotion, you can create an online deployment for the model, and enter the new customer data to get an immediate prediction.

## Create the deployment

---

1. From the deployment space, click the name of the saved model you want to deploy. The model detail page opens.
2. From the **Deployments** tab, click **Add new deployment**.
3. Choose **Online** as the deployment type.
4. Provide a name and optional description for the deployment, then click **Create** to create the deployment.
5. When the deployment status shows **Deployed**, click the name of the deployment to view the deployment details.

## Working with an online deployment

---

There are two tabs for an online deployment:

- **API Reference** displays the API endpoint you will need to access the deployment programmatically. You will need the endpoint to use the deployment in an application. There are also code snippets in a variety of programming languages that illustrate how to access the deployment.

- **Test** provides a place where you can enter data and get a prediction back from the deployed model. If your model has a defined schema, you will see a form where you can submit data that matches the schema and get a score, or prediction, back. Otherwise, you can submit input data in JSON format and get a score back.

## Sample deployment code

When you submit JSON code as the payload, or input data, for a deployment, your input data must match the schema of the model. The "fields" must match the column headers for the data, and the "values" must contain the data, in the same order. Use this format:

```
{
  "input_data": [
    {
      "fields": [<field1>, <field2>, ...],
      "values": [<value1>, <value2>, ...]
    }
  ]
}
```

For example:

```
{
  "input_data": [
    {
      "fields": [
        "PassengerId", "Pclass", "Name", "Sex", "Age", "SibSp", "Parch", "Ticket", "Fare", "Cabin", "Embarked"
      ],
      "values": [
        [1, 3, "Braund, Mr. Owen Harris", 0, 22, 1, 0, "A/5 21171", 7.25, null, "S"]
      ]
    }
  ]
}
```

### Notes:

- Note that all strings are enclosed in double-quotes. The Python notation for dictionaries looks very similar, but Python strings in single-quotes are not accepted in the JSON data.
- Missing values can be indicated with `null`.

## Creating a batch deployment

A batch deployment processes input data from a file, data connection, or connected data in a storage bucket and writes the output to a file.

## Before you begin

1. Save a model to a deployment space.
2. Promote or add the input file for the batch deployment to the space. For details on promoting an asset to a space, see [Deployment spaces](#).

## Structuring the input data

How you structure the input data, also known as the payload, for the batch job depends on the framework for the asset you are deploying.

Framework	Input format	Notes
Scikit-learn XGBoost	.csv	The first line in the .csv file should contain column names separated by commas
TensorFlow Keras Caffe Pytorch with ONNX	.zip	Zip file should contain .json files with input data that matches the model schema
SPSS	.csv	Can also accept a zip file containing multiple .csv files

**Note:** A .csv input file or other structured data formats should be formatted to match the schema of the asset. List the column names (fields) in the first row and values to be scored in subsequent rows. For example:

```
PassengerId, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked
1,3,"Braund, Mr. Owen Harris",0,22,1,0,A/5 21171,7.25,,S
4,1,"Winslet, Mr. Leo Brown",1,65,1,0,B/5 200763,7.50,,S
```

A JSON input file should provide the same information on fields and values, using this format:

```
{"input_data":[{"fields": [<field1>, <field2>, ...],
"values": [[<value1>, <value2>, ...]]
}]}
```

For example:

```
{"input_data":[{"fields":
["PassengerId", "Pclass", "Name", "Sex", "Age", "SibSp", "Parch", "Ticket", "Fare", "Cabin", "Embarked"],
"values": [[1,3,"Braund, Mr. Owen Harris",0,22,1,0,"A/5 21171",7.25,null,"S"],
[4,1,"Winslet, Mr. Leo Brown",1,65,1,0,"B/5 200763",7.50,null,"S"]]
}]}
```

## Creating a batch deployment job

1. From the deployment space, click the name of the saved model you want to deploy. The model detail page opens.
2. Click **Create deployment**.
3. Choose **Batch** as the deployment type and enter a name for your deployment.
4. Choose a hardware definition based on the CPU and RAM that should be allocated for this deployment.
5. Click **Create** to create the deployment.
6. When the status changes to *Deployed*, click the deployment name, then click **Run** to configure how to run the deployment.
7. Select the input data source you promoted to the space and provide a name for the output file that will contain the results. If you did not promote a data asset to the space, choose **Browse for files** to upload a data asset file directly to the space for use with a batch deployment.
8. Click **Create** to create the run. Results of the run are written to the specified output file and saved as a project asset.

## Creating a virtual deployment

---

In a virtual deployment, you download a model for use in a iOS app. This deployment type is only supported on certain frameworks.

To create a virtual deployment:

1. From the deployment space, click the name of the saved model you want to deploy. The model detail page opens.
2. From the Deployments tab, click **Add new deployment**.
3. Choose **Virtual** as the deployment type.
4. Provide a name and adjust any optional settings for the deployment, then click **Create** to create the deployment.
5. When the deployment is complete, click the deployment name to view the details and to get the URL for the virtual deployment.

## Saving a model to a project using Python

---

When you are working with a model using the Python client, you have two options:

- Save the model programmatically to a Watson Studio Desktop project. You will be able to view the model as an asset in the project. From the project, you can use the user interface to promote the model and related data

assets to a deployment space, then deploy the model. This functionality is available from Watson Studio Desktop, even if you are not connected to Watson Machine Learning.

- Save the model and related assets programmatically to a deployment space. You can create the deployment programmatically or you can create it from the deployment space, but you will not be able to view the model or asset from the Watson Studio project unless you explicitly save the assets to a project. You must be able to authenticate with Watson Machine Learning to save to a deployment space.

This topic describes using the Python client to save a model to a Watson Studio project. Steps for saving a model directly to a deployment are detailed in [Deploying models to Watson Machine Learning Server](#)

## Accessing the Python client while connected to the internet

[Watson Machine Learning Python client library](#) 

## Authenticating with the Python client for Watson Studio Desktop

---

To authenticate with the Python client for Watson Studio Desktop, start by importing the Project library to get the project details you need to connect.

```
# Import libs
from project_lib import Project
from project_lib.utils import environment
project = Project.access()
projectId = project.get_metadata()["metadata"]["guid"]
test_server = environment.get_common_api_url()
wml_credentials = {
    "instance_id": "wsd_local",
    "url" : test_server,
    "version": "1.1"
}
client = WatsonMachineLearningAPIClient(wml_credentials)
client.set.default_project(project_id)
```

Where:

- "instance id" is "wsd\_local"
- "url" represents the local hostname of Watson Studio Desktop and the associated port number. This information is extracted using the Project library and written to the test\_server variable.
- "version" represents the version of Watson Studio Desktop

## Python commands available for Watson Studio Desktop

These commands that are supported for use with Watson Studio Desktop let you set and get details about a Watson Studio project, and work with the data assets for the project.

For details on supported commands and options, see the Python client documentation at

## Example of saving a model to a project

---

This example shows how to save a trained model to a project in Watson Studio Desktop. You can view the full sample notebook in the github repository: <https://github.com/IBMDaScience/sample-notebooks/blob/master/Desktop/notebooks/Use%20scikit-learn%20to%20predict%20car%20price%20-%20Batch.ipynb>

## Connect to the Python client and specify the default project

1. First, import the watson-machine-learning-client module and authenticate the service instance.  
**Important:** On Windows, you must use "Run as Administrator" to install libraries or packages in your notebook.

```
!pip install watson-machine-learning-client-V4
```

then:

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient
```

2. To associate the python client with Watson Studio Desktop, import the project library and use the following credentials.

```
from project_lib.utils import environment
url = environment.get_common_api_url()

wml_credentials = {
    "instance_id": "wsd_local",
    "url": url,
    "version": "1.1"
}
```

3. Next, instantiate a WatsonMachineLearningAPIClient object.

```
client = WatsonMachineLearningAPIClient(wml_credentials)
```

4. Set the default project.

```
from project_lib import Project
project = Project.access()
project_id = project.get_metadata()["metadata"]["guid"]

client.set.default_project(project_id)
```

**Out: 'SUCCESS'**

## Save the model to the project

1. First, create the model metadata.

```
meta_props={
    client.repository.ModelMetaNames.NAME: "Car Price Regression model",
    client.repository.ModelMetaNames.RUNTIME_UID: "scikit-learn_0.20-py3.6",
    client.repository.ModelMetaNames.TYPE: "scikit-learn_0.20"
}
```

2. Extract the model UID from the metadata.

```
model_artifact = client.repository.store_model(mod,
                                                meta_props=meta_props,
                                                training_data=X_train,
                                                training_target=y_train)

model_uid = client.repository.get_model_uid(model_artifact)
print("Model UID = " + model_uid)
```

**Out: Model UID = 710e0e32-3c60-400a-878a-682f8ee87287**

3. Get the saved model metadata:

```
model_details = client.repository.get_details(model_uid)
from pprint import pprint
pprint(model_details)
```

Results in:

```
{'entity': {'content_status': {'state': 'persisted'},
  'label_column': 'l1',
  'name': 'Car Price Regression model',
  'runtime': {'href': '/v4/runtimes/scikit-learn_0.20-py3.6'},
  'space': {'href': '/v4/spaces/d4281f85-bc53-4d41-b1be-5296d8950010'},
  'training_data_references': [{'connection': {'access_key_id': 'not_applicable',
    'endpoint_url': 'not_applicable',
    'secret_access_key': 'not_applicable'},
    'location': {'bucket': 'not_applicable'},
    'schema': {'fields': [{'name': 'f0',
      'type': 'float'},
      {'name': 'f1',
      'type': 'float'},
      {'name': 'f2',
      'type': 'float'},
      {'name': 'f3',
      'type': 'float'},
      {'name': 'f4',
      'type': 'float'},
      {'name': 'f5',
      'type': 'float'},
      .
      .
      .
      {'name': 'f39',
      'type': 'float'}]},
    'id': '1',
    'type': 'ndarray'},
    'type': 'fs'}]},
  'type': 'scikit-learn_0.20'},
'metadata': {'created_at': '2020-01-15T18:42:07.002Z',
  'guid': '710e0e32-3c60-400a-878a-682f8ee87287',
  'href': '/v4/models/710e0e32-3c60-400a-878a-682f8ee87287?space_id=d4281f85-bc53-4d41-b1be-5296d8950010',
  'id': '710e0e32-3c60-400a-878a-682f8ee87287',
  'modified_at': '2020-01-15T18:42:09.002Z',
  'owner': '1000330999'}}
```

4. List all of the stored models using the `list_models` method.

```
client.repository.list_models()
```

Results in:

GUID	NAME	CREATED	TYPE
710e0e32-3c60-400a-878a-682f8ee87287	Car Price Regression model	2020-01-15T18:42:07.002Z	scikit-learn_0.20

From the list of stored models, you can see that model is successfully saved to the project. You can see the saved model listed as a project asset by clicking on the project name in the navigation breadcrumb at the top of the application.

You can view and download complete notebook samples that demonstrate saving models to projects from this repository: <https://github.com/IBMDaScience/sample-notebooks/tree/master/Desktop/notebooks>

## Next steps

If you can connect to Watson Machine Learning and you want to deploy a saved model:

- Follow the steps in [Creating deployments from a space](#) to deploy using the Deployment Spaces interface.
- Follow the steps in [Deploying using the Python client](#) to deploy a model in a notebook.

## Deploying from the Python client (Version 1.1)

Starting with a trained machine learning model, save the model to IBM Watson Machine Learning using the Python client, then deploy and score it.

- [Python client library reference](#)
- [Persisting a model to Watson Machine Learning](#)
- [Creating and scoring an online deployment](#)



- [Creating and scoring a batch deployment](#)

You will also learn how to list all of the deployments for given space:

- [Listing deployment jobs for given space](#)

**Note:** This topic provides excerpts from a model to demonstrate techniques, but it is not intended as a working tutorial. You can view and download complete notebook samples from this repository:  
<https://github.com/IBMDaScience/sample-notebooks/tree/master/Desktop/notebooks>

## Prerequisites

---

To deploy a model to IBM Watson Machine Learning, you need:

- A trained model
- The [credentials](#) to connect to the server.

## Watson Machine Learning Python client library reference

---

### Downloading the Python client for off-line use

If you will be disconnected from the internet when you build your notebook models, connect and download the Python client for off-line use.

**Note:** Python client version 3.6 or newer is required for use with Watson Machine Learning Server.

The watson-machine-learning-client-V4 has a list of dependencies in the file `requires.txt`:

- requests
- urllib3
- pandas
- certifi
- tqdm
- lomond
- tabulate
- ibm-cos-sdk

While connected to internet, open a shell or cmd window and run the following commands to download all dependent files to a specific directory:

```
`pip download -d /some/particular/directory/ requests urllib3 pandas certifi tqdm
lomond tabulate ibm-cos-sdk`
```

Some packages might require that you download specific versions. The command to load a specific version is:

```
mac: pip download -d /some/particular/directory "idna<2.6,>=2.5"
win: pip download -d c:\some\particular\directory "idna<2.6,>=2.5"
```

To download watson-machine-learning-client-V4:

```
mac: pip download -d /some/particular/directory watson-machine-learning-client-V4
win: pip download -d c:\some\particular\directory watson-machine-learning-client-V4
```

For off-line installation, run pip install and specify the directory where all dependencies were downloaded:

**Important:** On Windows, you must use "Run as Administrator" to install libraries or packages in your notebook.

```
mac: pip install --no-index --find-links=/some/particular/directory watson-machine-
learning-client-V4
win: pip install --no-index --find-links=c:\some\particular\directory watson-machine-
```

## Accessing the Python client while connected to the internet

You can access a reference to all of the Python commands for Watson Machine Learning here: [Watson Machine Learning Python client library](#)

Before you can deploy, you must save the model programmatically to the associated deployment space on Watson Machine Learning.

**Note:** After you save a model to a deployment space, you can view it from the space and create a deployment in the user interface. For details, see [Creating a deployment](#).

This topic steps you through the process of saving, then deploying, a sample model.

## Save a model to the repository

---

1. Install the watson-machine-learning-client package to store your model in the Watson Machine Learning repository. For example:

```
!pip install --upgrade watson-machine-learning-client-V4
```

2. Next, authenticate with the server. This sample shows authentication from a notebook. Note that fictional credentials are used in this example:

```
wml_credentials = {
    "instance_id": "wml_local",
    "url" : "https://<WML Server Hostname>:31843",
    "username": "<USER NAME>",
    "password": "<PASSWORD>",
    "version": "1.1"
}
```

3. Initialize the client with the credentials:

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient
client = WatsonMachineLearningAPIClient(wml_credentials)
```

4. Create a deployment space.

```
space_details = client.spaces.store(meta_props=
{client.spaces.ConfigurationMetaNames.NAME: "<SPACE_NAME>"})
space_id = client.spaces.get_uid(space_details)
space_id
```

The output is the space ID. For example: **Out: '62748c9a-5353-46e7-8fa2-c3ec971eccf2'**

5. Set the space.

```
client.set.default_space(space_id)
```

6. Store the trained model to the repository and get the model ID. To do so, enter the absolute path of the trained model file, as well as the model name, model type and model runtime.

```
model_details = client.repository.store_model(model="<Trained Model
file>", meta_props={
    client.repository.ModelMetaNames.NAME: "<Model Name>",
    client.repository.ModelMetaNames.TYPE: "<model type>",
    client.repository.ModelMetaNames.RUNTIME_UID: "<model runtime>"
})
model_id = client.repository.get_model_uid(model_details)
```

For example, a trained SPSS model might have metadata like this:

```

model_details =
client.repository.store_model(model="example.com/my_spss_model",meta_props={
    client.repository.ModelMetaNames.NAME:"my_spss_model",
    client.repository.ModelMetaNames.TYPE:"spss-modeler_18.1",
    client.repository.ModelMetaNames.RUNTIME_UID: "spss-modeler_18.1"}
)
model_id = client.repository.get_model_uid(model_details)

```

7. Print the model ID:

```
print(model_id)
```

**Out: 8a8e68a6-038c-4e13-90d6-729bee9a99cd**

## Create an online deployment

Follow these steps to create an online deployment where you can get realtime scores for your model.

1. Create and name the online deployment for the model you persisted.

```

dep_details =
client.deployments.create(artifact_uid=model_id,meta_props={ client.deployments.ConfigurationMetaNames.NAME:"
<ONLINE_DEPLOYMENT_NAME>",client.deployments.ConfigurationMetaNames.ONLINE:{}}) Out:
Successfully finished deployment creation, deployment_uid='9a095a83-3c91-4d10-8a5c-f967a7702902'

```

2. Get the deployment ID.

```
dep_id = client.deployments.get_uid(dep_details)
```

3. To score the deployed model construct the model payload, following the schema of the model. `fields =`

```

["Age", "Sex", "BP", "Cholesterol", "Na", "K"]
values = [[23, "F", "HIGH", "HIGH", 0.792535, 0.031258]]

scoring_payload = {
    client.deployments.ScoringMetaNames.INPUT_DATA: [{
        "fields": fields,
        "values": values
    }]
}

```

4. Run the `score` function to generate the prediction.

```
client.deployments.score(deployment_id=dep_id,meta_props=scoring_payload)
```

```

Out[13]: {'predictions': [{'fields': ['Age',
    'Sex',
    'BP',
    'Cholesterol',
    'Na',
    'K',
    '$N-Drug',
    '$NC-Drug'],
    'values': [[23,
    'F',
    'HIGH',
    'HIGH',
    0.792535,
    0.031258,
    'drugY',
    0.9999929901781122]]}]}

```

5. Finally, delete the deployment.

```
client.deployments.delete(dep_id)
```

## Create a batch deployment

Create a batch deployment to submit a data asset containing multiple payloads, and write the resulting predictions to an output file.

1. Create and name the batch deployment.

```
dep_details = client.deployments.create(artifact_uid=model_id,meta_props={
client.deployments.ConfigurationMetaNames.NAME:"<BATCH_DEPLOYMENT_NAME>",
client.deployments.ConfigurationMetaNames.BATCH:{},
client.deployments.ConfigurationMetaNames.COMPUTE:{ "name":"S", "nodes":1}
})
```

**Out: Successfully finished deployment creation, deployment\_uid='e6936222-4efb-4f03-997e-e1cf0ce29991'**

2. Get the deployment ID.

```
dep_id = client.deployments.get_uid(dep_details)
print(dep_id)
```

3. Create the data asset for the batch deployment, entering the input data asset file name and input data asset name. Make sure you use a supported data asset file formats for the runtime type.

```
asset_details = client.data_assets.create(name="
<INPUT_DATA_ASSET_NAME>",file_path="<Input Data Asset File>")
```

4. Get the asset ID.

```
asset_id = client.data_assets.get_uid(asset_details)
```

5. Get the Href for the input data asset.

```
asset_href = client.data_assets.get_href(asset_details)
```

6. Print the asset ID and asset Href.

```
print(asset_id)
print(asset_href)
```

**Out: 8c11502f-e047-4822-bc6e-7d6c93a24336 /v2/assets/8c11502f-e047-4822-bc6e-7d6c93a24336?space\_id=62748c9a-5353-46e7-8fa2-c3ec971eccf2**

## Create and run the batch deployment job

1. Construct the deployment job payload. Enter the input data asset Href and name, and the output data asset name and description.

```
job_payload_ref = {
client.deployments.ScoringMetaNames.INPUT_DATA_REFERENCES: [{
    "id": "input_data",
    "name": "<INPUT_DATA_ASSET_NAME>",
    "type": "data_asset",
    "connection": {},
    "location": {
        "href": "<INPUT_DATA_ASSET_HREF>"
    }
}],
client.deployments.ScoringMetaNames.OUTPUT_DATA_REFERENCE: {
    "type": "data_asset",
    "connection": {},
    "location": {
        "name": "<OUTPUT_DATA_ASSET_NAME>_{}.csv".format(dep_id),
        "description": "<DESCRIPTION>"
    }
}
```

```
}
}
```

2. Enter the batch job and supply the deployment ID.

```
job =
client.deployments.create_job(deployment_id=dep_id,meta_props=job_payload_ref)
```

3. Get the job ID.

```
job_id = client.deployments.get_job_uid(job)
```

4. Execute the job until the state of Get Job Status changes to 'completed', 'failed' or 'cancelled'

```
client.deployments.get_job_status(job_id)
```

**Out: {'state': 'completed', 'running\_at': '2019-12-06T13:26:03.699Z', 'completed\_at': '2019-12-06T13:26:18.515Z'}**

5. Get the job details by ID.

```
client.deployments.get_job_details(job_id)
```

Results in:

```
Out[32]: {'metadata': {'guid': '1e7113d2-5b5b-47d4-9420-b7a3a1f7e946',
'href': '/v4/deployment_jobs/1e7113d2-5b5b-47d4-9420-b7a3a1f7e946',
'created_at': '2019-12-06T13:25:49.927Z',
'parent': {'href': ''}},
'entity': {'deployment': {'href': '/v4/deployments/e6936222-4efb-4f03-997e-e1cf0ce29991'},
'scoring': {'input_data_references': [{'name': 'drug_ref_input',
'location': {'href': '/v2/assets/8c11502f-e047-4822-bc6e-7d6c93a24336?space_id=62748c9a-5353-46e7-8fa2-c3ec971eccf2'},
'id': 'input_data',
'schema': {},
'connection': {},
'type': 'data_asset'}],
'output_data_reference': {'type': 'data_asset',
'connection': {},
'location': {'name': 'drug_predictions_e6936222-4efb-4f03-997e-e1cf0ce29991.csv',
'description': 'testing csv file',
'href': '/v2/assets/7249af68-626c-48ce-9b9c-2a7d4b9e0270?space_id=62748c9a-5353-46e7-8fa2-c3ec971eccf2'}},
'status': {'state': 'completed',
'running_at': '2019-12-06T13:26:03.699Z',
'completed_at': '2019-12-06T13:26:18.515Z'}}}}
```

6. Download the prediction results to the output file.

```
client.data_assets.download(<OUTPUT_DATA_ASSET_ID>, <PREDICTION_RESULT_FILENAME>)
```

7. Finally, clean up the deployment by deleting the batch deployment, the saved model, and the input data asset.

```
client.deployments.delete(dep_id)
client.repository.delete(model_id)
client.data_assets.delete(asset_id)
```

## Listing deployment jobs for given space

To list all of the deployment jobs currently available for a site, use the `list_jobs` method with a specified space id:

```
client.deployment.list_jobs(<deployment_id>)
```

Note that only deployment jobs are listed. Deleting a deployment job means it will no longer be listed for the space.

## Deploying functions (Version 1.1)

This topic describes how to deploy Python functions.

You can deploy Python functions in Watson Machine Learning the same way that you can deploy models. Your tools and apps can use the Watson Machine Learning Python client or REST API to send data to your deployed functions the same way that they send data to deployed models. Deploying functions gives you the ability to hide details (such as

credentials), preprocess data before passing it to models, perform error handling, and include calls to multiple models, all within the deployed function instead of in your application.

**Note:** This topic provides excerpts from a model to demonstrate techniques, but it is not intended as a working tutorial. You can view and download samples from this repository: <https://github.com/IBMDaScience/sample-notebooks/tree/master/Desktop/notebooks>

## Deploying a function from a deployment space

---

This topic describes how to deploy a function using the Python client, but you can also deploy a function from a deployment space via the user interface. For details on creating and deploying from a deployment space, see [Deployment spaces](#).

### Watson Machine Learning Python client library reference

[Watson Machine Learning Python client library](#) 

There are five basic steps for creating and deploying functions in Watson Machine Learning:

1. [Define the function](#)
2. [Connect to Watson Studio Desktop and specify a project](#)
3. [Store the function in the repository](#)
4. [Deploy the stored function](#)
5. [Send data to the function for processing](#)

#### Step 1: Define the function

To define a function, create a *Python closure* with a nested function named 'score'.

This topic assumes you have a Python function. For a sample of how to create one, download a [notebook demonstrating how to scrape a Web page](#) from the collection of sample notebooks.

#### Step 2: Connect to Watson Machine Learning and specify a project

This step describes saving a Python function to a project in Watson Studio Desktop. This step does not require a connection to Watson Machine Learning Server.

1. First, import the `watson-machine-learning-client` module and authenticate the service instance.  
**Important:** On Windows, you must use "Run as Administrator" to install libraries or packages in your notebook.

```
!pip install --upgrade watson-machine-learning-client-V4
```

then:

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient
```

2. Use the following credentials to associate the python client with Watson Studio Desktop.

```
from project_lib.utils import environment
url = environment.get_common_api_url()

wml_credentials = {
    "instance_id": "wsd_local",
    "url": url,
    "version": "1.1"
}
```

3. Instantiate a `WatsonMachineLearningAPIClient` object.

```
client = WatsonMachineLearningAPIClient(wml_credentials)
```

then:

```
client.version
```

**Out: '1.0.60'**

4. Initialize the client with the credentials:

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient
client = WatsonMachineLearningAPIClient(wml_credentials)
```

5. Set the default project where you will save the function.

```
from project_lib import Project
project = Project.access()
project_id = project.get_metadata()["metadata"]["guid"]

client.set.default_project(project_id)
#client.set.default_project(client.default_project_id)
```

**Out: 'Success'**

When you store the function it will be saved to the project you specified.

### Step 3: Store the function in the repository

To store the function for deployment, you must connect to Watson Machine Learning Server.

1. Enter your credentials to connect to Watson Machine Learning Server.

```
wml_credentials = {
    "url": "<URL>:31843",
    "username": "----",
    "password": "----",
    "instance_id": "wml_local",
    "version": "1.1"
}
```

2. Next, instantiate a WatsonMachineLearningAPIClient object.

```
client = WatsonMachineLearningAPIClient(wml_credentials)
```

then:

```
client.version
```

**Out: '1.0.327'**

3. (Optional) Create a new deployment space. To use an existing deployment space, skip this step and enter the name of the space in the next step.

```
space_details = client.spaces.store(meta_props=
{client.spaces.ConfigurationMetaNames.NAME: "YOUR DEPLOYMENT SPACE"})
space_uid = client.spaces.get_uid(space_details) print("Space UID = " + space_uid)
```

4. Get the ID for the deployment space:

```
def guid_from_space_name(client, space_name):
    instance_details = client.service_instance.get_details()
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']['name'] ==
space_name) ['metadata'] ['guid'])
```

5. Enter the details for the deployment space, putting the name of your deployment space in place of 'YOUR DEPLOYMENT SPACE'.

```
space_uid = guid_from_space_name(client, 'YOUR DEPLOYMENT SPACE')
print("Space UID = " + space_uid)
```

**Out: Space UID = b8eb6ec0-dcc7-425c-8280-30a1d7a9c58a**

6. Enter the default space. This is a required step.

```
client.set.default_space(space_uid)
```

**Out: 'SUCCESS'**

## Step 4: Save the function

1. Create the function metadata.

```
meta_props = {
    client.repository.FunctionMetaNames.NAME: "Web scraping python function",
    client.repository.FunctionMetaNames.RUNTIME_UID: "ai-function_0.1-py3.6"
    # client.repository.FunctionMetaNames.SPACE_UID: space_uid
}
```

2. Extract the function UID from the details.

```
function_artifact = client.repository.store_function(meta_props=meta_props,
function=my_deployable_function)
function_uid = client.repository.get_function_uid(function_artifact)
print("Function UID = " + function_uid)
```

**Function UID = 0f263463-21ec-4d2f-a277-2a7525f64b4e**

3. Get the saved function metadata from Watson Machine Learning using the function UID.

```
function_details = client.repository.get_details(function_uid)
from pprint import pprint
pprint(function_details)
```

4. To confirm the function was saved, list all of the stored functions using the `list_functions` method.

```
client.repository.list_functions()
```

## Step 5: Deploy the stored function

1. Deploy the Python function to the deployment space by creating deployment metadata and using the function UID obtained in the previous section.

```
deploy_meta = {
    client.deployments.ConfigurationMetaNames.NAME: "Web scraping python function
deployment",
    client.deployments.ConfigurationMetaNames.ONLINE: {}
}
```

2. Create the deployment.

```
deployment_details = client.deployments.create(function_uid,
meta_props=deploy_meta)
```



```
#####

Synchronous deployment creation for uid: '0f263463-21ec-4d2f-a277-2a7525f64b4e' started

#####

initializing..
ready

-----
Successfully finished deployment creation, deployment_uid='f451cc11-e1a4-4b74-a361-cca0ab457298'
-----
```

### 3. View the deployment details.

```
deployment_details

{'metadata': {'parent': {'href': ''},
  'guid': 'f451cc11-e1a4-4b74-a361-cca0ab457298',
  'modified_at': '',
  'created_at': '2020-01-14T00:06:22.171Z',
  'href': '/v4/deployments/f451cc11-e1a4-4b74-a361-cca0ab457298'},
 'entity': {'name': 'Web scraping python function deployment',
  'custom': {},
  'online': {},
  'description': '',
  'space': {'href': '/v4/spaces/b8eb6ec0-dcc7-425c-8280-30a1d7a9c58a'},
  'status': {'state': 'ready',
  'online_url': {'url': 'https://wmlserver-svt.ml.test.cloud.ibm.com:31843/v4/deployments/f451cc11-e1a4-4b74-a361-cca0ab457298/predictions'}},
  'asset': {'href': '/v4/functions/0f263463-21ec-4d2f-a277-2a7525f64b4e?space_id=b8eb6ec0-dcc7-425c-8280-30a1d7a9c58a'},
  'auto_redeploy': False}}
```

### 4. To confirm that the deployment was created successfully, list all deployments.

```
client.deployments.list()
```

GUID	NAME	STATE	CREATED	ARTIFACT_TYPE
f451cc11-e1a4-4b74-a361-cca0ab457298	Web scraping python function deployment	ready	2020-01-14T00:06:22.171Z	function

## Step 5: Send data to the function for processing

Follow these steps to score the function and return a prediction.

#### 1. List the function you plan to score.

```
client.repository.list_functions()
```

#### 2. Prepare the scoring payload, matching the schema of the function.

```
job_payload = {
    client.deployments.ScoringMetaNames.INPUT_DATA: [{
#       "input_data": [{
            'fields': ['url'],
            'values': [
                'https://www.ibm.com/cloud/machine-learning'
            ]
        }]
    }
}
pprint(job_payload)
```

```
{'input_data': [{'fields': ['url'], 'values': ['https://www.ibm.com/
cloud/machine-learning']}]}
```

#### 3. Generate the prediction and display the results:

```
job_details = client.deployments.score(deployment_uid, job_payload)
pprint(job_details['predictions'][0]['values'][0][:10])
```

## SPSS Modeler tutorials

---

These tutorials are based on assets included with the example project that's installed with the product. Under My Projects, click Example Project.

See [Example projects](#) for more information.

- **Introduction to modeling**  
A model is a set of rules, formulas, or equations that can be used to predict an outcome based on a set of input fields or variables. For example, a financial institution might use a model to predict whether loan applicants are likely to be good or bad risks, based on information that is already known about past applicants.
- **Automated modeling for a flag target**  
With the Auto Classifier node, you can automatically create and compare a number of different models for either flag (such as whether or not a given customer is likely to default on a loan or respond to a particular offer) or nominal (set) targets.
- **Automated modeling for a continuous target**  
You can use the Auto Numeric node to automatically create and compare different models for continuous (numeric range) outcomes, such as predicting the taxable value of a property. With a single node, you can estimate and compare a set of candidate models and generate a subset of models for further analysis. The node works in the same manner as the Auto Classifier node, but for continuous rather than flag or nominal targets.
- **Automated data preparation**  
Preparing data for analysis is one of the most important steps in any data-mining project - and traditionally, one of the most time consuming. The Auto Data Prep node handles the task for you, analyzing your data and identifying fixes, screening out fields that are problematic or not likely to be useful, deriving new attributes when appropriate, and improving performance through intelligent screening techniques.
- **Drug treatment - exploratory graphs**  
In this example, imagine you're a medical researcher compiling data for a study. You've collected data about a set of patients, all of whom suffered from the same illness. During their course of treatment, each patient responded to one of five medications. Part of your job is to use data mining to find out which drug might be appropriate for a future patient with the same illness.
- **Screening predictors**  
The Feature Selection node helps you identify the fields that are most important in predicting a certain outcome. From a set of hundreds or even thousands of predictors, the Feature Selection node screens, ranks, and selects the predictors that may be most important. Ultimately, you may end up with a quicker, more efficient model - one that uses fewer predictors, runs more quickly, and may be easier to understand.
- **Reducing input data string length**  
For binomial logistic regression, and auto classifier models that include a binomial logistic regression model, string fields are limited to a maximum of eight characters. Where strings are more than eight characters, you can recode them using a Reclassify node.
- **Classifying telecommunications customers**  
Logistic regression is a statistical technique for classifying records based on values of input fields. It is analogous to linear regression, but takes a categorical target field instead of a numeric one.
- **Telecommunications churn**  
Logistic regression is a statistical technique for classifying records based on values of input fields. It is analogous to linear regression, but takes a categorical target field instead of a numeric one.
- **Forecasting bandwidth utilization**  
An analyst for a national broadband provider is required to produce forecasts of user subscriptions to predict utilization of bandwidth. Forecasts are needed for each of the local markets that make up the national subscriber base.
- **Forecasting catalog sales**  
A catalog company is interested in forecasting monthly sales of its men's clothing line, based on 10 years of their sales data.

- **Making offers to customers (self-learning)**

The Self-Learning Response Model (SLRM) node generates and enables the updating of a model that allows you to predict which offers are most appropriate for customers and the probability of the offers being accepted. These sorts of models are most beneficial in customer relationship management, such as marketing applications or call centers.

- **Retail sales promotion**

This example deals with fictitious data that describes retail product lines and the effects of promotion on sales.

- **Condition monitoring**

This example concerns monitoring status information from a machine and the problem of recognizing and predicting fault states.

- **Hotel satisfaction example for Text Analytics**

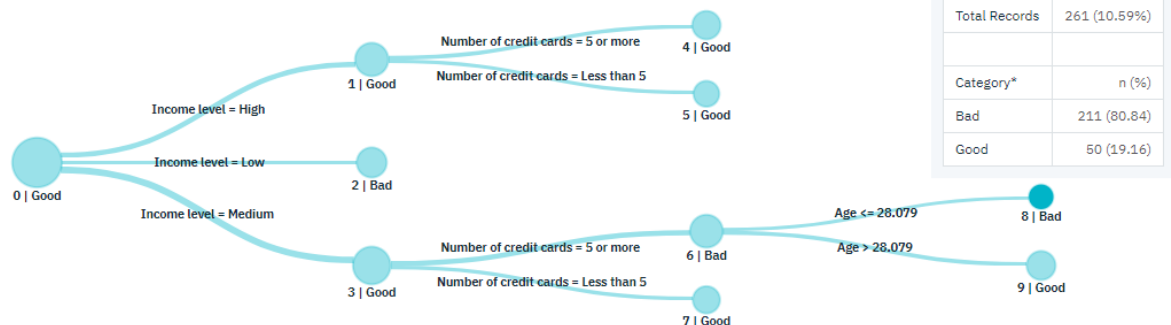
The Text Analytics nodes offer powerful text analytic capabilities, which use advanced linguistic technologies and Natural Language Processing (NLP) to rapidly process a large variety of unstructured text data and, from this text, extract and organize the key concepts. Text Analytics can also group these concepts into categories.

## Introduction to modeling

A model is a set of rules, formulas, or equations that can be used to predict an outcome based on a set of input fields or variables. For example, a financial institution might use a model to predict whether loan applicants are likely to be good or bad risks, based on information that is already known about past applicants.

The ability to predict an outcome is the central goal of predictive analytics, and understanding the modeling process is the key to using flows in Watson Studio.

Figure 1. A decision tree model



This example uses a *decision tree* model, which classifies records (and predicts a response) using a series of decision rules. For example:

```
IF income = Medium
AND cards < 5
THEN -> 'Good'
```

While this example uses a CHAID (Chi-squared Automatic Interaction Detection) model, it is intended as a general introduction, and most of the concepts apply broadly to other modeling types in Watson Studio.

To understand any model, you first need to understand the data that goes into it. The data in this example contains information about the customers of a bank. The following fields are used:

Field name	Description
Credit_rating	Credit rating: 0=Bad, 1=Good, 9=missing values
Age	Age in years
Income	Income level: 1=Low, 2=Medium, 3=High
Credit_cards	Number of credit cards held: 1=Less than five, 2=Five or more

Field name	Description
Education	Level of education: 1=High school, 2=College
Car_loans	Number of car loans taken out: 1=None or one, 2=More than two

The bank maintains a database of historical information on customers who have taken out loans with the bank, including whether or not they repaid the loans (Credit rating = Good) or defaulted (Credit rating = Bad). Using this existing data, the bank wants to build a model that will enable them to predict how likely future loan applicants are to default on the loan.

Using a decision tree model, you can analyze the characteristics of the two groups of customers and predict the likelihood of loan defaults.

This example uses the flow named Introduction to Modeling, available in the example project installed with the product. The data file is `tree_credit.csv`.

Let's take a look at the flow.

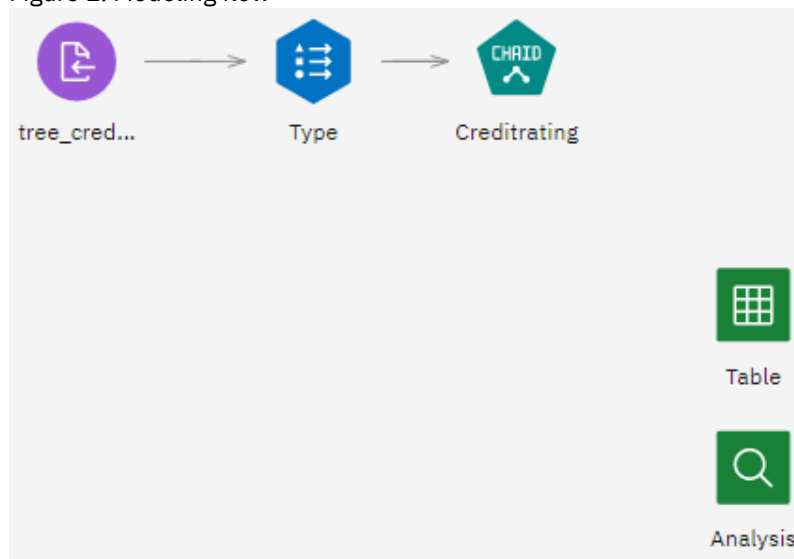
1. On the My Projects screen, click Example Project.
2. Scroll down to the Modeler flows section, click View all, and select the Introduction to Modeling flow.

- [Building the flow](#)
- [Browsing the model](#)
- [Evaluating the model](#)
- [Scoring records](#)
- [Summary](#)

**Parent topic:** [SPSS Modeler tutorials](#)

## Building the flow

Figure 1. Modeling flow



To build a flow that will create a model, we need at least three elements:

- A Data Asset node that reads in data from an external source, in this case a .csv data file
- An Import or Type node that specifies field properties, such as measurement level (the type of data that the field contains), and the role of each field as a target or input in modeling
- A modeling node that generates a model nugget when the flow runs

In this example, we're using a CHAID modeling node. CHAID, or Chi-squared Automatic Interaction Detection, is a classification method that builds decision trees by using a particular type of statistics known as chi-square statistics

to work out the best places to make the splits in the decision tree.

If measurement levels are specified in the source node, the separate Type node can be eliminated. Functionally, the result is the same.

This flow also has Table and Analysis nodes that will be used to view the scoring results after the model nugget has been created and added to the flow.

The Data Asset import node reads data in from the sample tree\_credit.csv data file.

The Type node specifies the *measurement level* for each field. The measurement level is a category that indicates the type of data in the field. Our source data file uses three different measurement levels:

A Continuous field (such as the Age field) contains continuous numeric values, while a Nominal field (such as the Credit rating field) has two or more distinct values, for example Bad, Good, or No credit history. An Ordinal field (such as the Income level field) describes data with multiple distinct values that have an inherent order - in this case Low, Medium and High.

Figure 2. Setting the target and input fields with the Type node

Type

Settings

Default Mode ⓘ

☒ Read metadata ☐ Pass (do not scan)

Type Operations

Read Values

Clear All Values

Q Search in column Field

<input type="checkbox"/>	Field	Measure		Role		Value Mode	Values	Check	
<input type="checkbox"/>	abc Credit ratir	Flag	▼	Target	▼	Instantiated ▼	Bad, Good	None	▼ ⚙
<input type="checkbox"/>	# Age	Continuous	▼	Input	▼	Instantiated ▼	20.002699523368...	None	▼ ⚙
<input type="checkbox"/>	abc Income lev	Ordinal	▼	Input	▼	Instantiated ▼	High, Low, Medium	None	▼ ⚙
<input type="checkbox"/>	abc Number of	Nominal	▼	Input	▼	Instantiated ▼	5 or more, Less tha...	None	▼ ⚙
<input type="checkbox"/>	abc Education	Nominal	▼	Input	▼	Instantiated ▼	College, High school	None	▼ ⚙
<input type="checkbox"/>	abc Car loans	Nominal	▼	Input	▼	Instantiated ▼	More than 2, None ...	None	▼ ⚙

For each field, the Type node also specifies a *role* to indicate the part that each field plays in modeling. The role is set to Target for the field Credit rating, which is the field that indicates whether or not a given customer defaulted on the loan. This is the target, or the field for which we want to predict the value.

Role is set to Input for the other fields. Input fields are sometimes known as predictors, or fields whose values are used by the modeling algorithm to predict the value of the target field.

The CHAID modeling node generates the model. In the node's properties, under FIELDS, the option Use custom field roles is available. We could select this option and change the field roles, but for this example we'll use the default targets and inputs as specified in the Type node.

1. Double-click the CHAID node (named Creditrating). The node properties are displayed.

Figure 3. CHAID modeling node properties

**Creditrating** ✎ ℹ

**Fields** ^

☐ Use custom field roles

Target ℹ

Credit rating ▼

Inputs ℹ

⊖ Add Columns ⊕

<input type="checkbox"/> Field Name
<input type="checkbox"/> Age
<input type="checkbox"/> Income level
<input type="checkbox"/> Number of credit cards
<input type="checkbox"/> Education

Weight ℹ

... ▼

**Objectives** ▼

Basic ▼

Stopping Rules ▼

Costs ▼

**Cancel** **Save**

Here there are several options where we could specify the kind of model we want to build.

We want a brand-new model, so under OBJECTIVES we'll use the default option Build new model.

We also just want a single, standard decision tree model without any enhancements, so we'll also use the default objective option Create a standard model.

Figure 4. CHAID modeling node objectives

Objectives ^

What do you want to do ⓘ

☒ Build new model

☐ Continue training existing model

What is your main objective? ⓘ

☒ Create a standard model

☐ Enhance model accuracy (boosting)

☐ Enhance model stability (bagging)

☐ Create a model for very large datasets (requires Server)

For this example, we want to keep the tree fairly simple, so we'll limit the tree growth by raising the minimum number of cases for parent and child nodes.

2. Under STOPPING RULES, select Use absolute value.
3. Set Minimum records in parent branch to 400.
4. Set Minimum records in child branch to 200.

Figure 5. Setting the stopping criteria for decision tree building

Stopping Rules ^

Stopping rules ⓘ

☐ Use percentage

☒ Use absolute value

Minimum records in parent branch

400

Minimum records in child branch

200

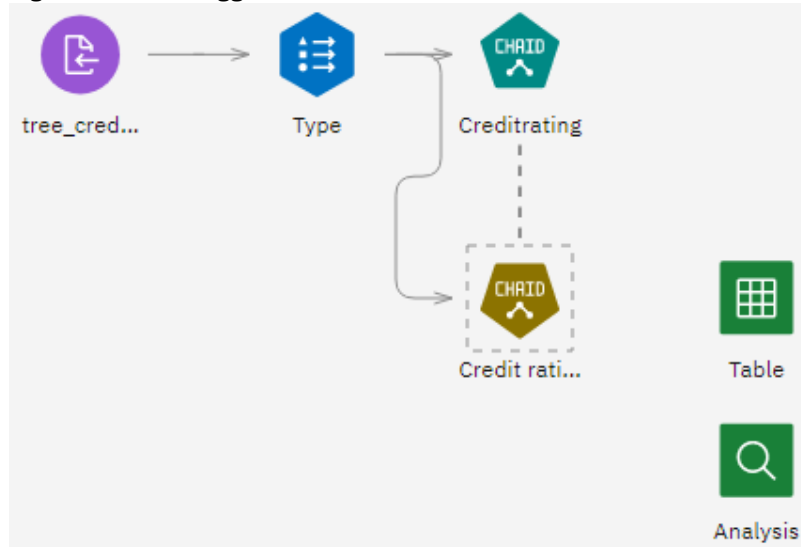
We can use all the other default options for this example, so click Save and then click the Run button on the toolbar to create the model. (Alternatively, right-click the CHAID node and choose Run from the context menu.)

**Parent topic:** [Introduction to modeling](#)

## Browsing the model

After running a flow, an orange model nugget is added to the canvas with a link to the modeling node from which it was created. To view the model details, right-click the model nugget and choose View Model.

Figure 1. Model nugget



In the case of the CHAID nugget, the CHAID Tree Model screen includes pages for Model Information, Predictor Importance, Top Decision Rules, and Tree Diagram. For example, you can see details in the form of a rule set - essentially a series of rules that can be used to assign individual records to child nodes based on the values of different input fields.

Figure 2. CHAID model nugget, rule set

Rule ID	Rule	Mode category	Record count	Record percentage	Rule confidence
2	((Income level = Low))	Bad	553	22.443	82.098
9	((Income level = Medium)) and ((Number of credit cards = 5 or more)) and ((Age > 28.079206))	Good	483	19.602	56.315
4	((Income level = High)) and ((Number of credit cards = 5 or more))	Good	455	18.466	82.418
7	((Income level = Medium)) and ((Number of credit cards = Less than 5))	Good	390	15.828	86.154
5	((Income level = High)) and ((Number of credit cards = Less than 5))	Good	322	13.068	96.894
8	((Income level = Medium)) and ((Number of credit cards = 5 or more)) and ((Age <= 28.079206))	Bad	261	10.593	80.843

For each decision tree terminal node - meaning those tree nodes that are not split further - a prediction of Good or Bad is returned. In each case, the prediction is determined by the *mode*, or most common response, for records that fall within that node.

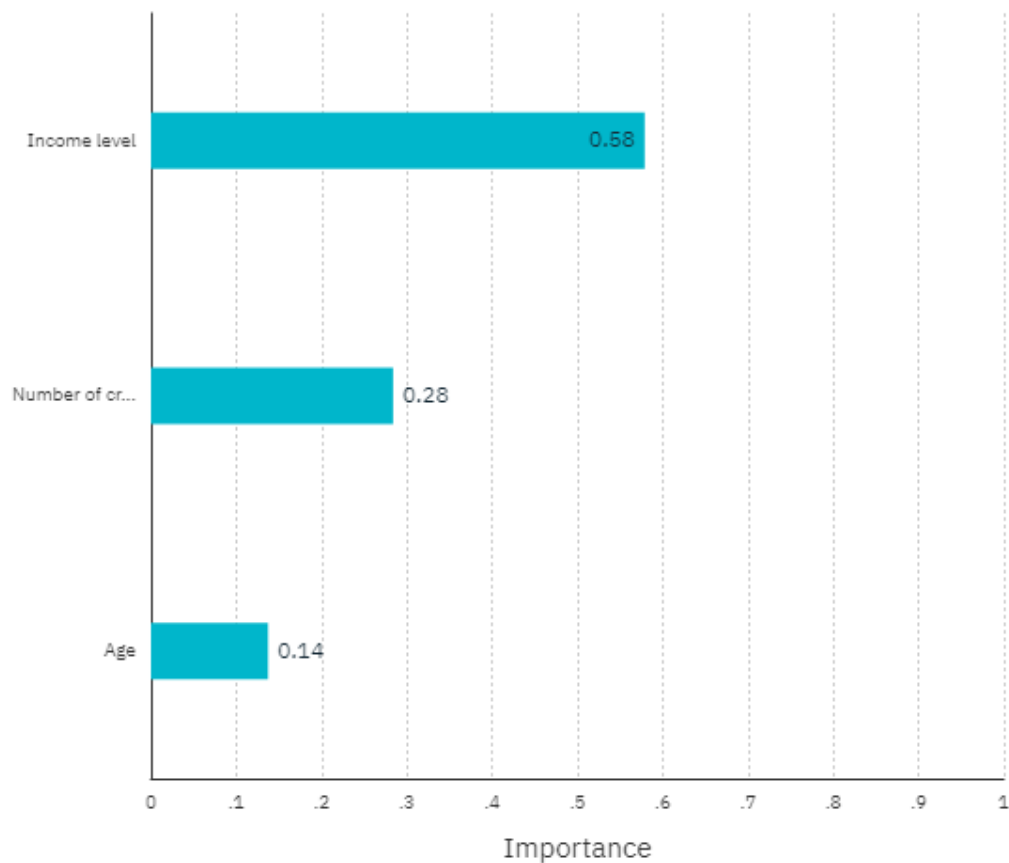
The Feature Importance chart shows the relative importance of each predictor in estimating the model. From this, we can see that Income level is easily the most significant in this case, with Number of credit cards being the next most significant factor.

Figure 3. Feature Importance chart



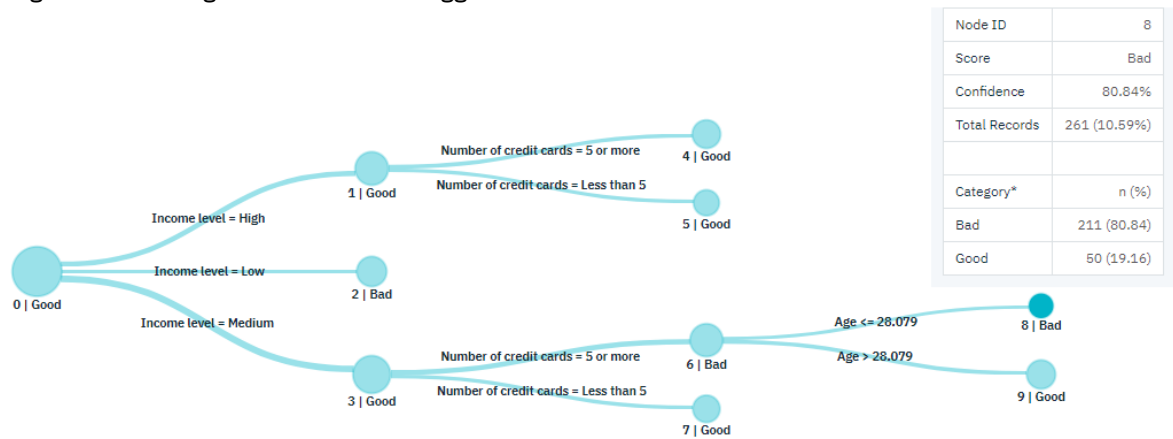
# Feature Importance

TARGET : CREDIT RATING



The Tree Diagram page displays the same model in the form of a tree, with a node at each decision point. Hover over branches and nodes to explore details.

Figure 4. Tree diagram in the model nugget



Looking at the start of the tree, the first node (node 0) gives us a summary for all the records in the data set. Just over 40% of the cases in the data set are classified as a bad risk. This is quite a high proportion, so let's see if the tree can give us any clues as to what factors might be responsible.

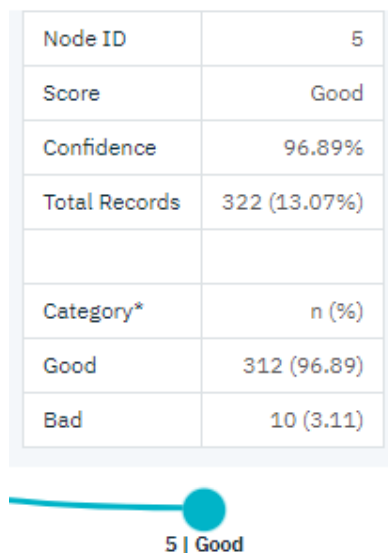
We can see that the first split is by Income level. Records where the income level is in the Low category are assigned to node 2, and it's no surprise to see that this category contains the highest percentage of loan defaulters. Clearly,

lending to customers in this category carries a high risk. However, almost 18% of the customers in this category actually *didn't* default, so the prediction won't always be correct. No model can feasibly predict every response, but a good model should allow us to predict the *most likely* response for each record based on the available data.

In the same way, if we look at the high income customers (node 1), we see that the vast majority (over 88%) are a good risk. But more than 1 in 10 of these customers has also defaulted. Can we refine our lending criteria to minimize the risk here?

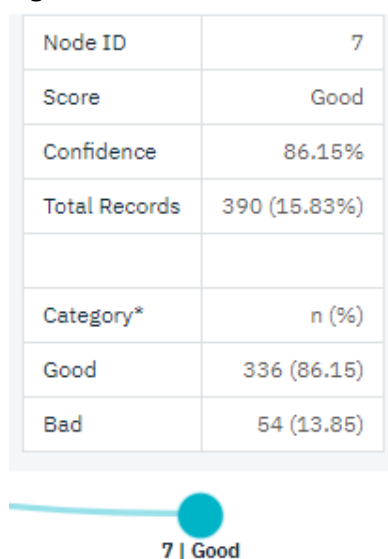
Notice how the model has divided these customers into two sub-categories (nodes 4 and 5), based on the number of credit cards held. For high-income customers, if we lend only to those with fewer than five credit cards, we can increase our success rate from 88% to almost 97% - an even more satisfactory outcome.

Figure 5. High-income customers with fewer than five credit cards



But what about those customers in the Medium income category (node 3)? They're much more evenly divided between Good and Bad ratings. Again, the sub-categories (nodes 6 and 7 in this case) can help us. This time, lending only to those medium-income customers with fewer than five credit cards increases the percentage of Good ratings from 58% to 86%, a significant improvement.

Figure 6. Tree view of medium-income customers

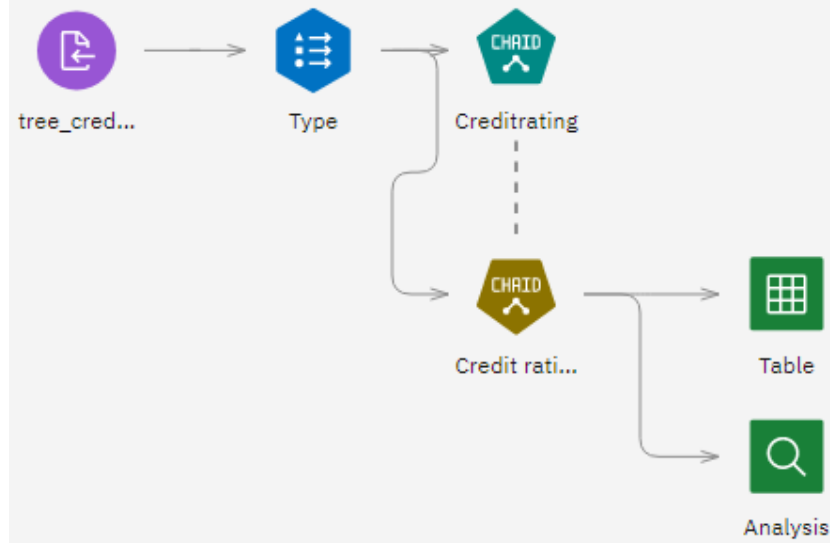


So, we've learned that every record that is input to this model will be assigned to a specific node, and assigned a prediction of Good or Bad based on the most common response for that node. This process of assigning predictions to individual records is known as *scoring*. By scoring the same records used to estimate the model, we can evaluate how accurately it performs on the training data - the data for which we know the outcome. Let's examine how to do this.

## Evaluating the model

We've been browsing the model to understand how scoring works. But to evaluate *how accurately* it works, we need to score some records and compare the responses predicted by the model to the actual results. We're going to score the same records that were used to estimate the model, allowing us to compare the observed and predicted responses.

Figure 1. Attaching the model nugget to output nodes for model evaluation



1. To see the scores or predictions, attach the Table node to the model nugget and then right-click the Table node and select Run. A table will be generated and added to the Outputs panel. Double-click it to open it.

The table displays the predicted scores in a field named `$R-Credit rating`, which was created by the model. We can compare these values to the original `Credit rating` field that contains the actual responses.

By convention, the names of the fields generated during scoring are based on the target field, but with a standard prefix. Prefixes `$G` and `$GE` are generated by the Generalized Linear Model, `$R` is the prefix used for the prediction generated by the CHAID model in this case, `$RC` is for confidence values, `$X` is typically generated by using an ensemble, and `$XR`, `$XS`, and `$XF` are used as prefixes in cases where the target field is a Continuous, Categorical, Set, or Flag field, respectively. Different model types use different sets of prefixes. A *confidence value* is the model's own estimation, on a scale from 0.0 to 1.0, of how accurate each predicted value is.

Figure 2. Table showing generated scores and confidence values

Credit rating	Age	Income level	Number of credit cards	Education	Car loans	\$R-Credit rating	\$RC-Credit rating
Bad	26.090	Low	5 or more	College	More than 2	Bad	0.820
Bad	31.628	Low	5 or more	College	More than 2	Bad	0.820
Bad	41.670	Medium	5 or more	College	More than 2	Good	0.563
Bad	23.075	Medium	5 or more	High school	None or 1	Bad	0.806
Bad	24.860	Low	5 or more	College	More than 2	Bad	0.820
Bad	23.628	Medium	5 or more	High school	More than 2	Bad	0.806
Bad	25.963	Medium	5 or more	High school	More than 2	Bad	0.806
Bad	44.005	Medium	Less than 5	College	None or 1	Good	0.860
Bad	21.957	Low	5 or more	High school	More than 2	Bad	0.820
Bad	37.583	Medium	5 or more	College	More than 2	Good	0.563
Bad	53.863	Low	5 or more	College	More than 2	Bad	0.820
Bad	29.797	Medium	5 or more	High school	More than 2	Good	0.563
Bad	26.724	Medium	5 or more	College	More than 2	Bad	0.806
Bad	24.202	Medium	5 or more	High school	More than 2	Bad	0.806

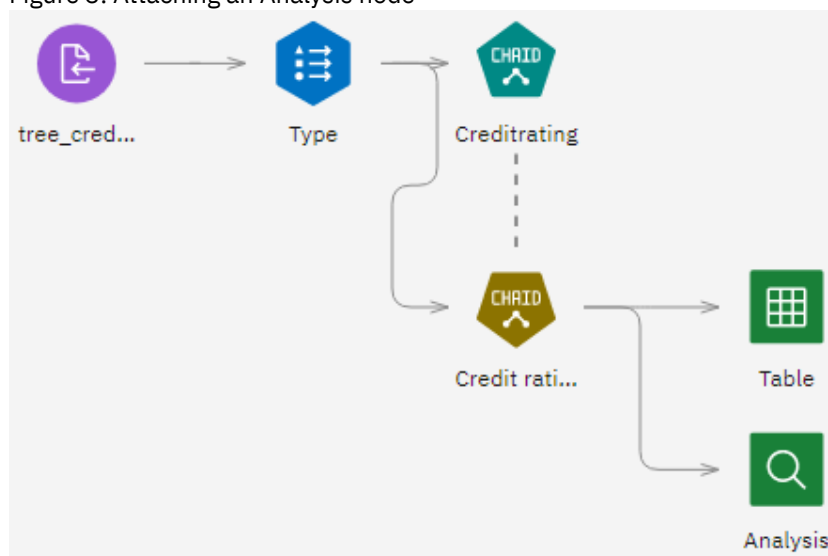
As expected, the predicted value matches the actual responses for many records but not all. The reason for this is that each CHAID terminal node has a mix of responses. The prediction matches the *most common* one, but will be wrong for all the others in that node. (Recall the 18% minority of low-income customers who did not default.)

To avoid this, we could continue splitting the tree into smaller and smaller branches, until every node was 100% pure - all Good or Bad with no mixed responses. But such a model would be extremely complicated and would probably not generalize well to other datasets.

To find out exactly how many predictions are correct, we could read through the table and tally the number of records where the value of the predicted field `$R-Credit rating` matches the value of `Credit rating`. Fortunately, there's a much easier way; we can use an Analysis node, which does this automatically.

2. Connect the model nugget to the Analysis node.
3. Right-click the Analysis node and select Run. An Analysis entry will be added to the Outputs panel. Double-click it to open it.

Figure 3. Attaching an Analysis node



The analysis shows that for 1960 out of 2464 records - over 79% - the value predicted by the model matched the actual response.

Figure 4. Analysis results comparing observed and predicted responses

Results for output field Credit rating		
Comparing SR-Credit rating with Credit rating		
Correct	1,960	79.55%
Wrong	504	20.45%
Total	2,464	

This result is limited by the fact that the records being scored are the same ones used to estimate the model. In a real situation, you could use a Partition node to split the data into separate samples for training and evaluation. By using one sample partition to generate the model and another sample to test it, you can get a much better indication of how well it will generalize to other datasets.

The Analysis node allows us to test the model against records for which we already know the actual result. The next stage illustrates how we can use the model to score records for which we don't know the outcome. For example, this might include people who are not currently customers of the bank, but who are prospective targets for a promotional mailing.

**Parent topic:** [Introduction to modeling](#)

## Scoring records

Earlier, we scored the same records used to estimate the model so we could evaluate how accurate the model was. Now we'll score a different set of records from the ones used to create the model. This is the goal of modeling with a target field: Study records for which you know the outcome, to identify patterns that will allow you to predict outcomes you don't yet know.

Figure 1. Attaching new data for scoring



You could update the data asset Import node to point to a different data file, or you could add a new Import node that reads in the data you want to score. Either way, the new dataset must contain the same input fields used by the model (Age, Income level, Education and so on), but not the target field Credit rating.

Alternatively, you could add the model nugget to any flow that includes the expected input fields. Whether read from a file or a database, the source type doesn't matter as long as the field names and types match those used by the model.

## Summary

---

This example Introduction to Modeling flow demonstrates the basic steps for creating, evaluating, and scoring a model.

- The modeling node estimates the model by studying records for which the outcome is known, and creates a model nugget. This is sometimes referred to as training the model.
- The model nugget can be added to any flow with the expected fields to score records. By scoring the records for which you already know the outcome (such as existing customers), you can evaluate how well it performs.
- Once you are satisfied that the model performs acceptably well, you can score new data (such as prospective customers) to predict how they will respond.
- The data used to train or estimate the model may be referred to as the analytical or historical data; the scoring data may also be referred to as the operational data.

Parent topic: [Introduction to modeling](#)

## Automated modeling for a flag target

---

With the Auto Classifier node, you can automatically create and compare a number of different models for either flag (such as whether or not a given customer is likely to default on a loan or respond to a particular offer) or nominal (set) targets.

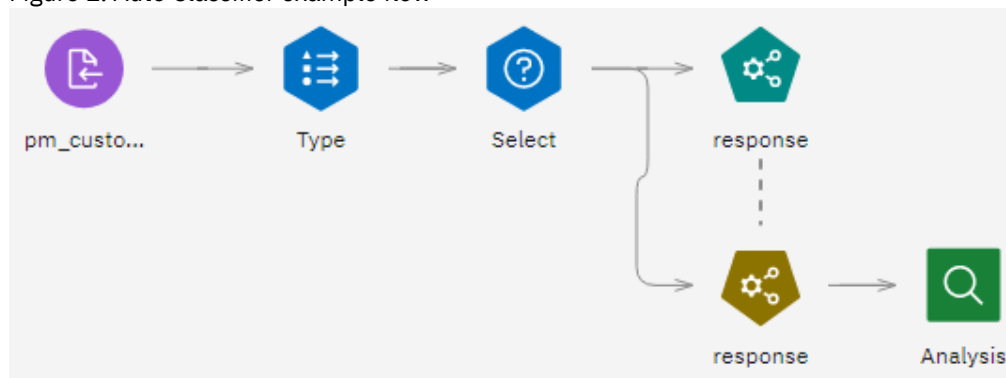
### Modeling customer response (Auto Classifier)

---

In this example, we'll search for a flag (yes or no) outcome. Within a relatively simple flow, the node generates and ranks a set of candidate models, chooses the ones that perform the best, and combines them into a single aggregated (Ensembled) model. This approach combines the ease of automation with the benefits of combining multiple models, which often yield more accurate predictions than can be gained from any one model.

This example is based on a fictional company that wants to achieve more profitable results by matching the right offer to each customer. This approach stresses the benefits of automation. For a similar example that uses a continuous (numeric range) target, see [Automated modeling for a continuous target](#).

Figure 1. Auto Classifier example flow



This example uses the flow named Automated Modeling for a Flag Target, available in the example project installed with the product. The data file is pm\_customer\_train1.csv.

Let's take a look at the flow.

1. On the My Projects screen, click Example Project.

2. Scroll down to the Modeler flows section, click View all, and select the Automated Modeling for a Flag Target flow.

- **Historical data**

This example uses the data file `pm_customer_train1.csv`, which contains historical data that tracks the offers made to specific customers in past campaigns, as indicated by the value of the `campaign` field. The largest number of records fall under the `Premium account` campaign.

- **Building the flow**
- **Generating and comparing models**
- **Summary**

Parent topic: [SPSS Modeler tutorials](#)

## Historical data

This example uses the data file `pm_customer_train1.csv`, which contains historical data that tracks the offers made to specific customers in past campaigns, as indicated by the value of the `campaign` field. The largest number of records fall under the `Premium account` campaign.

The values of the `campaign` field are actually coded as integers in the data (for example 2 = `Premium account`). Later, you'll define labels for these values that you can use to give more meaningful output.

Figure 1. Data about previous promotions

customer_id Type: String	campaign Type: String	response Type: String	response_date Type: String	purchase Type: String	purchase_date Type: String	product_id Type: String	Rowid Type: String
7	2	0		0			1
13	2	0		0			2
15	2	0		0			3
16	2	1	2006-07-05 00:00:00	0		183	761
23	2	0		0			4
24	2	0		0			5
30	2	0		0			6
30	3	0		0			7
33	2	0		0			8
42	3	0		0			9
42	2	0		0			10
52	2	0		0			11
57	2	0		0			12
63	2	1	2006-07-14 00:00:00	0		183	1501
74	2	0		0			13
74	3	0		0			14
75	2	0		0			15
82	2	0		0			16
89	3	0		0			17

The file also includes a `response` field that indicates whether the offer was accepted (0 = `no`, and 1 = `yes`). This will be the *target field*, or value, that you want to predict. A number of fields containing demographic and financial information about each customer are also included. These can be used to build or "train" a model that predicts response rates for individuals or groups based on characteristics such as income, age, or number of transactions per month.

## Building the flow

1. Add a Data Asset node that points to pm\_customer\_train1.csv.
2. Add a Type node, and select `response` as the target field (Role = Target). Set the measure for this field to Flag.

Figure 1. Setting the measurement level and role

<input type="checkbox"/>	Field	Measure		Role
<input type="checkbox"/>	# customer_	Continuous	▼	None
<input type="checkbox"/>	# campaign	Nominal	▼	Input
<input type="checkbox"/>	# response	Flag	▼	Target

3. Set the role to None for the following fields: `customer_id`, `campaign`, `response_date`, `purchase`, `purchase_date`, `product_id`, `Rowid`, and `X_random`. These fields will be ignored when you are building the model.
4. Click Read Values in the Type node to make sure that values are instantiated.

As we saw earlier, our source data includes information about four different campaigns, each targeted to a different type of customer account. These campaigns are coded as integers in the data, so to make it easier to remember which account type each integer represents, let's define labels for each one.

Figure 2. Choosing to specify values for a field

<input type="checkbox"/>	Field	Measure		Role		Value Mode	
<input type="checkbox"/>	# customer_	Continuous	▼	None	▼	Specify	▼
<input type="checkbox"/>	# campaign	Nominal	▼	Input	▼	Specify	▼
<input type="checkbox"/>	# response	Flag	▼	Target	▼	Read Pass Specify Extend Current	
<input type="checkbox"/>	Ⓜ response_	Continuous	▼	None	▼		

5. On the row for the campaign field, click the entry in the Value mode column.
6. Choose Specify from the drop-down.

Figure 3. Defining labels for the field values



### Values

1  
2  
3  
4

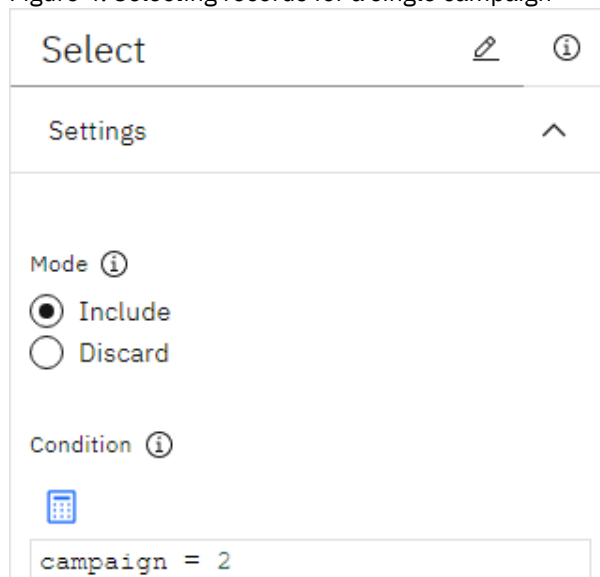
### Value Labels

Standard account  
Premium account  
Gold account  
Platinum account

7. Click the Edit icon in the far right column for the campaign field. Type the labels as shown for each of the four values.
8. Click OK. Now the labels will be displayed in output windows instead of the integers.
9. Attach a Table node to the Type node.
10. Right-click the Table node and select Run.
11. In the Outputs panel, double-click the table output to open it.
12. Click OK to close the output window.

Although the data includes information about four different campaigns, you will focus the analysis on one campaign at a time. Since the largest number of records fall under the Premium account campaign (coded `campaign=2` in the data), you can use a Select node to include only these records in the flow.

Figure 4. Selecting records for a single campaign



Select


Settings

Mode ⓘ

☒ Include

☐ Discard

Condition ⓘ



`campaign = 2`

**Parent topic:** [Automated modeling for a flag target](#)

## Generating and comparing models

1. Attach an Auto Classifier node, open its BUILD OPTIONS properties, and select Overall accuracy as the metric used to rank models.

2. Set the Number of models to use to 3. This means that the three best models will be built when you run the node.

Figure 1. Auto Classifier node, build options

response

Build Options

Model Name ⓘ

☒ Auto

☐ Custom

☒ Use partitioned data

☒ Cross-validate

Number of folds ⓘ

5

☐ Repeating Cross Validation partition assignment

Random seed [Generate](#)

1234567

☒ Build model for each split

Rank models by ⓘ

Overall accuracy

Rank models using ⓘ

☐ Training partition

☒ Test partition

Number of models to use ⓘ































3

Under the EXPERT options, you can choose from many different modeling algorithms.

3. Deselect the Discriminant and SVM model types. (These models take longer to train on this data, so deselecting them will speed up the example. If you don't mind waiting, feel free to leave them selected.)

Because you set Number of models to use to 3 under BUILD OPTIONS, the node will calculate the accuracy of the remaining algorithms and generate a single model nugget containing the three most accurate.

Figure 2. Auto Classifier node, expert options  
Select Models

<input type="checkbox"/>	Model Type	Settings	Number of Models
<input checked="" type="checkbox"/>	 C5	Default  	1
<input checked="" type="checkbox"/>	 Logistic regression	Default  	1
<input checked="" type="checkbox"/>	 Decision List	Default  	1
<input checked="" type="checkbox"/>	 Bayesian Network	Default  	1
<input type="checkbox"/>	 Discriminant	Default  	1
<input checked="" type="checkbox"/>	 KNN Algorithm	Default  	1
<input checked="" type="checkbox"/>	 LSVM	Default  	1
<input checked="" type="checkbox"/>	 Random Trees	Default  	1
<input type="checkbox"/>	 SVM	Default  	1
<input checked="" type="checkbox"/>	 Tree-AS	Default  	1

- Under the ENSEMBLE options, select Confidence-weighted voting for the ensemble method. This determines how a single aggregated score is produced for each record.

With simple voting, if two out of three models predict *yes*, then *yes* wins by a vote of 2 to 1. In the case of confidence-weighted voting, the votes are weighted based on the confidence value for each prediction. Thus, if one model predicts *no* with a higher confidence than the two *yes* predictions combined, then *no* wins.

Figure 3. Auto Classifier node, ensemble options

Ensemble

☒ Filter out fields generated by ensembled models

Ensemble method for Set Targets

Confidence-weighted voting

If voting is tied, select value using

☒ Random selection
☐ Highest confidence

Ensemble method for Flag Targets

Confidence-weighted voting

If voting is tied, select value using

☒ Random selection
☐ Highest confidence
☐ Raw propensity

- Run the flow. After a few minutes, the generated model nugget is built and placed on the canvas, and results are added to the Outputs panel. You can view the model nugget, or save or deploy it in a number of other ways.
- Right-click the model nugget and select View Model. You'll see details about each of the models created during the run. (In a real situation, in which hundreds of models may be created on a large dataset, this could take many hours.)

If you want to explore any of the individual models further, you can click their links in the Estimator column to drill down and browse the individual model results.

Figure 4. Auto Classifier results

## Auto Classifier - Models ⓘ

TARGET : RESPONSE

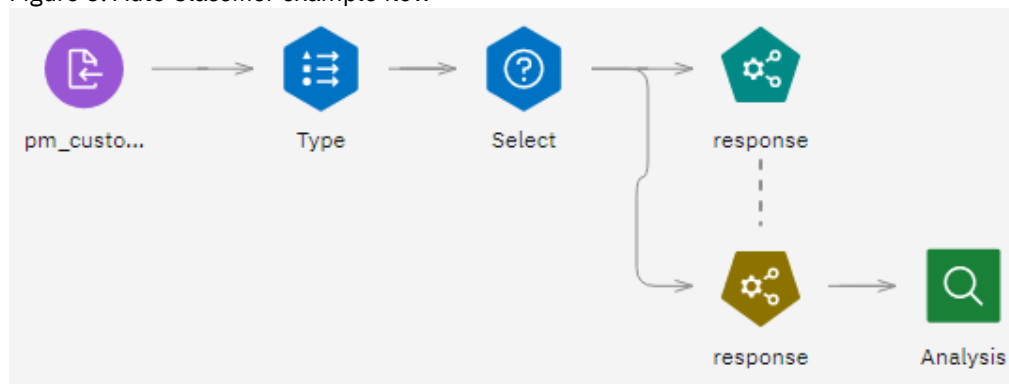
USE	ESTIMATOR	ACCURACY	BUILD TIME (MINS)	NO. FIELDS USED	ACTIONS
<input checked="" type="checkbox"/>	XGBoost Tree 1	93.009	2	23	
<input checked="" type="checkbox"/>	<a href="#">C5.0</a>	92.817	2	10	
<input checked="" type="checkbox"/>	<a href="#">C&amp;RT</a>	92.380	2	8	

By default, models are sorted based on overall accuracy, because this was the measure you selected in the Auto Classifier node properties. The XGBoost Tree model ranks best by this measure, but the C5.0 and C&RT models are nearly as accurate.

Based on these results, you decide to use all three of these most accurate models. By combining predictions from multiple models, limitations in individual models may be avoided, resulting in a higher overall accuracy.

7. In the USE column, select the three models. Return to the flow.
8. Attach an Analysis output node after the model nugget. Right-click the Analysis node and choose Run to run the flow.

Figure 5. Auto Classifier example flow



The aggregated score generated by the ensembled model is shown in a field named `$XF-response`. When measured against the training data, the predicted value matches the actual response (as recorded in the original `response` field) with an overall accuracy of 92.77%. While not quite as accurate as the best of the three individual models in this case (92.82% for C5.0), the difference is too small to be meaningful. In general terms, an ensembled model will typically be more likely to perform well when applied to datasets other than the training data.

Figure 6. Analysis of the three ensembled models

Results for output field response

Comparing `$XF-response` with response

Correct	12,527	92.77%
Wrong	977	7.23%
Total	13,504	

## Summary

With this example Automated Modeling for a Flag Target flow, you used the Auto Classifier node to compare a number of different models, used the three most accurate models, and added them to the flow within an ensembled Auto Classifier model nugget.

- Based on overall accuracy, the XGBoost Tree, C5.0, and C&R Tree models performed best on the training data.
- The ensembled model performed nearly as well as the best of the individual models and may perform better when applied to other datasets. If your goal is to automate the process as much as possible, this approach allows you to obtain a robust model under most circumstances without having to dig deeply into the specifics of any one model.

Parent topic: [Automated modeling for a flag target](#)

## Automated modeling for a continuous target

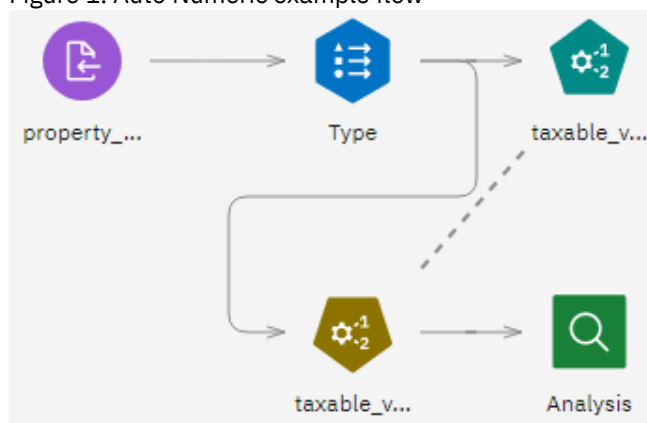
You can use the Auto Numeric node to automatically create and compare different models for continuous (numeric range) outcomes, such as predicting the taxable value of a property. With a single node, you can estimate and compare a set of candidate models and generate a subset of models for further analysis. The node works in the same manner as the Auto Classifier node, but for continuous rather than flag or nominal targets.

### Property values (Auto Numeric)

The node combines the best of the candidate models into a single aggregated (Ensembled) model nugget. This approach combines the ease of automation with the benefits of combining multiple models, which often yield more accurate predictions than can be gained from any one model.

This example focuses on a fictional municipality responsible for adjusting and assessing real estate taxes. To do this more accurately, they will build a model that predicts property values based on building type, neighborhood, size, and other known factors.

Figure 1. Auto Numeric example flow



This example uses the flow named Automated Modeling for a Continuous Target, available in the example project installed with the product. The data file is property\_values\_train.csv.

Let's take a look at the flow.

1. On the My Projects screen, click Example Project.
2. Scroll down to the Modeler flows section, click View all, and select the Automated Modeling for a Continuous Target flow.

- [Training data](#)
- [Building the flow](#)
- [Comparing the models](#)
- [Summary](#)

**Parent topic:** [SPSS Modeler tutorials](#)

## Training data

The data file includes a field named `taxable_value`, which is the *target field*, or value, that you want to predict. The other fields contain information such as neighborhood, building type, and interior volume, and may be used as predictors.

Field name	Label
<code>property_id</code>	Property ID
<code>neighborhood</code>	Area within the city
<code>building_type</code>	Type of building
<code>year_built</code>	Year built
<code>volume_interior</code>	Volume of interior
<code>volume_other</code>	Volume of garage and extra buildings
<code>lot_size</code>	Lot size
<code>taxable_value</code>	Taxable value

**Parent topic:** [Automated modeling for a continuous target](#)

## Building the flow

1. Add a Data Asset node that points to `property_values_train.csv`.
2. Add a Type node, and select `taxable_value` as the target field (Role = Target). Other fields will be used as predictors.

Figure 1. Setting the measurement level and role

Field	Measure ^	Role
# taxable_va	Continuous v	Target v

3. Attach an Auto Numeric node, and select Correlation as the metric used to rank models (under BASICS in the node properties).
4. Set the Number of models to use to 3. This means that the three best models will be built when you run the node.

Figure 2. Auto Numeric node BASICS

Rank models by ⓘ

- ☒ Correlation
- ☐ Number of fields
- ☐ Relative error

Rank models using ⓘ











- ☐ Training partition
- ☒ Test partition

Number of models to use ⓘ

5. Under EXPERT, leave the default settings in place. The node will estimate a single model for each algorithm, for a total of six models. (Alternatively, you can modify these settings to compare multiple variants for each model type.)

Because you set Number of models to use to 3 under BASICS, the node will calculate the accuracy of the six algorithms and build a single model nugget containing the three most accurate.

Figure 3. Auto Numeric node EXPERT options

<input type="checkbox"/>	Model Type	Settings	Number of Models
<input checked="" type="checkbox"/>	 Regression	Default	1
<input checked="" type="checkbox"/>	 Generalized Linear	Default	1
<input type="checkbox"/>	 Generalized linear engine	Default	1
<input type="checkbox"/>	 KNN Algorithm	Default	1
<input checked="" type="checkbox"/>	 Linear-AS	Default	1
<input type="checkbox"/>	 LSVM	Default	1
<input type="checkbox"/>	 Random Trees	Default	1
<input type="checkbox"/>	 SVM	Default	1
<input type="checkbox"/>	 Tree-AS	Default	1
<input type="checkbox"/>	 XGBoost Linear	Default	1

6. Under ENSEMBLE, leave the default settings in place. Since this is a continuous target, the ensemble score is generated by averaging the scores for the individual models.



Ensemble

☒ Filter out fields generated by ensembled models
   
☒ Calculate standard error

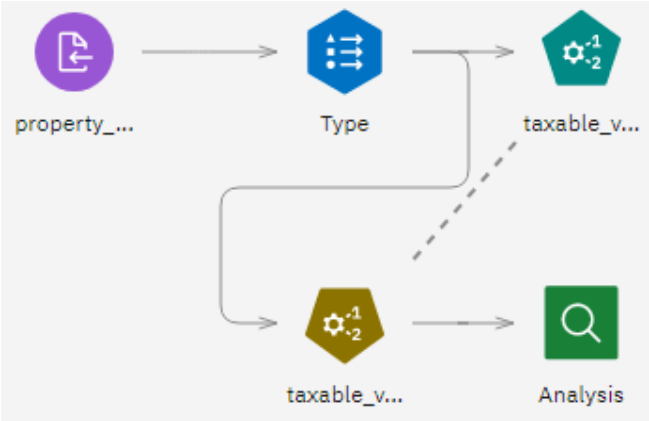
Parent topic:
 [Automated modeling for a continuous target](#)

## Comparing the models

1. Run the flow. A generated model nugget is built and placed on the canvas, and results are added to the Outputs panel. You can view the model nugget, or save or deploy it in a number of ways.

Right-click the model nugget and select View Model. You'll see details about each of the models created during the run. (In a real situation, in which hundreds of models are estimated on a large dataset, this could take many hours.)

Figure 1. Auto numeric example flow with model nugget



If you want to explore any of the individual models further, you can click a model name in the ESTIMATOR column to drill down and explore the individual model results.

Figure 2. Auto Numeric results  
 Auto Numeric - Models ⓘ  
 TARGET : TAXABLE\_VALUE

USE	ESTIMATOR	ACCURACY	RELATIVE ERROR	BUILD TIME (MINS)	NO. FIELDS USED	ACTIONS
<input checked="" type="checkbox"/>	<a href="#">MLP Neural Network</a>	0.933	0.130	< 1	7	
<input checked="" type="checkbox"/>	<a href="#">LE</a>	0.931	0.133	< 1	7	
<input checked="" type="checkbox"/>	<a href="#">GZLM</a>	0.915	0.162	< 1	7	

By default, models are sorted by accuracy (correlation) because correlation this was the measure you selected in the Auto Numeric node's properties. For purposes of ranking, the absolute value of the accuracy is used, with values closer to 1 indicating a stronger relationship.

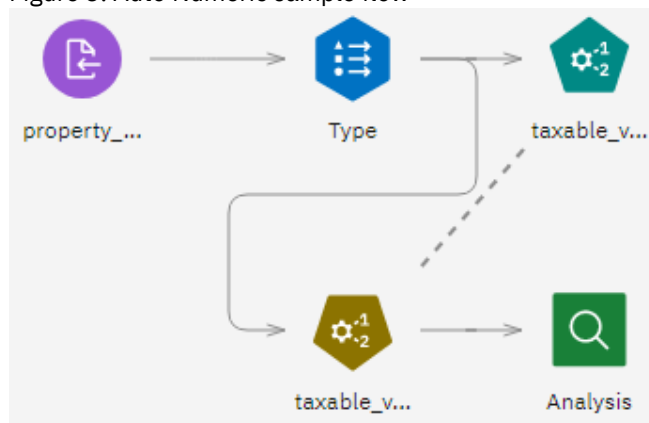
You can sort on a different column by clicking the header for that column.

Based on these results, you decide to use all three of these most accurate models. By combining predictions from multiple models, limitations in individual models may be avoided, resulting in a higher overall accuracy.

In the USE column, make sure all three models are selected.

Attach an Analysis node (from the Outputs palette) after the model nugget. Right-click the Analysis node and choose Run to run the flow again.

Figure 3. Auto Numeric sample flow



The averaged score generated by the ensembled model is added in a field named `$XR-taxable_value`, with a correlation of 0.934, which is higher than those of the three individual models. The ensemble scores also show a low mean absolute error and may perform better than any of the individual models when applied to other datasets.

Figure 4. Auto Numeric sample flow analysis results

Results for output field taxable_value	
Comparing \$XR-taxable_value with taxable_value	
Minimum Error	-144610.653
Maximum Error	135785.686
Mean Error	421.232
Mean Absolute Error	20353.87
Standard Deviation	28449.888
Linear Correlation	0.934
Occurrences	1,138

**Parent topic:** [Automated modeling for a continuous target](#)

## Summary

With this example Automated Modeling for a Flag Target flow, you used the Auto Numeric node to compare a number of different models, selected the three most accurate models, and added them to the flow within an ensembled Auto Numeric model nugget.

The ensembled model showed performance that was better than two of the individual models and may perform better when applied to other datasets. If your goal is to automate the process as much as possible, this approach allows you to obtain a robust model under most circumstances without having to dig deeply into the specifics of any one model.

## Automated data preparation

Preparing data for analysis is one of the most important steps in any data-mining project - and traditionally, one of the most time consuming. The Auto Data Prep node handles the task for you, analyzing your data and identifying fixes, screening out fields that are problematic or not likely to be useful, deriving new attributes when appropriate, and improving performance through intelligent screening techniques.

You can use the Auto Data Prep node in fully automated fashion, allowing the node to choose and apply fixes, or you can preview the changes before they're made and accept or reject them as desired. With this node, you can ready your data for data mining quickly and easily, without the need for prior knowledge of the statistical concepts involved. If you run the node with the default settings, models will tend to build and score more quickly.

This example uses the flow named Automated Data Preparation, available in the example project installed with the product. The data file is telco.csv. This example demonstrates the increased accuracy you can find by using the default Auto Data Prep node settings when building models.

Let's take a look at the flow.

1. On the My Projects screen, click Example Project.
2. Scroll down to the Modeler flows section, click View all, and select the Automated Data Preparation flow.

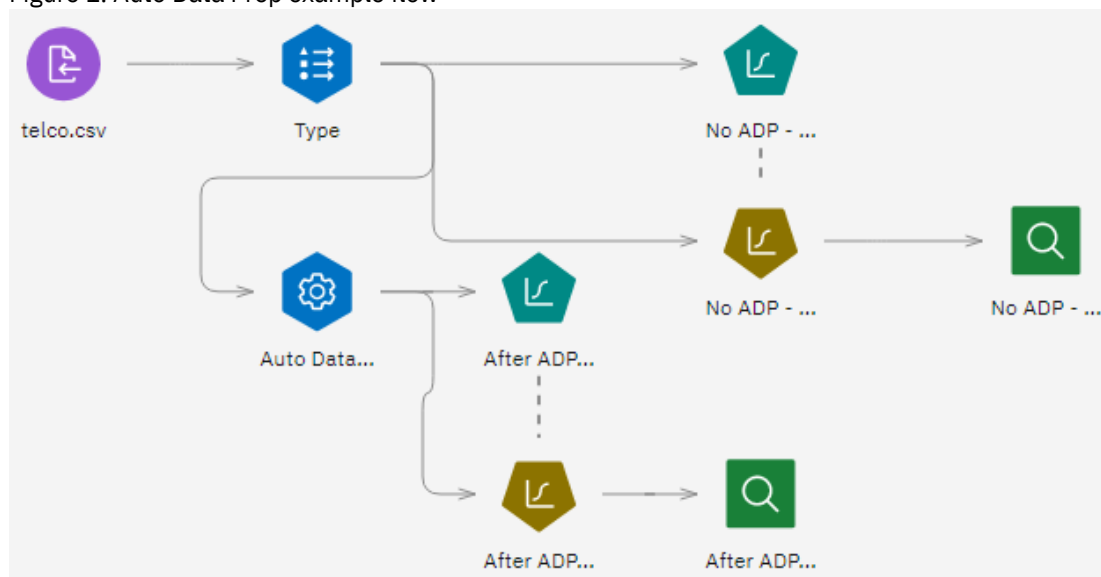
- [Building the flow](#)
- [Comparing the models](#)

Parent topic: [SPSS Modeler tutorials](#)

## Building the flow

1. Add a Data Asset node that points to telco.csv.

Figure 1. Auto Data Prep example flow



2. Attach a Type node to the Data Asset node. Set the measure for the `churn` field to Flag, and set the role to Target. Make sure the role for all other fields is set to Input.

Figure 2. Setting the measurement level and role

<input type="checkbox"/>	Field ^	Measure		Role	
<input type="checkbox"/>	# churn	Flag	▼	Target	▼
<input type="checkbox"/>	# confer	Continuous	▼	Input	▼
<input type="checkbox"/>	# custcat	Continuous	▼	Input	▼
<input type="checkbox"/>	# ebill	Continuous	▼	Input	▼
<input type="checkbox"/>	# ed	Continuous	▼	Input	▼
<input type="checkbox"/>	# employ	Continuous	▼	Input	▼
<input type="checkbox"/>	# equip	Continuous	▼	Input	▼
<input type="checkbox"/>	# equipmon	Continuous	▼	Input	▼

3. Attach a Logistic node to the Type node.
4. In the Logistic node's properties, under MODEL SETTINGS, select the Binomial procedure. For Model Name, select Custom and enter No ADP - churn.

Figure 3. Choosing model options

Model Name ⓘ

☐ Auto

☒ Custom

☐ Use partitioned data

☒ Build model for each split

Procedure ⓘ

☐ Multinomial

☒ Binomial

5. Attach an Auto Data Prep node to the Type node. Under OBJECTIVES, leave the default settings in place to analyze and prepare your data by balancing both speed and accuracy.
6. Run the flow to analyze and process your data. Other Auto Data Prep node properties allow you to specify that you want to concentrate more on accuracy, more on the speed of processing, or to fine tune many of the data preparation processing steps.

Note: If you want to adjust the node properties and run the flow again in the future, since the model already exists, you must first click Clear Analysis, under OBJECTIVES before running the flow again.

Figure 4. Auto Data Prep default objectives

Objectives ^

ⓘ *If any option except Custom Analysis is chosen then most settings for the manual configuration of ADP will be disabled.*

What is your objective? ⓘ

☒ Balance speed and accuracy

☐ Optimize for speed

☐ Optimize for accuracy

☐ Custom analysis

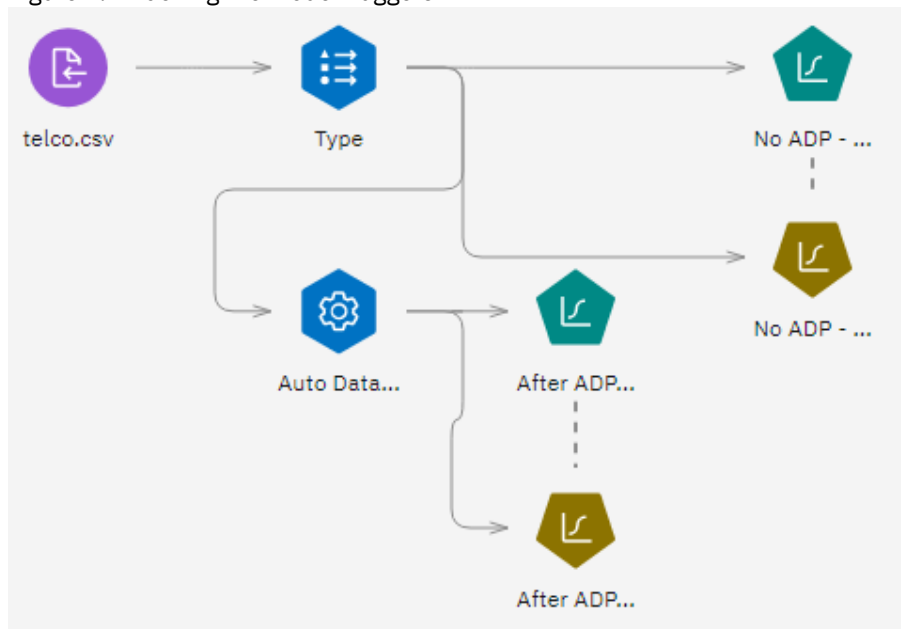
- Attach a Logistic node to the Auto Data Prep node.
- In the Logistic node's properties, under MODEL SETTINGS, select the Binomial procedure. For Model Name, select Custom and enter `After ADP - churn`.

**Parent topic:** [Automated data preparation](#)

## Comparing the models

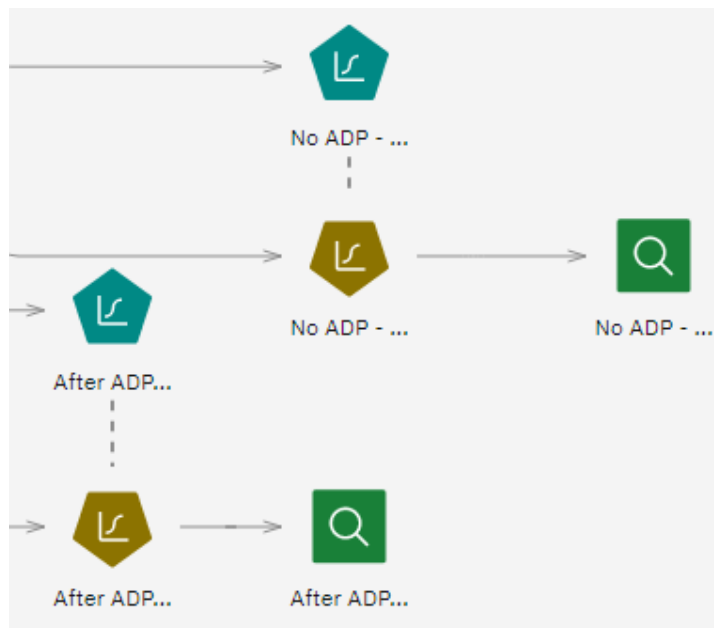
- Right-click each Logistic node and run it to create the model nuggets, which are added to the flow. Results are also added to the Outputs panel.

Figure 1. Attaching the model nuggets



- Attach Analysis nodes to the model nuggets and run the Analysis nodes (using their default settings).

Figure 2. Attaching the Analysis nodes



The Analysis of the non Auto Data Prep-derived model shows that just running the data through the Logistic Regression node with its default settings gives a model with low accuracy - just 10.6%.

Figure 3. Non ADP-derived model results

Results for output field churn

Comparing SL-churn with churn

Correct	106	10.6%
Wrong	894	89.4%
Total	1,000	

The Analysis of the Auto-Data Prep-derived model shows that by running the data through the default Auto Data Prep settings, you have built a much more accurate model that's 78.3% correct.

Figure 4. ADP-derived model results

Results for output field churn

Comparing SL-churn with churn

Correct	783	78.3%
Wrong	217	21.7%
Total	1,000	

In summary, by just running the Auto Data Prep node to fine tune the processing of your data, you were able to build a more accurate model with little direct data manipulation.

Obviously, if you're interested in proving or disproving a certain theory, or want to build specific models, you may find it beneficial to work directly with the model settings. However, for those with a reduced amount of time, or with a large amount of data to prepare, the Auto Data Prep node may give you an advantage.

Note that the results in this example are based on the training data only. To assess how well models generalize to other data in the real world, you would use a Partition node to hold out a subset of records for purposes of testing and validation.

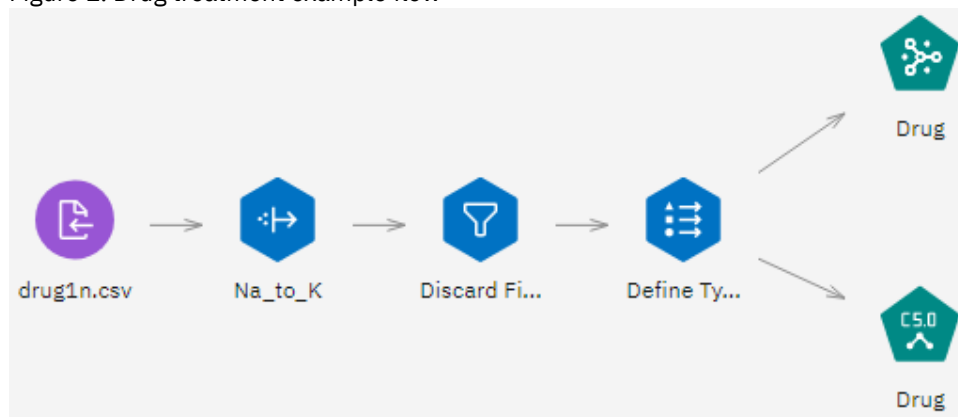
**Parent topic:** [Automated data preparation](#)

## Drug treatment - exploratory graphs

In this example, imagine you're a medical researcher compiling data for a study. You've collected data about a set of patients, all of whom suffered from the same illness. During their course of treatment, each patient responded to one of five medications. Part of your job is to use data mining to find out which drug might be appropriate for a future patient with the same illness.

This example uses the flow named Drug Treatment - Exploratory Graphs, available in the example project installed with the product. The data file is drug1n.csv.

Figure 1. Drug treatment example flow



The data fields used in this example are:

Data field	Description
Age	Age of patient (number)
Sex	M or F
BP	Blood pressure: HIGH, NORMAL, or LOW
Cholesterol	Blood cholesterol: NORMAL or HIGH
Na	Blood sodium concentration
K	Blood potassium concentration
Drug	Prescription drug to which a patient responded

- [Reading in text data](#)
- [Creating a distribution chart](#)
- [Creating a scatterplot](#)
- [Creating a web chart](#)
- [Creating advanced visualizations](#)
- [Deriving a new field](#)
- [Building a model](#)
- [Browsing the model](#)
- [Using an Analysis node](#)

**Parent topic:** [SPSS Modeler tutorials](#)

## Reading in text data

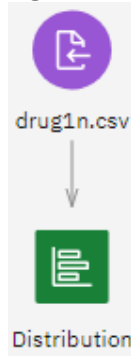
1. You can read in delimited text data using a Data Asset import node. From the Palette, under Import, add a Data Asset node to your flow.
2. Double-click the node to display its properties and select the data file drug1n.csv.
3. Now that you've added the data file, you may want to glance at the values for some of the records. One way to do this is by building a flow that includes a Table node. An easier way is to simply right-click the Data Asset node you just added and select Preview.

**Parent topic:** [Drug treatment - exploratory graphs](#)

## Creating a distribution chart

During data mining, it is often useful to explore the data by creating visual summaries. Watson Studio offers many different types of charts to choose from, depending on the kind of data you want to summarize. For example, to find out what proportion of the patients responded to each drug, use a Distribution node.

Figure 1. Distribution node



1. Under Graphs on the Palette, add a Distribution node to the flow and connect it to the drug1n.csv Data Asset node. Then double-click the node to edit its options.
2. Select Drug as the target field whose distribution you want to show. Then click Save, right-click the Distribution node, and select Run. A distribution chart is added to the Outputs panel.

The chart helps you see the "shape" of the data. It shows that patients responded to drug Y most often and to drugs B and C least often.

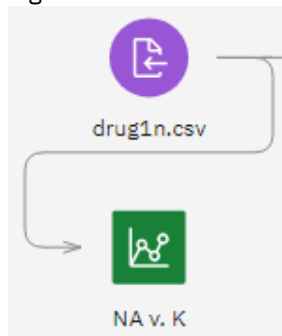
Alternatively, you can attach and run a Data Audit node for a quick glance at distributions and histograms for all fields at once. The Data Audit node is available under Outputs on the Palette.

**Parent topic:** [Drug treatment - exploratory graphs](#)

## Creating a scatterplot

Now let's take a look at what factors might influence Drug, the target variable. As a researcher, you know that the concentrations of sodium and potassium in the blood are important factors. Since these are both numeric values, you can create a scatterplot of sodium versus potassium, using the drug categories as a color overlay.

Figure 1. Plot node

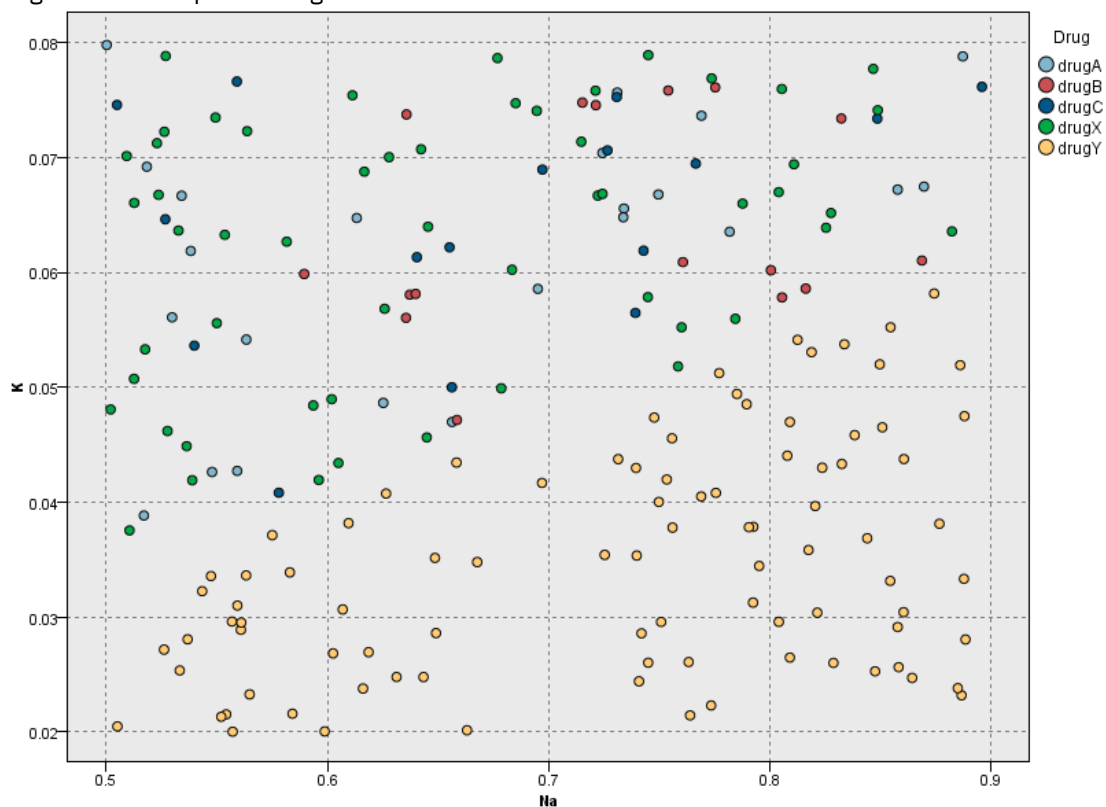


1. Place a Plot node on the canvas and connect it to the drug1n.csv Data Asset node. Then double-click the Plot node to edit its properties.
2. Select Na as the X field, K as the Y field, and Drug as the Color (overlay) field. Click Save, then right-click the Plot node and select Run. A plot chart is added to the Outputs pane.

The plot clearly shows a threshold above which the correct drug is always drug Y and below which the correct drug is never drug Y. This threshold is a ratio -- the ratio of sodium (Na) to potassium (K).



Figure 2. Scatterplot of drug distribution

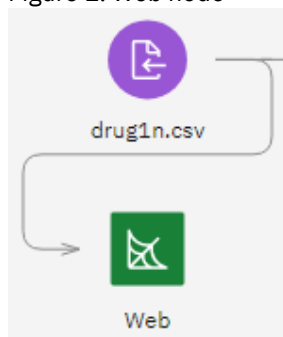


**Parent topic:** [Drug treatment - exploratory graphs](#)

## Creating a web chart

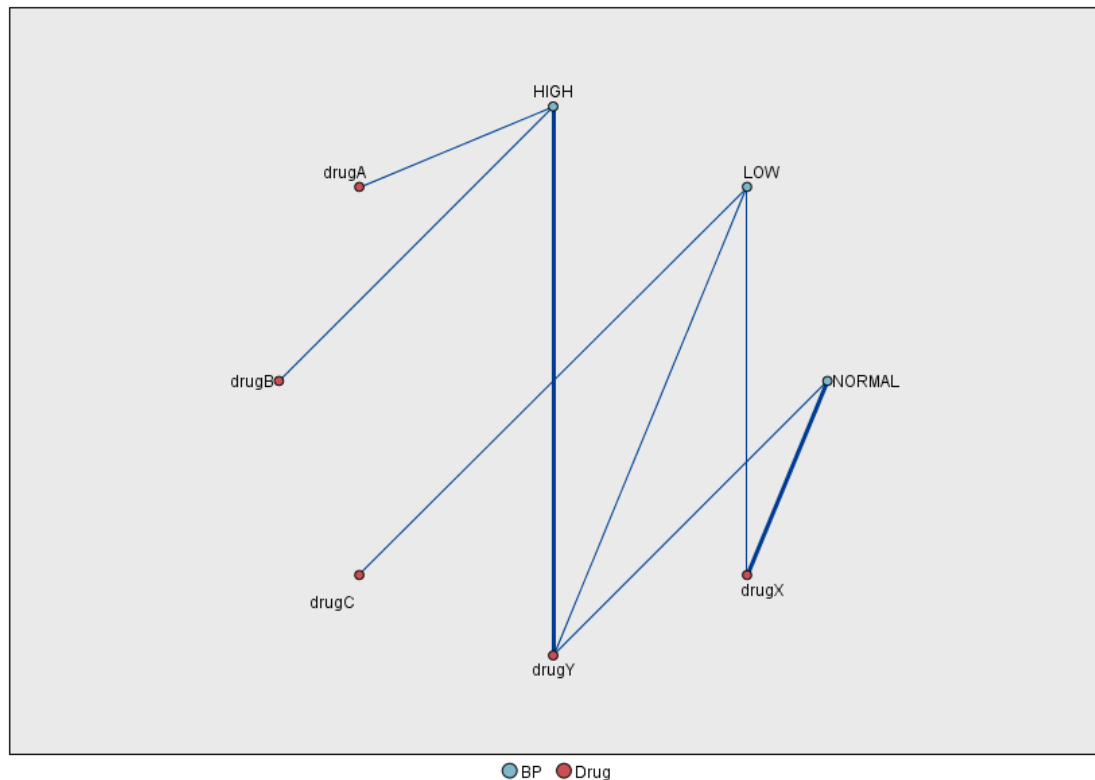
Since many of the data fields are categorical, you can also try plotting a web chart, which maps associations between different categories.

Figure 1. Web node



1. Place a Web node on the canvas and connect it to the drug1n.csv Data Asset node. Then double-click the Web node to edit its properties.
2. Select the fields BP (for blood pressure) and Drug. Click Save, then right-click the Web node and select Run. A web chart is added to the Outputs pane.

Figure 2. Web graph of drugs vs. blood pressure



From the plot, it appears that drug Y is associated with all three levels of blood pressure. This is no surprise; you have already determined the situation in which drug Y is best.

But if you ignore drug Y and focus on the other drugs, you can see that drugs A and B are also associated with high blood pressure. And drugs C and X are associated with low blood pressure. And normal blood pressure is associated with drug X. At this point, though, you still don't know how to choose between drugs A and B or between drugs C and X, for a given patient. This is where modeling can help.

**Parent topic:** [Drug treatment - exploratory graphs](#)

## Creating advanced visualizations

The previous three sections use different types of graph nodes. Another way to explore data is with the advanced visualizations feature.

You can use the Charts node to launch the chart builder and create advanced charts to explore your data from different perspectives and identify patterns, connections, and relationships within your data.

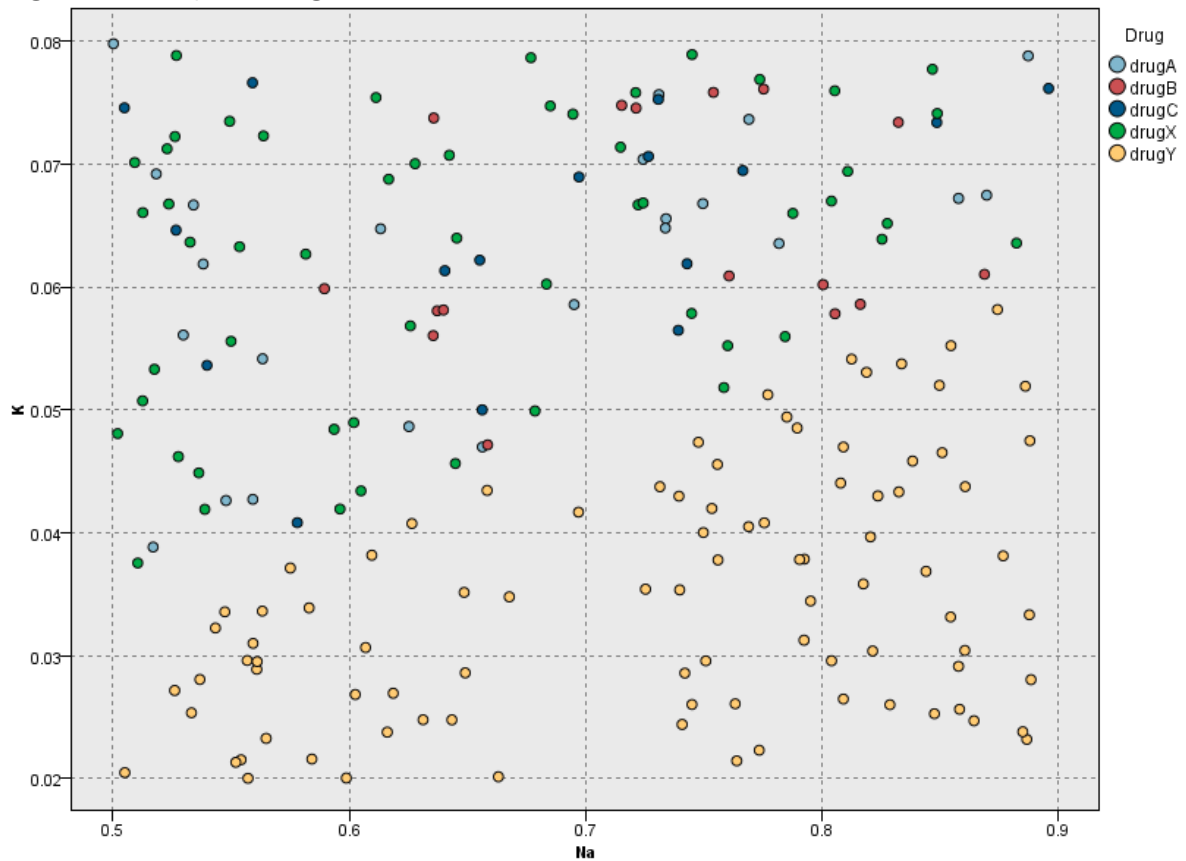
Figure 1. Advanced visualizations



Parent topic: [Drug treatment - exploratory graphs](#)

## Deriving a new field

Figure 1. Scatterplot of drug distribution



Since the ratio of sodium to potassium seems to predict when to use drug Y, you can derive a field that contains the value of this ratio for each record. This field might be useful later when you build a model to predict when to use each of the five drugs.

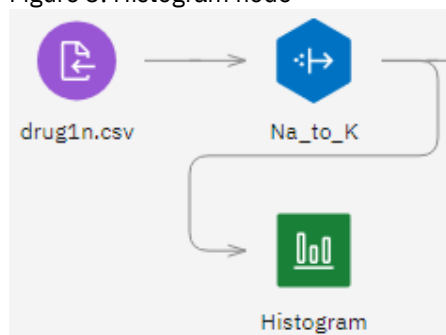
1. To simplify your flow layout, start by deleting all the nodes except the drug1n.csv Data Asset node.
2. Place a Derive node on the canvas and connect it to the drug1n.csv Data Asset node.

Figure 2. Derive node



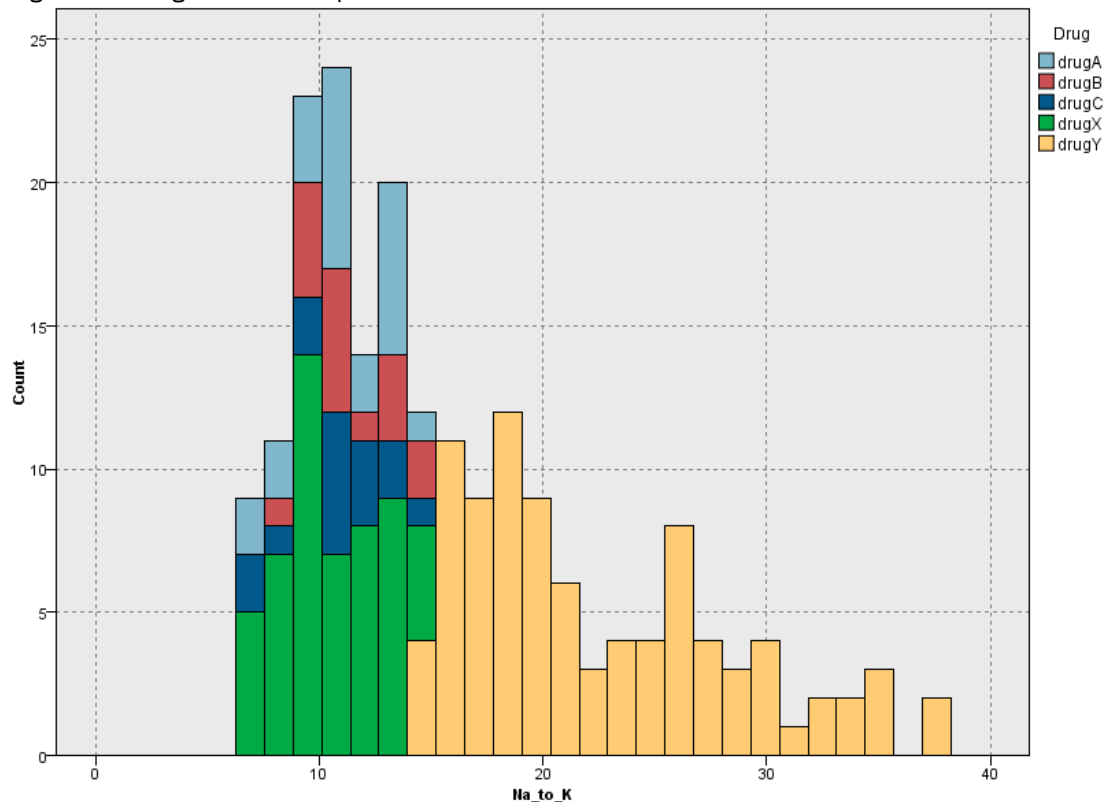
3. Double-click the Derive node to edit its properties.
4. Name the new field `Na_to_K`. Since you obtain the new field by dividing the sodium value by the potassium value, enter `Na/K` for the expression. You can also create an expression by clicking the calculator icon. This opens the Expression Builder, a way to interactively create expressions using built-in lists of functions, operands, and fields and their values.
5. You can check the distribution of your new field by attaching a Histogram node to the Derive node. In the Histogram node properties, specify `Na_to_K` as the field to be plotted and `Drug` as the color overlay field.

Figure 3. Histogram node



6. Right-click the Histogram node and select Run. A histogram chart is added to the Outputs pane. Based on the chart, you can conclude that when the `Na_to_K` value is about 15 or above, drug Y is the drug of choice.

Figure 4. Histogram chart output



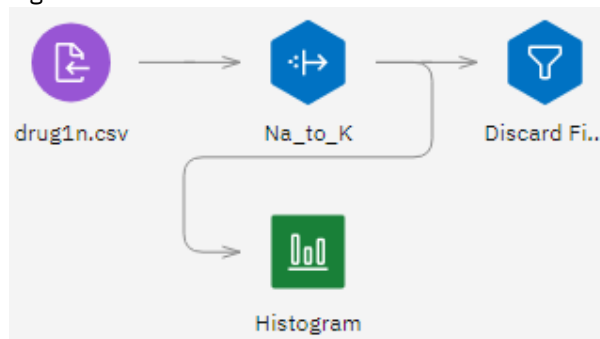
## Building a model

By exploring and manipulating the data, you have been able to form some hypotheses. The ratio of sodium to potassium in the blood seems to affect the choice of drug, as does blood pressure. But you cannot fully explain all of the relationships yet. This is where modeling will likely provide some answers. In this case, you will try to fit the data using a rule-building model called C5.0.

Since you're using a derived field, `Na_to_K`, you can filter out the original fields, `Na` and `K`, so they're not used twice in the modeling algorithm. You can do this by using a Filter node.

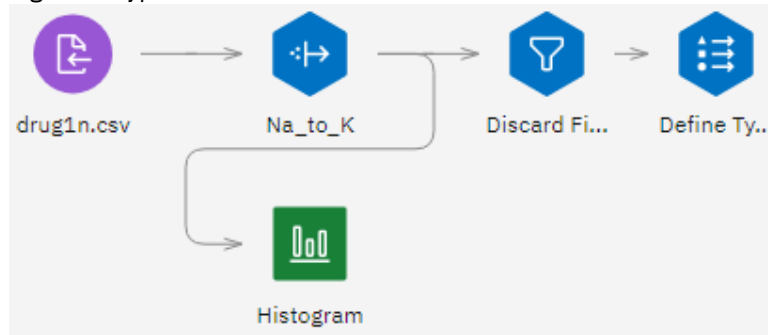
1. Place a Filter node on the canvas and connect it to the Derive node.

Figure 1. Filter node



2. Double-click the Filter node to edit its properties. Name it Discard Fields.
3. For Mode, make sure Filter the selected fields is selected. Then select the `K` and `Na` fields. Click Save.
4. Place a Type node on the canvas and connect it to the Filter node. With the Type node, you can indicate the types of fields you're using and how they're used to predict the outcomes.

Figure 2. Type node



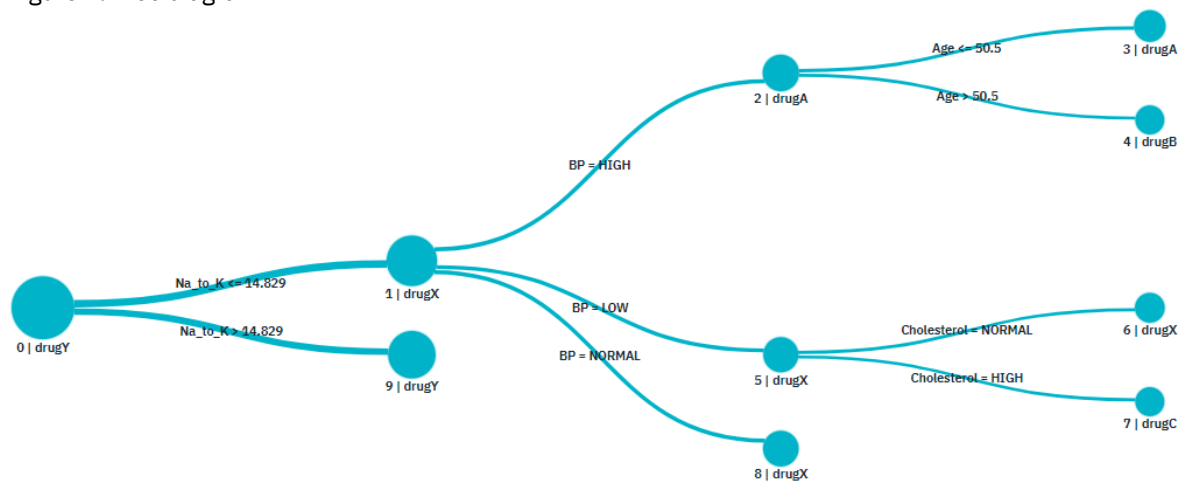
5. Double-click the Type node to edit its properties. Name it Define Types.
6. Set the role for the `Drug` field to Target, indicating that `Drug` is the field you want to predict. Leave the role for the other fields set to Input so they'll be used as predictors. Click Save.
7. To estimate the model, place a C5.0 node on the canvas and attach it to the end of the flow. Then click the Run button on the toolbar to run the flow.

## Browsing the model

When the C5.0 node runs, its model nugget is added to the flow. To browse the model, right-click the model nugget and choose View Model.

The Tree Diagram displays the set of rules generated by the C5.0 node in a tree format. Now you can see the missing pieces of the puzzle. For people with an Na-to-K ratio less than 14.829 and high blood pressure, age determines the choice of drug. For people with low blood pressure, cholesterol level seems to be the best predictor.

Figure 1. Tree diagram



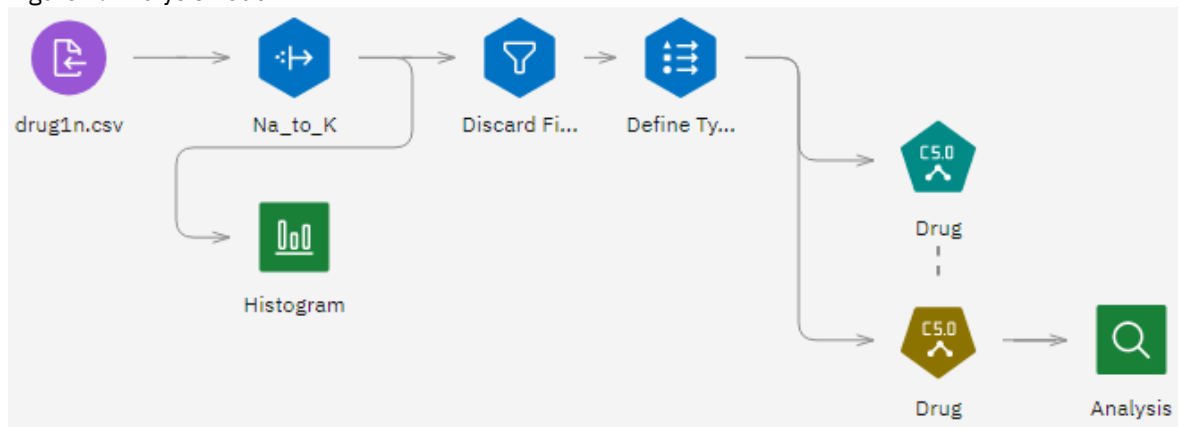
You can hover over the nodes in the tree to see more details such as the number of cases for each blood pressure category and the confidence percentage of cases.

Parent topic: [Drug treatment - exploratory graphs](#)

## Using an Analysis node

You can assess the accuracy of the model using an Analysis node. From the Palette, under Outputs, place an Analysis node on the canvas and attach it to the C5.0 model nugget. Then right-click the Analysis node and select Run.

Figure 1. Analysis node



The Analysis node output shows that with this artificial dataset, the model correctly predicted the choice of drug for every record in the dataset. With a real dataset you are unlikely to see 100% accuracy, but you can use the Analysis node to help determine whether the model is acceptably accurate for your particular application.

Figure 2. Analysis node output

Results for output field Drug

Comparing \$C-Drug with Drug

Correct	200	100%
Wrong	0	0%
Total	200	

## Screening predictors

The Feature Selection node helps you identify the fields that are most important in predicting a certain outcome. From a set of hundreds or even thousands of predictors, the Feature Selection node screens, ranks, and selects the predictors that may be most important. Ultimately, you may end up with a quicker, more efficient model - one that uses fewer predictors, runs more quickly, and may be easier to understand.

The data used in this example represents a data warehouse for a hypothetical telephone company and contains information about responses to a special promotion by 5,000 of the company's customers. The data includes many fields that contain customers' age, employment, income, and telephone usage statistics. Three "target" fields show whether or not the customer responded to each of three offers. The company wants to use this data to help predict which customers are most likely to respond to similar offers in the future.

This example uses the flow named Screening Predictors, available in the example project installed with the product. The data file is customer\_dbase.csv.

This example focuses on only one of the offers as a target. It uses the CHAID tree-building node to develop a model to describe which customers are most likely to respond to the promotion. It contrasts two approaches:

- Without feature selection. All predictor fields in the dataset are used as inputs to the CHAID tree.
- With feature selection. The Feature Selection node is used to select the top 10 predictors. These are then input into the CHAID tree.

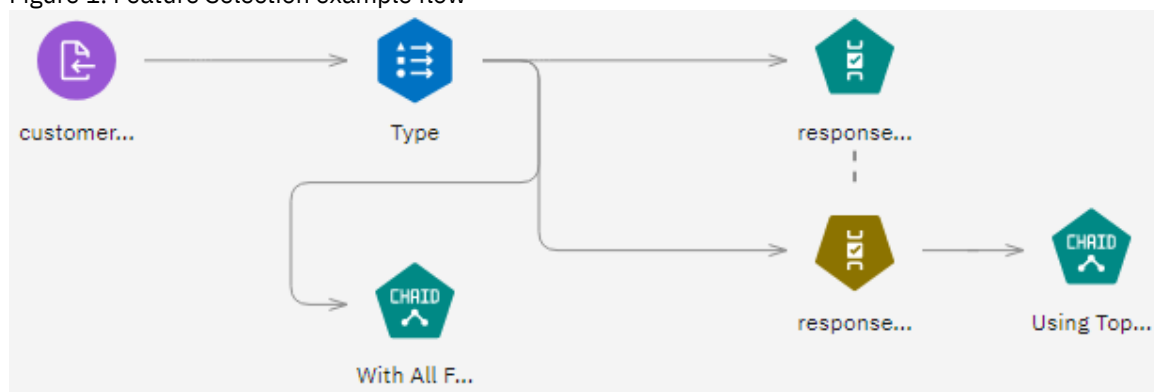
By comparing the two resulting tree models, we can see how feature selection can produce effective results.

- [Building the flow](#)
- [Building the models](#)

Parent topic: [SPSS Modeler tutorials](#)

## Building the flow

Figure 1. Feature Selection example flow



1. Add a Data Asset node that points to customer\_dbase.csv.
2. Add a Type node after the Data Asset node.
3. Double-click the Type node to open its properties, and change the role for response\_01 to Target. Change the role to None for the other response fields (response\_02 and response\_03) and for the customer ID (custid) field. Leave the role set to Input for all other fields.

Figure 2. Adding a Type node

Search in column Field

<input type="checkbox"/>	Field ^	Measure		Role	
<input type="checkbox"/>	# response_	Continuous	▼	Target	▼
<input type="checkbox"/>	# response_	Continuous	▼	None	▼
<input type="checkbox"/>	# response_	Continuous	▼	None	▼
<input type="checkbox"/>	# retire	Continuous	▼	Input	▼

- Click Read Values and then click Save.
- Add a Feature Selection modeling node after the Type node. In the node properties, the rules and criteria used for screening or disqualifying fields are defined.

Figure 3. Adding a Feature Selection node

☒ Use partitioned data

☒ Screen max. percentage of missing values
 

70

☒ Screen max. percentage with single category
 

90

☒ Screen max. percentage of a single category
 

95

☒ Screen min. coefficient of variation
 

0.1

☒ Screen min. standard deviation
 

0

- Run the flow to generate the Feature Selection model nugget.
- To look at the results, right-click the model nugget and choose View Model. The results show the fields found to be useful in the prediction, ranked by importance. By examining these fields, you can decide which ones to use in subsequent modeling sessions.
- To compare results without feature selection, you must add two CHAID modeling nodes to the flow: one that uses feature selection and one that doesn't. Add two CHAID nodes, one connected to the Type node and the other connected to the Feature Selection model nugget, as shown in the example flow at the beginning of this section.



9. Double-click each CHAID node to open its properties. Under Objectives, make sure that Build new model and Create a standard model are selected. Under Basic, Maximum Tree Depth, select Custom and set it to 5.

**Parent topic:** [Screening predictors](#)

## Building the models

---

1. Run the CHAID node that uses all the predictors in the dataset (the one connected to the Type node). As it runs, notice how long it takes to finish.
2. Right-click the generated model nugget, select View Model, and look at the tree diagram.
3. Now run the other CHAID model, which uses less predictors. Again, look at its tree diagram.

It might be hard to tell, but the second model ran faster than the first one. Because this dataset is relatively small, the difference in run times is probably only a few seconds; but for larger real-world datasets, the difference might be very noticeable - minutes or even hours. Using feature selection may speed up your processing times dramatically.

The second tree also contains fewer tree nodes than the first. It's easier to comprehend. Using fewer predictors is less expensive. It means that you have less data to collect, process, and feed into your models. Computing time is improved. In this example, even with the extra feature selection step, model building was faster with the smaller set of predictors. With a larger real-world dataset, the time savings should be greatly amplified.

Using fewer predictors results in simpler scoring. For example, you might identify only four profiles of customers who are likely to respond to the promotion. Note that with larger numbers of predictors, you run the risk of overfitting your model. The simpler model may generalize better to other datasets (although you would need to test this to be sure).

You could instead use a tree-building algorithm to do the feature selection work, allowing the tree to identify the most important predictors for you. In fact, the CHAID algorithm is often used for this purpose, and it's even possible to grow the tree level-by-level to control its depth and complexity. However, the Feature Selection node is faster and easier to use. It ranks all of the predictors in one fast step, allowing you to identify the most important fields quickly.

**Parent topic:** [Screening predictors](#)

## Reducing input data string length

---

For binomial logistic regression, and auto classifier models that include a binomial logistic regression model, string fields are limited to a maximum of eight characters. Where strings are more than eight characters, you can recode them using a Reclassify node.

This example uses the flow named Reducing Input Data String Length, available in the example project installed with the product. The data file is drug\_long\_name.csv.

This example focuses on a small part of a flow to show the type of errors that may be generated with overlong strings, and explains how to use the Reclassify node to change the string details to an acceptable length. Although the example uses a binomial Logistic Regression node, it is equally applicable when using the Auto Classifier node to generate a binomial Logistic Regression model.

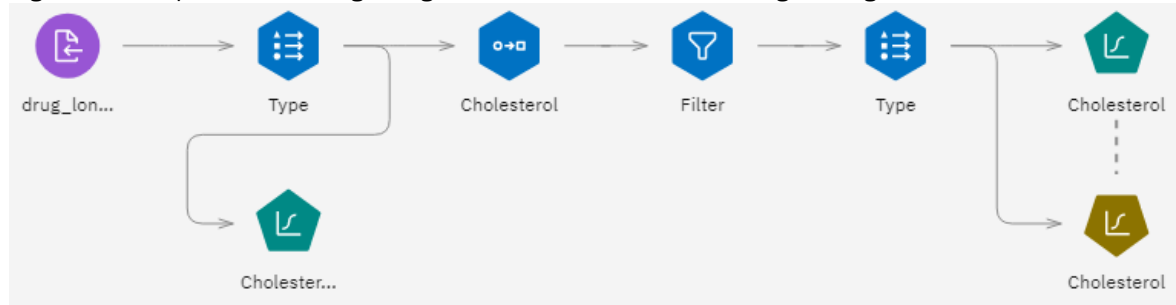
- [Reclassifying the data](#)

**Parent topic:** [SPSS Modeler tutorials](#)

## Reclassifying the data

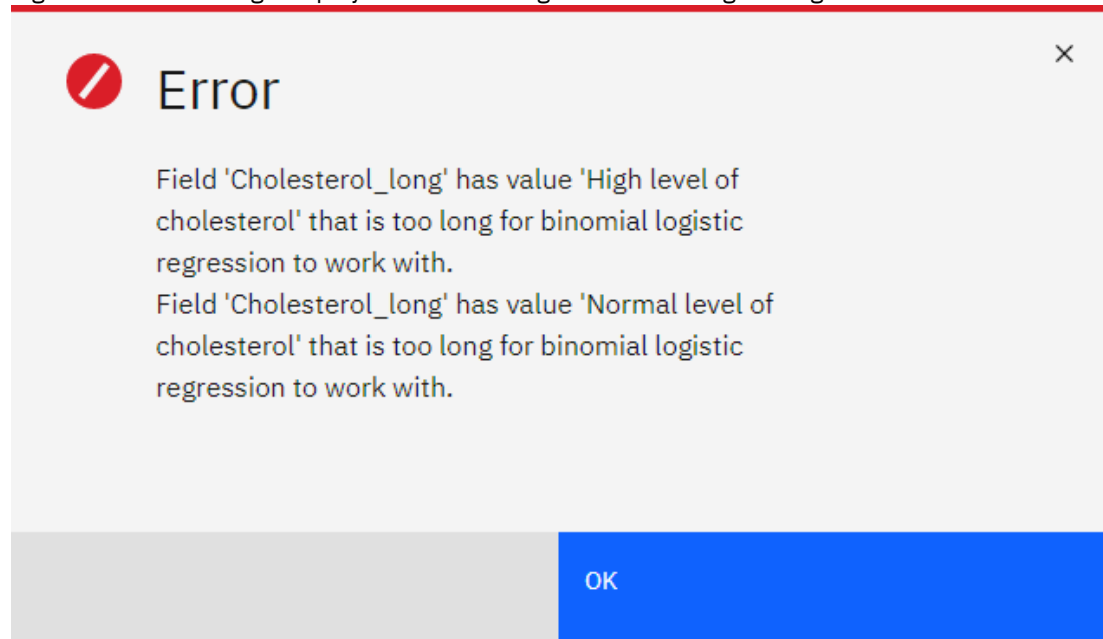
---

Figure 1. Example flow showing string reclassification for binomial logistic regression



1. Add a Data Asset node that points to drug\_long\_name.csv.
2. Add a Type node after the Data Asset node. Double-click the Type node to open its properties, and select Cholesterol\_long as the target.
3. Add a Logistic Regression node after the Type node. Double-click the node and select the Binomial procedure (instead of the default Multinomial procedure).
4. Right-click the Logistic Regression node and run it. An error message warns you that the Cholesterol\_long string values are too long. When you encounter this type of message, follow the procedure described in the rest of this example to modify your data.

Figure 2. Error message displayed when running the binomial logistic regression node



5. Add a Reclassify node after the Type node and double-click it to open its properties.
6. For the Reclassify Field, select Cholesterol\_long and type Cholesterol for the new field name.
7. Click Get values to add the Cholesterol\_long values to the original value column.
8. In the new value column, type High next to the original value of High level of cholesterol and Normal next to the original value of Normal level of cholesterol.

Figure 3. Reclassifying long strings

Mode ⓘ

☒ Single

☐ Multiple

Reclassify Into ⓘ

☒ New field

☐ Existing field

Reclassify Field ⓘ

Cholesterol\_long

New Field Name ⓘ

Cholesterol

[Get values](#) [Copy](#) [Clear new](#)

Automatically Reclassify

Values ⓘ

<input type="checkbox"/>	ORIGINAL VALUE	NEW VALUE
<input type="checkbox"/>	High level of cholesterol	High
<input type="checkbox"/>	Normal level of cholesterol	Normal

9. Add a Filter node after the Reclassify node. Double-click the node, choose Filter the selected fields, and select the Cholesterol\_long field.

Figure 4. Filtering the "Cholesterol\_long" field from the data

Filter ^

Mode ⓘ

☒ Filter the selected fields
 ☐ Retain the selected fields (all other fields are filtered)

Filter Options ▾

Select Fields ⓘ

⊖ Add Columns ⊕

☒ Field Name

☒ Cholesterol\_long

10. Add a Type node after the Filter node. Double-click the node and select `Cholesterol` as the target.

Figure 5. Short string details in the "Cholesterol" field

Field	Measure		Role		Value Mode		Values
# Age	Continuous	▾	Input	▾	Specify	▾	15, 74
abc Sex	Flag	▾	Input	▾	Specify	▾	F, M
abc BP	Nominal	▾	Input	▾	Specify	▾	HIGH, LOW, NORM...
# Na	Continuous	▾	Input	▾	Specify	▾	0.500517, 0.899774
# K	Continuous	▾	Input	▾	Specify	▾	0.020152, 0.079925
abc Drug	Nominal	▾	Input	▾	Specify	▾	drugA, drugB, drug...
abc Cholesterc	Flag	▾	Target	▾	Specify	▾	High, Normal

11. Add a Logistic node after the Type node. Double-click the node and select the Binomial procedure.

You can now run the binomial Logistic node and generate a model without encountering the error as you did before.

This example only shows part of a flow. For more information about the types of flows in which you might need to reclassify long strings, see the following example:

- Auto Classifier node. See [Automated modeling for a flag target](#).

**Parent topic:** [Reducing input data string length](#)

## Classifying telecommunications customers

Logistic regression is a statistical technique for classifying records based on values of input fields. It is analogous to linear regression, but takes a categorical target field instead of a numeric one.

For example, suppose a telecommunications provider has segmented its customer base by service usage patterns, categorizing the customers into four groups. If demographic data can be used to predict group membership, you can customize offers for individual prospective customers.

This example uses the flow named *Classifying Telecommunications Customers*, available in the example project installed with the product. The data file is *telco.csv*.

The example focuses on using demographic data to predict usage patterns. The target field *custcat* has four possible values that correspond to the four customer groups, as follows:

Table 1. Possible values for the target field

Value	Label
1	Basic Service
2	E-Service
3	Plus Service
4	Total Service

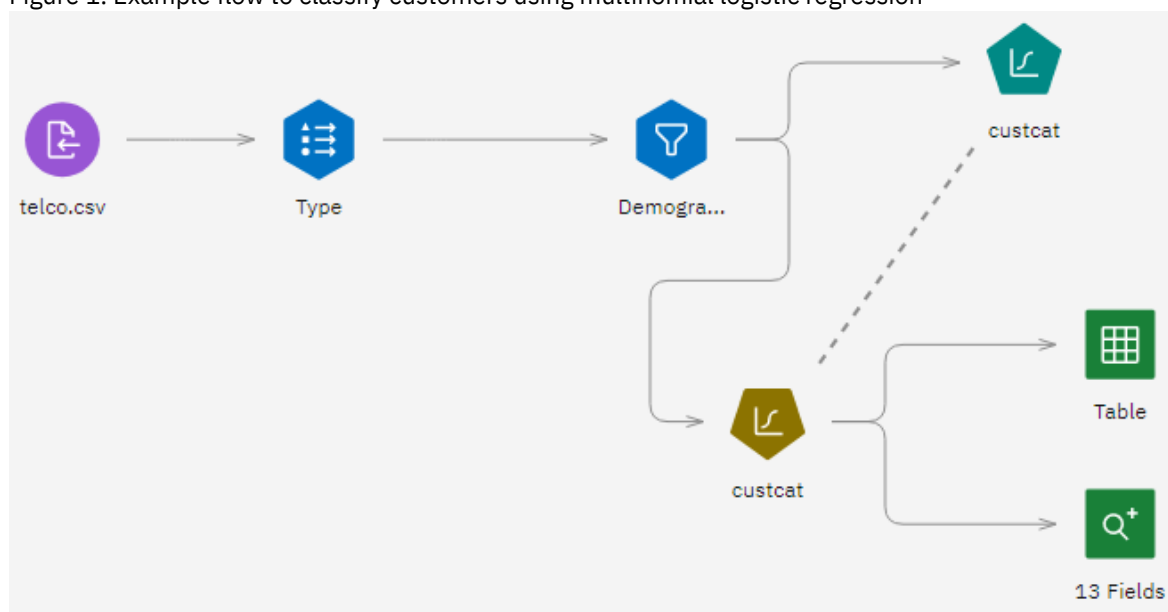
Because the target has multiple categories, a multinomial model is used. In the case of a target with two distinct categories, such as yes/no, true/false, or churn/don't churn, a binomial model could be created instead. See [Telecommunications churn](#) for more information.

- [Building the flow](#)
- [Browsing the model](#)

Parent topic: [SPSS Modeler tutorials](#)

## Building the flow

Figure 1. Example flow to classify customers using multinomial logistic regression



1. Add a Data Asset node that points to *telco.csv*.
2. Add a Type node, double-click it to open its properties, and click *Read Values*. Make sure all measurement levels are set correctly. For example, most fields with values of 0.0 and 1.0 can be regarded as flags.

Figure 2. Measurement levels

<input type="checkbox"/>	Field	Measure		Role		Value Mode		Values
<input type="checkbox"/>	#_# gender	Nominal	▼	Input	▼	Specify	▼	0.0, 1.0
<input type="checkbox"/>	#_# reside	Continuous	▼	Input	▼	Specify	▼	1, 8
<input type="checkbox"/>	#_# tollfree	Flag	▼	Input	▼	Specify	▼	0.0, 1.0
<input type="checkbox"/>	#_# equip	Flag	▼	Input	▼	Specify	▼	0.0, 1.0
<input type="checkbox"/>	#_# callcard	Flag	▼	Input	▼	Specify	▼	0.0, 1.0
<input type="checkbox"/>	#_# wireless	Flag	▼	Input	▼	Specify	▼	0.0, 1.0
<input type="checkbox"/>	#_# longmon	Continuous	▼	Input	▼	Specify	▼	0.9, 99.95
<input type="checkbox"/>	#_# tollmon	Continuous	▼	Input	▼	Specify	▼	0.0, 173.0

Notice that `gender` is more correctly considered as a field with a set of two values, instead of a flag, so leave its measurement value as Nominal.

- Set the role for the `custcat` field to Target. Leave the role for all other fields set to Input.
- Since this example focuses on demographics, use a Filter node to include only the relevant fields: `region`, `age`, `marital`, `address`, `income`, `ed`, `employ`, `retire`, `gender`, `reside`, and `custcat`). Other fields will be excluded for the purpose of this analysis. To filter them out, in the Filter node properties, click Add Columns and select the fields to exclude.

Figure 3. Filtering on demographic fields

Filter
^

Mode ⓘ
  
☒ Filter the selected fields
  
☐ Retain the selected fields (all other fields are filtered)
  
Filter Options
▼

Select Fields ⓘ
  

- Add Columns +

<input type="checkbox"/>	Field Name
<input type="checkbox"/>	callid
<input type="checkbox"/>	voice
<input type="checkbox"/>	equipmon
<input type="checkbox"/>	churn

Fields: 42 in, 31 filtered, 11 out

(Alternatively, you could change the role to None for these fields rather than excluding them, or select the fields you want to use in the modeling node.)

- In the Logistic node properties, under MODEL SETTINGS, select the Stepwise method. Also select Multinomial, Main Effects, and Include constant in equation.

Figure 4. Example flow to classify customers using multinomial logistic regression

☒ Use partitioned data

☒ Build model for each split

Procedure ⓘ

☒ Multinomial

☐ Binomial

Method ⓘ

Stepwise ▼

Base category for target ⓘ

---

Model type ⓘ

☒ Main Effects

☐ Full Factorial

☐ Custom

☒ Include constant in equation

6. Under EXPERT OPTIONS, select Expert mode, expand the Output section, and select Classification table.

Figure 5. Example flow to classify customers using multinomial logistic regression



---

### Output

---

- ☐ Summary statistics
- ☐ Likelihood ratio tests
- ☐ Asymptotic correlation
- ☐ Goodness of fit chi-square statistics
- ☐ Iteration history for every

Steps ⓘ

1

- ☐ Stepwise variable loadings
- ☐ Information criteria
- ☐ Parameter estimates

Confidence interval ⓘ

95

- ☐ Asymptotic covariance
- ☒ Classification table
- ☐ Monotonicity measures

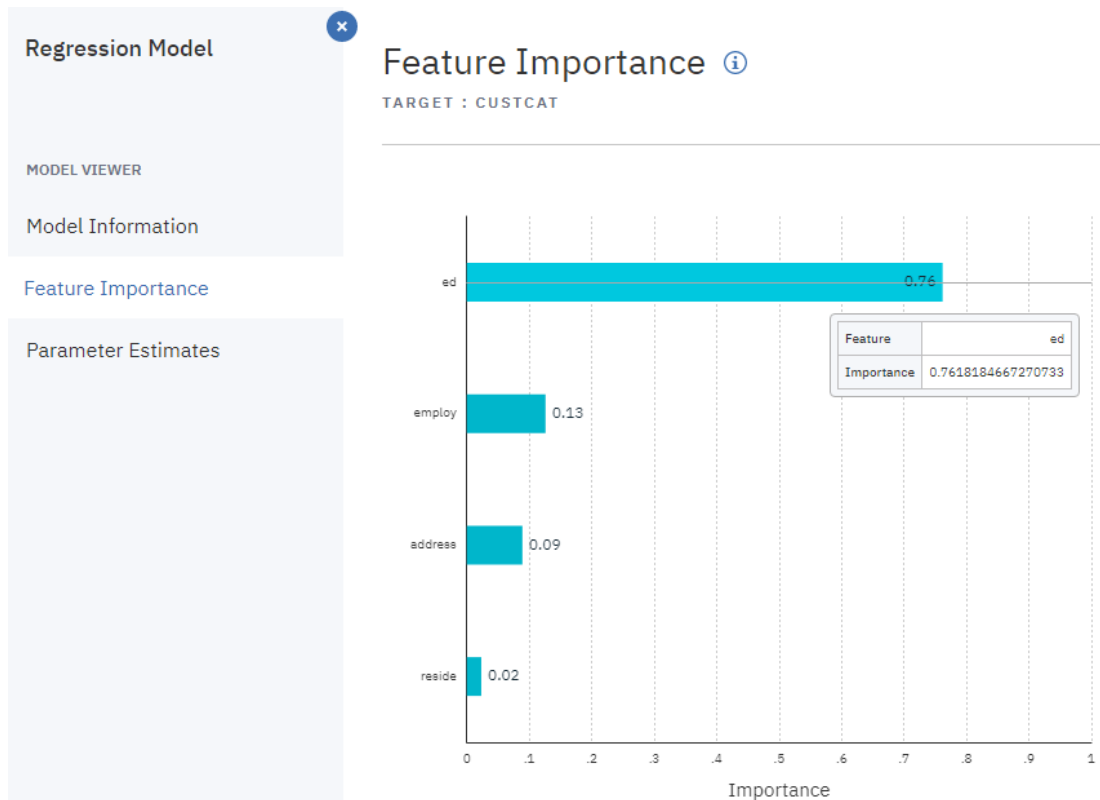
**Parent topic:** [Classifying telecommunications customers](#)

## Browsing the model

---

- Run the Logistic node to generate the model. Right-click the model nugget and select View Model.

Figure 1. Browsing the model results



You can then explore the model information, feature (predictor) importance, and parameter estimates information.

Note that these results are based on the training data only. To assess how well the model generalizes to other data in the real world, you can use a Partition node to hold out a subset of records for purposes of testing and validation.

**Parent topic:** [Classifying telecommunications customers](#)

## Telecommunications churn

Logistic regression is a statistical technique for classifying records based on values of input fields. It is analogous to linear regression, but takes a categorical target field instead of a numeric one.

For example, suppose a telecommunications provider is concerned about the number of customers it's losing to competitors. If service usage data can be used to predict which customers are liable to transfer to another provider, offers can be customized to retain as many customers as possible.

This example uses the flow named Telecommunications Churn, available in the example project installed with the product. The data file is telco.csv.

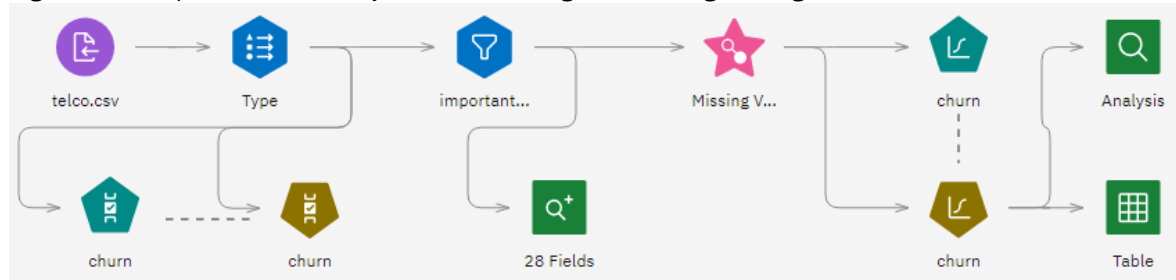
This example focuses on using usage data to predict customer loss (churn). Because the target has two distinct categories, a binomial model is used. In the case of a target with multiple categories, a multinomial model could be created instead. See [Classifying telecommunications customers](#) for more information.

- [Building the flow](#)
- [Browsing the model](#)

**Parent topic:** [SPSS Modeler tutorials](#)

## Building the flow

Figure 1. Example flow to classify customers using binomial logistic regression



1. Add a Data Asset node that points to telco.csv.
2. Add a Type node, double-click it to open its properties, and make sure all measurement levels are set correctly. For example, most fields with values of 0 and 1 can be regarded as flags, but certain fields, such as gender, are more accurately viewed as a nominal field with two values.

Figure 2. Measurement levels

<input type="checkbox"/>	Field	Measure	Role	Value Mode	Values	Check
<input type="checkbox"/>	# loglong	Continuous	Input	Instantiated	-0.105360515657...	None
<input type="checkbox"/>	# logtoll	Continuous	Input	Instantiated	1.749199854809...	None
<input type="checkbox"/>	# logequi	Continuous	Input	Instantiated	2.734367509419...	None
<input type="checkbox"/>	# logcard	Continuous	Input	Instantiated	1.011600911678...	None
<input type="checkbox"/>	# logwire	Continuous	Input	Instantiated	2.701361212951...	None
<input type="checkbox"/>	# lninc	Continuous	Input	Instantiated	2.197224577336...	None
<input type="checkbox"/>	# custcat	Nominal	Input	Instantiated	1, 2, 3, 4	None
<input type="checkbox"/>	# churn	Flag	Target	Instantiated	0, 1	None

3. Set the measurement level for the `churn` field to Flag, and set the role to Target. Leave the role for all other fields set to Input.
4. Add a Feature Selection modeling node to the Type node. You can use a Feature Selection node to remove predictors or data that don't add any useful information about the predictor/target relationship.
5. Run the flow. Right-click the resulting model nugget and select View Model. You'll see a list of the most important fields.
6. Add a Filter node after the Type node. Not all of the data in the telco.csv data file will be useful in predicting churn. You can use the filter to only select data considered to be important for use as a predictor (the fields marked as Important in the model generated in the previous step).
7. Double-click the Filter node to open its properties, select the option Retain the selected fields (all other fields are filtered), and add the following important fields from the Feature Selection model nugget:

```
tenure
age
address
income
ed
employ
equip
callcard
wireless
longmon
tollmon
equipmon
cardmon
wiremon
longten
tollten
cardten
```

voice  
pager  
internet  
callwait  
confer  
ebill  
loglong  
logtoll  
lninc  
custcat  
churn

8. Add a Data Audit output node after the Filter node. Right-click the node and run it, then open the output that was added to the Outputs pane.
9. In the lower section, look at the % Complete column, which lets you identify any fields with large amounts of missing data. In this case, the only field you need to amend is `logtoll`, which is less than 50% complete.
10. Close the output, and add a Filler node after the Filter node. Double-click the node to open its properties, click Add Columns, and select the `logtoll` field.
11. Under Replace, select Blank and null values. Click Save to close the node properties.
12. Right-click the Filler node you just created and select Create supernode. Double-click the supernode and change its name to Missing Value Imputation.
13. Add a Logistic node after the Filler node. Double-click the node to open its properties. Under MODEL SETTINGS, select the Binomial procedure and the Forwards Stepwise method.

Figure 3. Choosing model settings

Model Settings

Model Name ⓘ

☒ Auto

☐ Custom

☒ Use partitioned data

☒ Build model for each split

Procedure ⓘ

☐ Multinomial

☒ Binomial

Method ⓘ

Forwards Stepwise

14. Under EXPERT SETTINGS, select Expert.

Figure 4. Choosing expert options

Expert Options

Simple

Expert

Scale ⓘ

None

▼

Value ⓘ

1

▲▼

☐ Append all probabilities

Singularity tolerance ⓘ

tolerance.1.0E-8.label

▼

Convergence ⓘ

Output ⓘ

Stepping ⓘ

15. Click Output to open the display settings. Select At each step, Iteration history, and Parameter estimates, then click OK.

Figure 5. Choosing expert options

## Output

Display ⓘ

☒ At each step ☐ At last step

☒ Iteration history

☒ Parameter estimates

☐ Classification plots

☐ Hosmer-Lemeshow goodness-of-fit

☐ CI for exp(B) (%)

ⓘ

95

☐ Residual Diagnosis

ⓘ

☐ All cases ☒ Outliers outside (std. dev.):

ⓘ

2

Classification cutoff ⓘ

0.5

**Parent topic:** [Telecommunications churn](#)

## Browsing the model

- Right-click the Logistic node and run it to generate its model nugget. Right-click the nugget and select View Model.  
The Parameter Estimates page shows the target (churn) and inputs (predictor fields) used by the model. These are the fields that were actually chosen based on the Forwards Stepwise method, not the complete list submitted for consideration.

Figure 1. Parameter estimates showing input fields

## Parameter Estimates

TARGET : CHURN

Parameter		B	Exp(B)
Yes vs No	Constant	-0.112	0.894
	tenure	-0.037	0.964
	employ	-0.046	0.955
	equip(1)	-0.761	0.467
	callcard(1)	0.947	2.579
	cardmon	0.017	1.017
	voice(1)	-0.494	0.610
	internet(1)	-0.538	0.584
	lninc	0.294	1.341

To assess how well the model actually fits your data, a number of diagnostics are available in the expert node settings when you're building the flow.

Note also that these results are based on the training data only. To assess how well the model generalizes to other data in the real world, you would use a Partition node to hold out a subset of records for purposes of testing and validation.

**Parent topic:** [Telecommunications churn](#)

## Forecasting bandwidth utilization

An analyst for a national broadband provider is required to produce forecasts of user subscriptions to predict utilization of bandwidth. Forecasts are needed for each of the local markets that make up the national subscriber base.

You'll use time series modeling to produce forecasts for the next three months for a number of local markets.

- **Forecasting with the Time Series node**

This example uses the flow Forecasting Bandwidth Utilization, available in the example project installed with the product. The data file is broadband\_1.csv.

**Parent topic:** [SPSS Modeler tutorials](#)

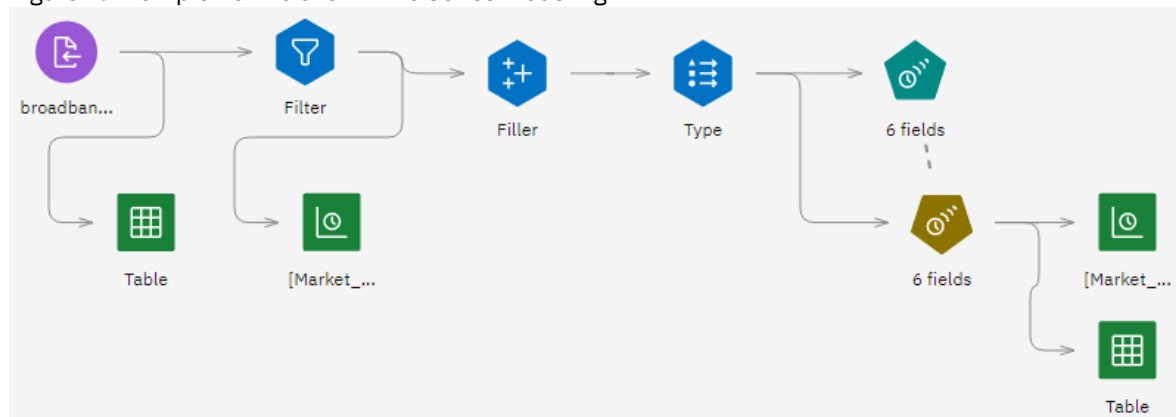
## Forecasting with the Time Series node

This example uses the flow Forecasting Bandwidth Utilization, available in the example project installed with the product. The data file is broadband\_1.csv.

In Watson Studio, you can produce multiple time series models in a single operation. The broadband\_1.csv data file has monthly usage data for each of 85 local markets. For the purposes of this example, only the first five series will be used; a separate model will be created for each of these five series, plus a total.

The file also includes a date field that indicates the month and year for each record. This field will be used to label records. The date field reads into Watson Studio as a string, but to use the field in Watson Studio you will convert the storage type to numeric Date format using a Filler node.

Figure 1. Example flow to show Time Series modeling



The Time Series node requires that each series be in a separate column, with a row for each interval. Watson Studio provides methods for transforming data to match this format if necessary.

Figure 2. Monthly subscription data for broadband local markets



	Market_1 Decimal	Market_2 Decimal	Market_3 Decimal	Market_4 Decimal	Market_5 Decimal	Market_6 Decimal
1	3750	11489	11659	4571	2205	5488
2	3846	11984	12228	4825	2301	5672
3	3894	12266	12897	5041	2352	5802
4	4010	12801	13716	5211	2490	5899
5	4147	13291	14647	5383	2534	6017
6	4335	13828	15419	5496	2664	6137
7	4554	14273	16108	5747	2738	6250
8	4744	14664	16958	5885	2754	6439
9	4885	15130	17642	6053	2874	6701
10	5020	15851	18453	6229	2975	6957
11	5208	16509	19181	6320	3042	7111
12	5379	17225	19885	6499	3095	7275
13	5574	18173	20565	6593	3199	7380
14	5828	19287	21155	6680	3207	7633
15	5942	20171	21655	6757	3298	7985
16	6139	21379	21964	6804	3387	8236
17	6244	22067	22756	6915	3450	8464
18	6274	23074	23464	7035	3528	8575
19	6347	23729	24324	7151	3546	8817
20	6399	24803	25351	7304	3604	9041

- [Creating the flow](#)
- [Examining the data](#)
- [Defining the dates](#)
- [Defining the targets](#)
- [Setting the time intervals](#)
- [Creating the model](#)
- [Examining the model](#)
- [Summary](#)

**Parent topic:** [Forecasting bandwidth utilization](#)

## Creating the flow

---

1. Add a Data Asset node that points to broadband\_1.csv.
2. To simplify the model, use a Filter node to filter out the Market\_6 to Market\_85 fields and the MONTH\_ and YEAR\_ fields.

Figure 1. Example flow to show Time Series modeling

Select Fields for Filter		
<input type="text"/> Search in column Field name		Filter: #
<input type="checkbox"/>	Field Name	Data Type
<input type="checkbox"/>	Market_1	# integer
<input type="checkbox"/>	Market_2	# integer
<input type="checkbox"/>	Market_3	# integer
<input type="checkbox"/>	Market_4	# integer
<input type="checkbox"/>	Market_5	# integer
<input checked="" type="checkbox"/>	Market_6	# integer
<input checked="" type="checkbox"/>	Market_7	# integer
<input checked="" type="checkbox"/>	Market_8	# integer
<input checked="" type="checkbox"/>	Market_9	# integer
<input checked="" type="checkbox"/>	Market_10	# integer

**Parent topic:** [Forecasting with the Time Series node](#)

## Examining the data

It's always a good idea to have a feel for the nature of your data before building a model.

Does the data exhibit seasonal variations? Although Watson Studio can automatically find the best seasonal or nonseasonal model for each series, you can often obtain faster results by limiting the search to nonseasonal models when seasonality is not present in your data. Without examining the data for each of the local markets, we can get a rough picture of the presence or absence of seasonality by plotting the total number of subscribers over all five markets.

Figure 1. Plotting the total number of subscribers

Plot ⓘ

- ☒ Selected series  
☐ Selected Time Series models

Series ⓘ

⊖ Add Columns ⊕

<input type="checkbox"/>	Field Name
<input type="checkbox"/>	Market_1
<input type="checkbox"/>	Market_2
<input type="checkbox"/>	Market_3
<input type="checkbox"/>	Market_4

☐ Use custom x axis field label

X axis label

...

☐ Display series in separate panel

☐ Normalize

Display:

☒ Line

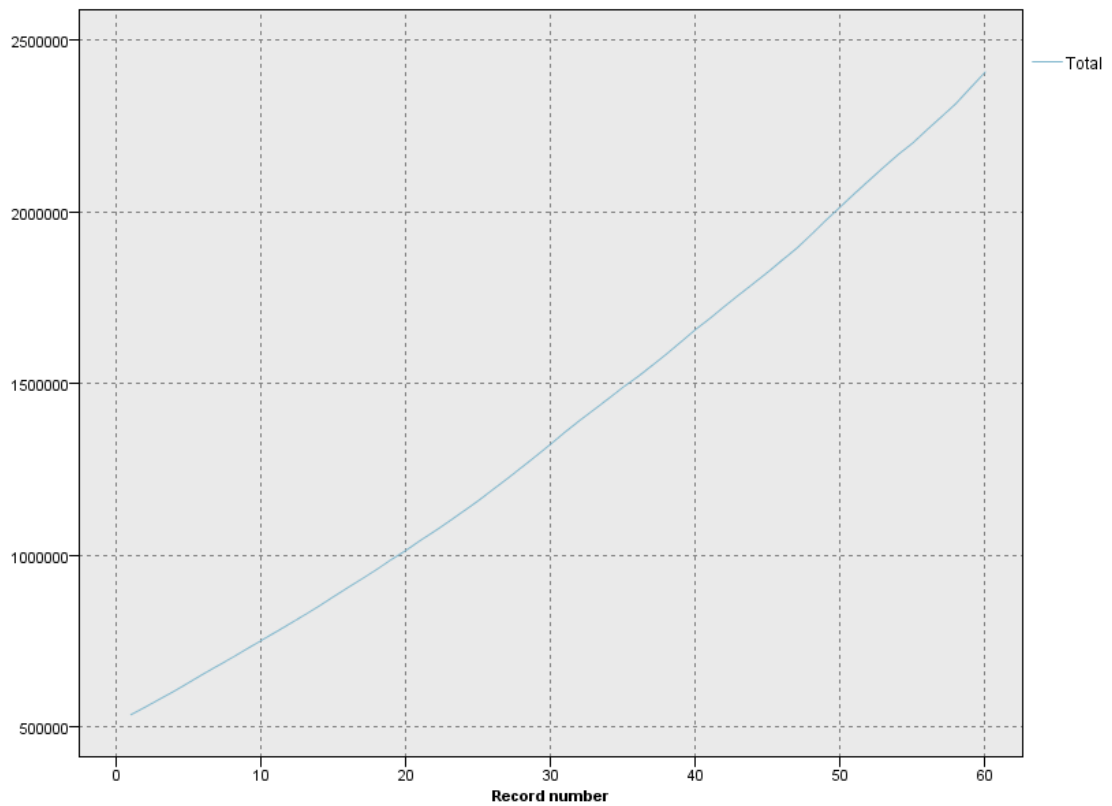
☐ Point

☐ Smoother

☒ Limit records

1. From the Graphs palette, attach a Time Plot node to the Filter node.
2. Add the `Total` field to the Series list.
3. Deselect the Display series in separate panel and Normalize options. Save the changes.
4. Right-click the Time Plot node and run it, then open the output that was generated.

Figure 2. Time plot of the Total field



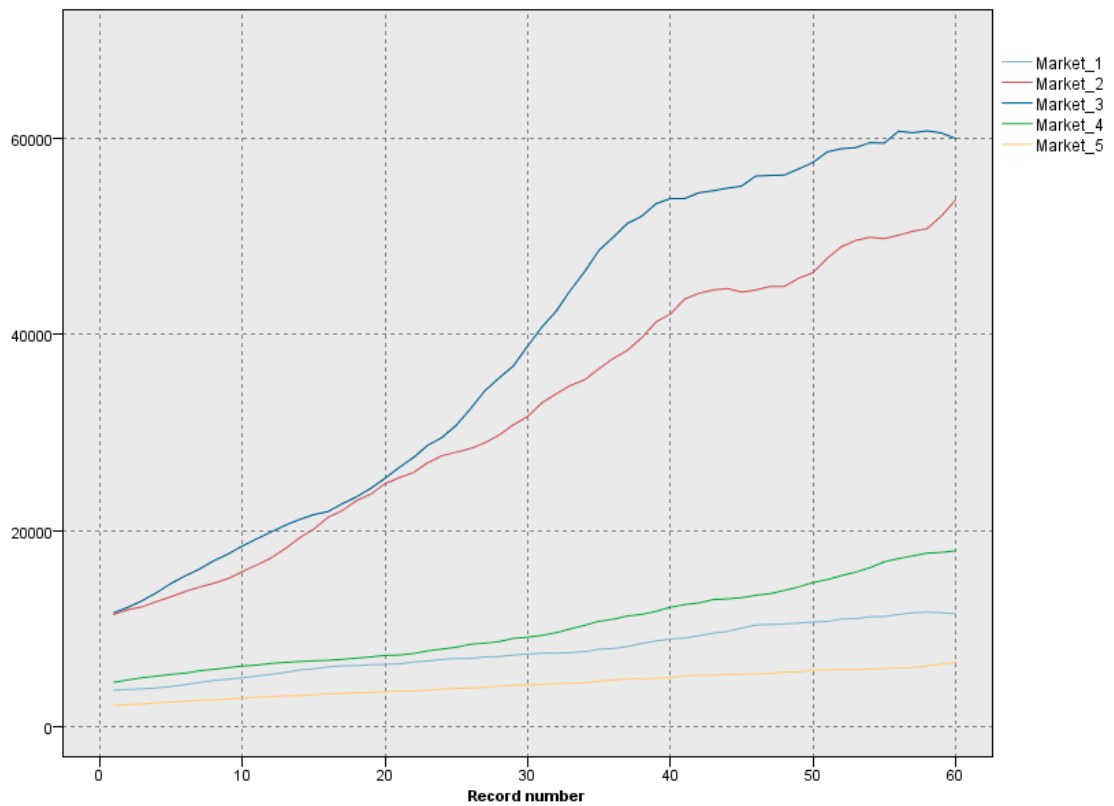
The series exhibits a very smooth upward trend with no hint of seasonal variations. There might be individual series with seasonality, but it appears that seasonality isn't a prominent feature of the data in general.

Of course, you should inspect each of the series before ruling out seasonal models. You can then separate out series exhibiting seasonality and model them separately.

Watson Studio makes it easy to plot multiple series together.

5. Double-click the Time Plot node to open its properties again.
6. Remove the `Total` field from the Series list.
7. Add the `Market_1` through `Market_5` fields to the list.
8. Run the Time Plot node again.

Figure 3. Time plot of multiple fields



Inspection of each of the markets reveals a steady upward trend in each case. Although some markets are a little more erratic than others, there's no evidence of seasonality.

**Parent topic:** [Forecasting with the Time Series node](#)

## Defining the dates

Now you need to change the storage type of the `DATE_` field to date format.

1. Attach a Filler node to the Filter node, then double-click the Filler node to open its properties
2. Add the `DATE_` field, set the Replace option to Always, and set the Replace with value to `to_date (DATE_)`.

Figure 1. Setting the date storage type

Settings ^

Fill in fields ⓘ

Field Name

DATE\_

DATE\_

Replace ⓘ

Always ▾

Condition ⓘ

@BLANK (@FIELD)

Replace with ⓘ

to\_date (DATE\_)

**Parent topic:** [Forecasting with the Time Series node](#)

## Defining the targets

1. Add a Type node after the Filler node, then double-click the Type node to open its properties.
2. Set the role to None for the `DATE_` field. Set the role to Target for all other fields (the `Market_n` fields plus the `Total` field).
3. Click Read Values to populate the Values column.

Figure 1. Setting the role for fields

Default Mode ⓘ

☒ Read metadata ☐ Pass (do not scan)

Type Operations

[Read Values](#) [Clear All Values](#)

🔍 Search in column Field

<input type="checkbox"/>	Field	Measure	Role	Value Mode	Values	Check
<input type="checkbox"/>	# Market_1	Continuous ▾	Target ▾	Specify ▾	3750, 11731	None ▾ ⚙️
<input type="checkbox"/>	# Market_2	Continuous ▾	Target ▾	Specify ▾	11489, 53704	None ▾ ⚙️
<input type="checkbox"/>	# Market_3	Continuous ▾	Target ▾	Specify ▾	11659, 60755	None ▾ ⚙️
<input type="checkbox"/>	# Market_4	Continuous ▾	Target ▾	Specify ▾	4571, 17977	None ▾ ⚙️
<input type="checkbox"/>	# Market_5	Continuous ▾	Target ▾	Specify ▾	2205, 6611	None ▾ ⚙️
<input type="checkbox"/>	# Total	Continuous ▾	Target ▾	Specify ▾	536413, 2406762	None ▾ ⚙️
<input type="checkbox"/>	📅 DATE_	Continuous ▾	None ▾	Specify ▾	1999-01-01, 2003...	None ▾ ⚙️

Parent topic: [Forecasting with the Time Series node](#)

## Setting the time intervals

1. Add a Time Series node and attach it to the Type node. Double-click the node to edit its properties.
2. Under OBSERVATIONS AND TIME INTERVAL, select `DATE_` as the Time/Date field.
3. Select Months as the time interval.

Figure 1. Setting the time interval

Observations and Time Interval ^

Observations ⓘ

☒ Observations are specified by a date/time field

☐ Observations are defined as periods or cyclic periods

Time/Date Field

DATE\_ ▾

Time Interval

Months ▾

4. Under MODEL OPTIONS, select the Extend records into the future option and set the value to 3.

Figure 2. Setting the forecast period

Model Options ^

Model Name ⓘ

☒ Auto

☐ Custom

Confidence Limit Width(%) ⓘ

95

☐ Continue estimation using existing model(s)

☐ Build scoring model only

Forecast

☒ Extend records into the future

3

**Parent topic:** [Forecasting with the Time Series node](#)

## Creating the model

1. Double-click the Time Series node to open its properties.
2. Under **FIELDS**, add all 5 of the markets to the Candidate Inputs lists. Also add the `Total` field to the Targets list.
3. Under **BUILD OPTIONS - GENERAL**, make sure the Expert Modeler method is selected using all default settings. Doing so enables the Expert Modeler to decide the most appropriate model to use for each time series.

Figure 1. Choosing the Expert Modeler method for Time Series



Build Options - General ^

Method ⓘ

Expert Modeler ▾

Model Type ⓘ

☒ All models
   
☐ Exponential smoothing models only
   
☐ ARIMA models only
   
☒ Expert Modeler considers seasonal modes
   
☐ Expert Modeler considers sophisticated exponential smoothing models

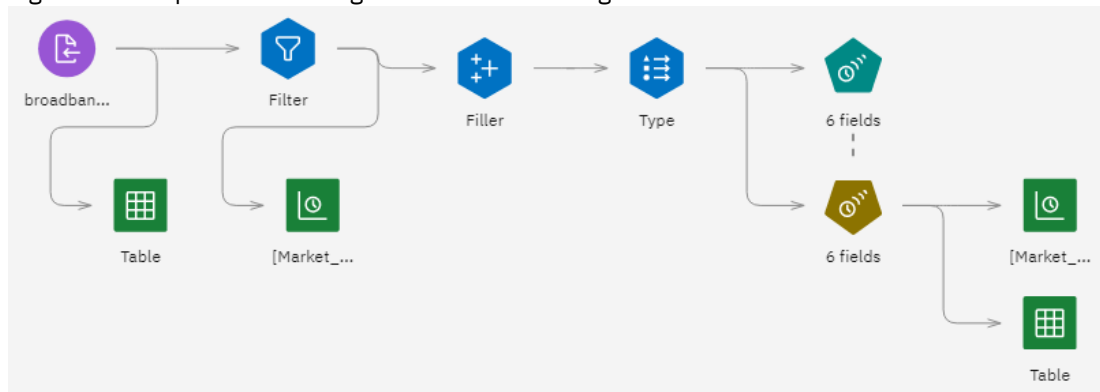
---

Outliers

☐ Detect outliers automatically

- Save the settings and then run the flow. A Time Series model nugget is generated. Attach it to the Time Series node.
- Attach a Table node to the Time Series model nugget and run the flow again.

Figure 2. Example flow showing Time Series modeling



There are now three new rows appended to the end of the original data. These are the rows for the forecast period, in this case January to March 2004.

Several new columns are also present now. The `$TS-` columns are added by the Time Series node. The columns indicate the following for each row (that is, for each interval in the time series data):

Column	Description
<code>\$TS-colname</code>	The generated model data for each column of the original data.
<code>\$TSLCI-colname</code>	The lower confidence interval value for each column of the generated model data.
<code>\$TSUCI-colname</code>	The upper confidence interval value for each column of the generated model data.

Column	Description
\$TS-Total	The total of the \$TS- <i>colname</i> values for this row.
\$TSLCI-Total	The total of the \$TSLCI- <i>colname</i> values for this row.
\$TSUCI-Total	The total of the \$TSUCI- <i>colname</i> values for this row.

The most significant columns for the forecast operation are the \$TS-Market\_n, \$TSLCI-Market\_n, and \$TSUCI-Market\_n columns. In particular, these columns in the last three rows contain the user subscription forecast data and confidence intervals for each of the local markets.

**Parent topic:** [Forecasting with the Time Series node](#)

## Examining the model

1. Right-click the Time Series model nugget and select View Model to see information about the models generated for each of the markets.

Figure 1. Time Series models generated for the markets

TARGET	MODELING METHOD	STATIONARY R SQUARED	R SQUARED	RMSE	MSE
<a href="#">Market 2</a>	ARIMA(1,0,0) (0,0,0)	0.999	0.999	481.562	231,902.035
<a href="#">Market 4</a>	ARIMA(1,0,0) (1,0,0)	0.998	0.998	182.265	33,220.526
<a href="#">Market 1</a>	ARIMA(1,0,0) (0,0,0)	0.998	0.998	101.526	10,307.483
<a href="#">Market 5</a>	ARIMA(0,0,2) (1,0,0)	0.997	0.997	65.066	4,233.584
<a href="#">Market 3</a>	ARIMA(0,2,1) (0,0,0)	0.157	0.999	418.262	174,943.378
<a href="#">Total</a>	ARIMA(1,2,0) (0,0,0)	0.093	1.000	1,996.750	3,987,008.852

2. In the left TARGET column, select any of the markets. Then go to Model Information. The Number of Predictors row shows how many fields were used as predictors for each target.

The other rows in the Model Information tables show various goodness-of-fit measures for each model. Stationary R-Squared measures how a model is better than a baseline model. If the final model is ARIMA(p,d,q) (P,D,Q), the baseline model is ARIMA(0,d,0)(0,D,0). If the final model is an Exponential Smoothing model, then d is 2 for Brown and Holt model and 1 for other models, and D is 1 if the seasonal length is greater than 1, otherwise D is 0. A negative stationary R squared means that the model under consideration is worse than the baseline model. Zero stationary R squared means that the model is as good or bad as the baseline model and a positive stationary R squared means the model is better than the baseline model

The Statistic and df lines, and the Significance under Parameter Estimates, relate to the Ljung-Box statistic, a test of the randomness of the residual errors in the model. The more random the errors, the better the model is

likely to be. Statistic is the Ljung-Box statistic itself, while df (degrees of freedom) indicates the number of model parameters that are free to vary when estimating a particular target.

The Significance gives the significance value of the Ljung-Box statistic, providing another indication of whether the model is correctly specified. A significance value less than 0.05 indicates that the residual errors are not random, implying that there is structure in the observed series that is not accounted for by the model.

Taking both the Stationary R-Squared and Significance values into account, the models that the Expert Modeler has chosen for `Market_3`, and `Market_4` are quite acceptable. The Significance values for `Market_1`, `Market_2`, and `Market_5` are all less than 0.05, indicating that some experimentation with better-fitting models for these markets might be necessary.

The display shows a number of additional goodness-of-fit measures. The R-Squared value gives an estimation of the total variation in the time series that can be explained by the model. As the maximum value for this statistic is 1.0, our models are fine in this respect.

RMSE is the root mean square error, a measure of how much the actual values of a series differ from the values predicted by the model, and is expressed in the same units as those used for the series itself. As this is a measurement of an error, we want this value to be as low as possible. At first sight it appears that the models for `Market_2` and `Market_3`, while still acceptable according to the statistics we have seen so far, are less successful than those for the other three markets.

These additional goodness-of-fit measures include the mean absolute percentage errors (MAPE) and its maximum value (MAXAPE). Absolute percentage error is a measure of how much a target series varies from its model-predicted level, expressed as a percentage value. By examining the mean and maximum across all models, you can get an indication of the uncertainty in your predictions.

The MAPE value shows that all models display a mean uncertainty of around 1%, which is very low. The MAXAPE value displays the maximum absolute percentage error and is useful for imagining a worst-case scenario for your forecasts. It shows that the largest percentage error for most of the models falls in the range of roughly 1.8% to 3.7%, again a very low set of figures, with only `Market_4` being higher at close to 7%.

The MAE (mean absolute error) value shows the mean of the absolute values of the forecast errors. Like the RMSE value, this is expressed in the same units as those used for the series itself. MAXAE shows the largest forecast error in the same units and indicates worst-case scenario for the forecasts.

Although these absolute values are interesting, it's the values of the percentage errors (MAPE and MAXAPE) that are more useful in this case, as the target series represent subscriber numbers for markets of varying sizes.

Do the MAPE and MAXAPE values represent an acceptable amount of uncertainty with the models? They are certainly very low. This is a situation in which business sense comes into play, because acceptable risk will change from problem to problem. We'll assume that the goodness-of-fit statistics fall within acceptable bounds, so let's go on to look at the residual errors.

Examining the values of the autocorrelation function (ACF) and partial autocorrelation function (PACF) for the model residuals provides more quantitative insight into the models than simply viewing goodness-of-fit statistics.

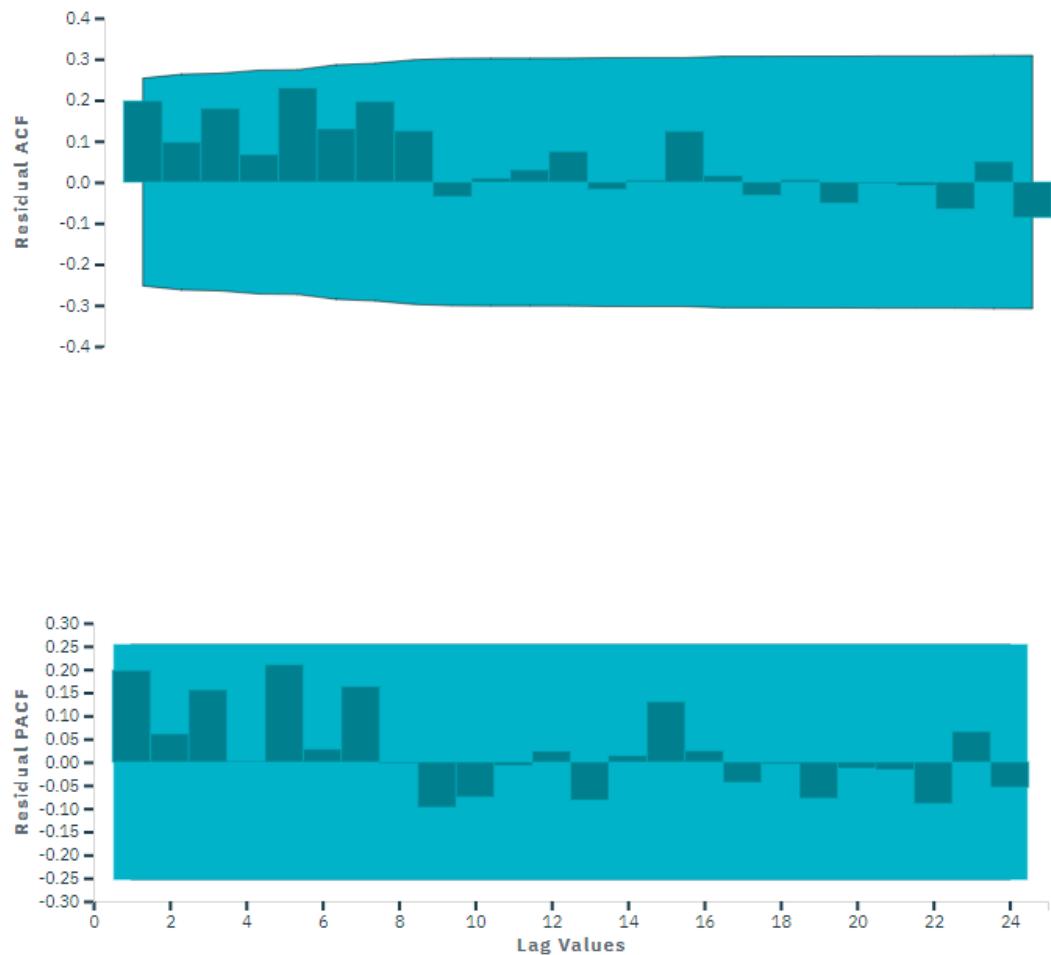
A well-specified time series model will capture all of the nonrandom variation, including seasonality, trend, and cyclic and other factors that are important. If this is the case, any error should not be correlated with itself (autocorrelated) over time. A significant structure in either of the autocorrelation functions would imply that the underlying model is incomplete.

3. For the fourth market, click Correlogram to display the values of the autocorrelation function (ACF) and partial autocorrelation function (PACF) for the residual errors in the model.

Figure 2. ACF and PACF values for the fourth market

## Correlogram ⓘ

TARGET : MARKET\_4



In these plots, the original values of the error variable have been lagged (under BUILD OPTIONS - OUTPUT) up to the default value of 24 time periods and compared with the original value to see if there's any correlation over time. Ideally, the bars representing all lags of ACF and PACF should be within the shaded area. However, in practice, there may be some lags that extend outside of the shaded area. This is because, for example, some larger lags may not have been tried for inclusion in the model in order to save computation time. Some lags are insignificant and are removed from the model. If you want to improve the model further and don't care whether these lags are redundant or not, these plots serve as tips for you as to which lags are potential predictors.

Should this occur, you'd need to check the lower (PACF) plot to see whether the structure is confirmed there. The PACF plot looks at correlations after controlling for the series values at the intervening time points.

The values for `Market_4` are all within the shaded area, so we can continue and check the values for the other markets.

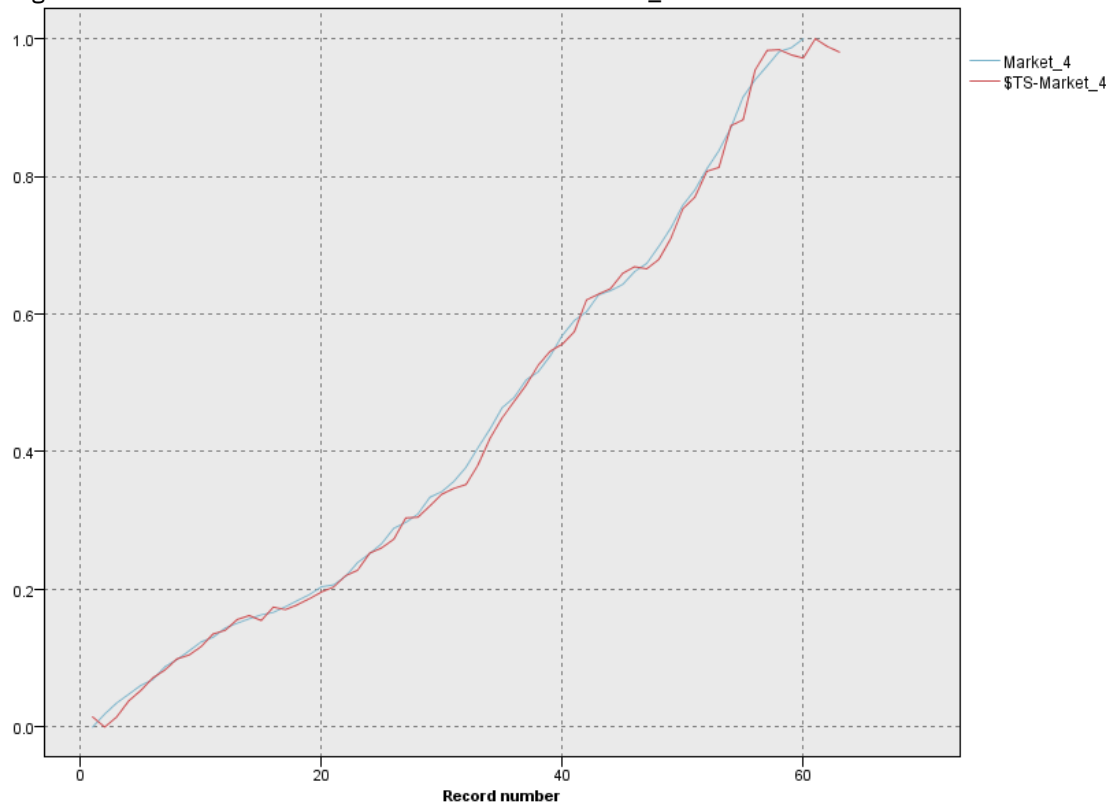
#### 4. Open the Correlogram for each of the other markets and the totals.

The values for the other markets all show some values outside the shaded area, confirming what we suspected earlier from their Significance values. We'll need to experiment with some different models for those markets at some point to see if we can get a better fit, but for the rest of this example, we'll concentrate on what else we can learn from the `Market_4` model.

5. Return to your flow canvas. Attach a new Time Plot node to the Time Series model nugget. Double-click the node to open its properties.
6. Deselect the Display series in separate panel option.
7. For the Series list, add the `Market_4` and `$TS-Market_4` fields.

8. Save the properties, then right-click the Time Plot node and select Run to generate a line graph of the actual and forecast data for the first of the local markets.  
Notice how the forecast (\$TS-Market\_4) line extends past the end of the actual data. You now have a forecast of expected demand for the next three months in this market.

Figure 3. Time Plot of actual and forecast data for Market\_4



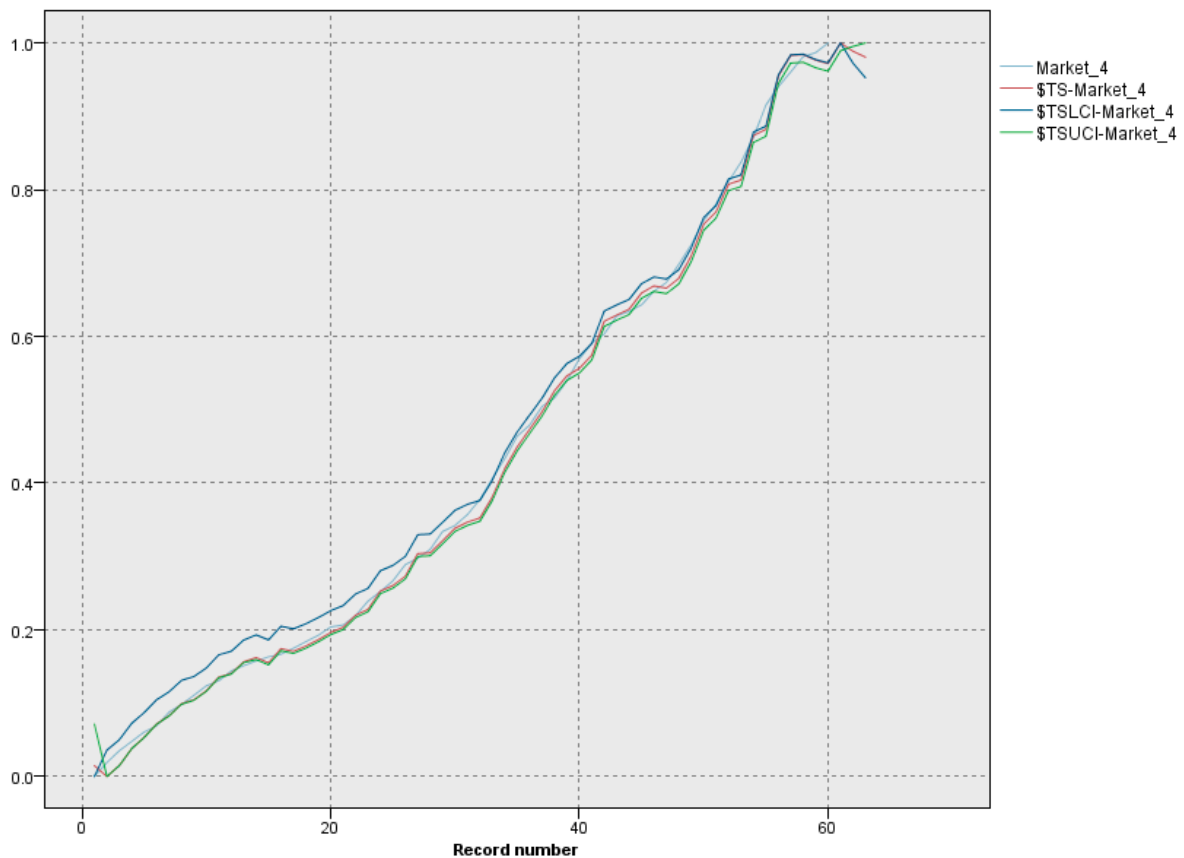
The lines for actual and forecast data over the entire time series are very close together on the graph, indicating that this is a reliable model for this particular time series.

You have a reliable model for this particular market, but what margin of error does the forecast have? You can get an indication of this by examining the confidence interval.

9. Double-click the last Time Plot node in the flow (the one labeled Market\_4 \$TS-Market\_4).
10. Add the \$TSLCI-Market\_4 and \$TSUCI-Market\_4 fields to the Series list.
11. Save the properties and run the node again.

Now you have the same graph as before, but with the upper (\$TSUCI) and lower (\$TSLCI) limits of the confidence interval added. Notice how the boundaries of the confidence interval diverge over the forecast period, indicating increasing uncertainty as you forecast further into the future. However, as each time period goes by, you'll have another (in this case) month's worth of actual usage data on which to base your forecast. In a real-world scenario, you could read the new data into the flow and reapply your model now that you know it's reliable.

Figure 4. Time Plot with confidence interval added



**Parent topic:** [Forecasting with the Time Series node](#)

## Summary

You've learned how to use the Expert Modeler to produce forecasts for multiple time series. In a real-world scenario, you could now transform nonstandard time series data into a format suitable for input to a Time Series node.

**Parent topic:** [Forecasting with the Time Series node](#)

## Forecasting catalog sales

A catalog company is interested in forecasting monthly sales of its men's clothing line, based on 10 years of their sales data.

This example uses the flow Forecasting Catalog Sales, available in the example project installed with the product. The data file is catalog\_seasfac.csv.

We've seen in an earlier tutorial how you can let the Expert Modeler decide which is the most appropriate model for your time series. Now it's time to take a closer look at the two methods that are available when choosing a model yourself: exponential smoothing and ARIMA.

To help you decide on an appropriate model, it's a good idea to plot the time series first. Visual inspection of a time series can often be a powerful guide in helping you choose. In particular, you need to ask yourself:

- Does the series have an overall trend? If so, does the trend appear constant or does it appear to be dying out with time?
- Does the series show seasonality? If so, do the seasonal fluctuations seem to grow with time or do they appear constant over successive periods?

- [Creating the flow](#)

- **Examining the data**  
The series shows a general upward trend; that is, the series values tend to increase over time. The upward trend is seemingly constant, which indicates a linear trend.
- **Exponential smoothing**  
Building a best-fit exponential smoothing model involves determining the model type (whether the model needs to include trend, seasonality, or both) and then obtaining the best-fit parameters for the chosen model.
- **ARIMA**  
With the ARIMA procedure, you can create an autoregressive integrated moving-average (ARIMA) model that is suitable for finely tuned modeling of time series.
- **Summary**  
You've successfully modeled a complex time series, incorporating not only an upward trend but also seasonal and other variations. You've also seen how, through trial and error, you can get closer and closer to an accurate model, which you can then use to forecast future sales.

Parent topic: [SPSS Modeler tutorials](#)

## Creating the flow

1. Create a new flow and add a Data Asset node that points to catalog\_seasfac.csv.
2. Connect a Type node to the Data Asset node and double-click it to open its properties.
3. Click Read Values. For the `men` field, set the role to Target.

Figure 1. Specifying the target field

Search in column Field							
<input type="checkbox"/>	Field	Measure	Role	Value Mode	Values	Check	
<input type="checkbox"/>	date	Continuous	Input	Specify	1989-01-01, 199...	None	
<input type="checkbox"/>	# men	Continuous	Target	Specify	3245.18, 38609.66	None	
<input type="checkbox"/>	# women	Continuous	Input	Specify	16578.93, 80245....	None	
<input type="checkbox"/>	# jewel	Continuous	Input	Specify	5983.55, 38231.57	None	
<input type="checkbox"/>	# mail	Continuous	Input	Specify	1147, 15263	None	

4. Set the role for all other fields to `None` and click `Save`.
5. Attach a Time Plot graph node to the Type node and double-click it.

Figure 2. Plotting the time series

Plot ^

Plot ⓘ

☒ Selected series
 ☐ Selected Time Series models

Series ⓘ

Add Columns

<input type="checkbox"/> Field Name
<input type="checkbox"/> men

☒ Use custom x axis field label

X axis label

date ▼

☒ Display series in separate panel

☐ Normalize

6. For the Plot, add the field `men` to the Series list.
7. Select Use custom x axis field label and select `date`.
8. Deselect the Normalize option and click Save.
9. Run the flow.

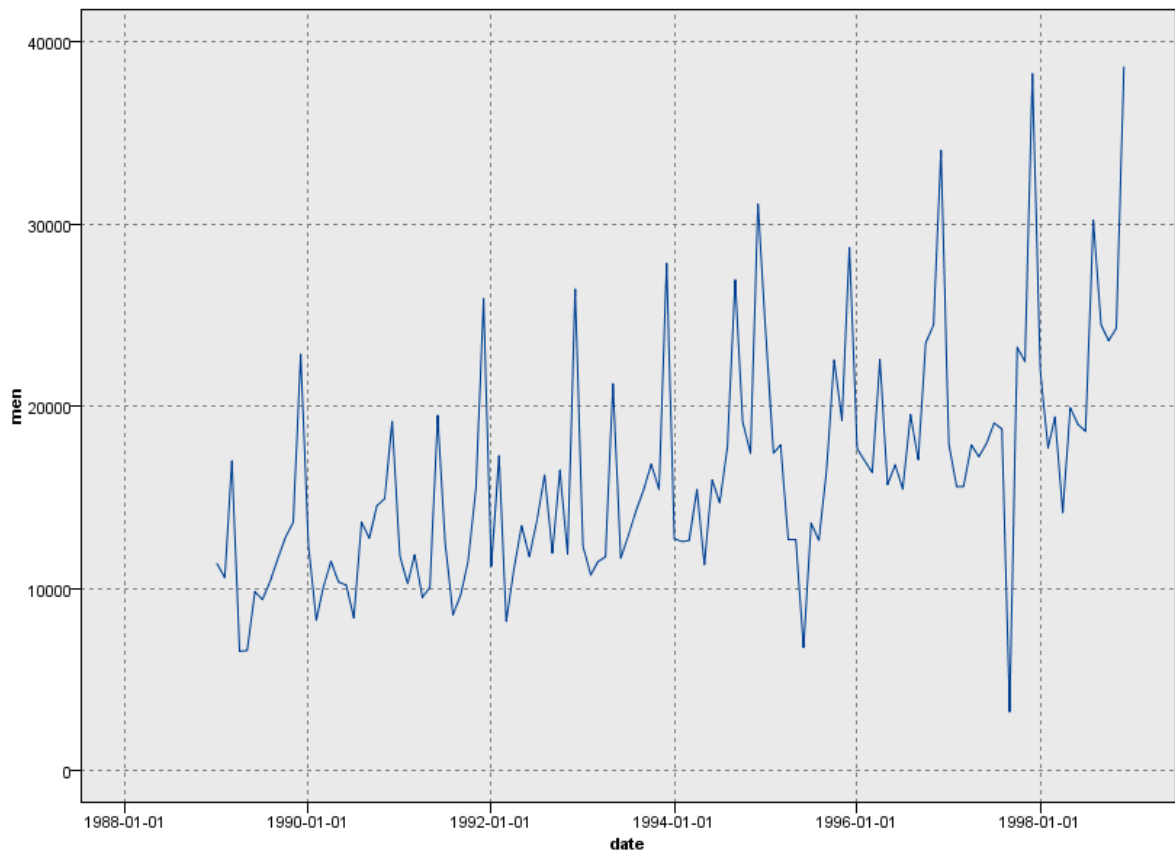
**Parent topic:** [Forecasting catalog sales](#)

## Examining the data

The series shows a general upward trend; that is, the series values tend to increase over time. The upward trend is seemingly constant, which indicates a linear trend.

Figure 1. Actual sales of men's clothing





The series also has a distinct seasonal pattern with annual highs in December, as indicated by the vertical lines on the graph. The seasonal variations appear to grow with the upward series trend, which suggests multiplicative rather than additive seasonality.

Now that you've identified the characteristics of the series, you're ready to try modeling it. The exponential smoothing method is useful for forecasting series that exhibit trend, seasonality, or both. As we've seen, this data exhibits both characteristics.

**Parent topic:** [Forecasting catalog sales](#)

## Exponential smoothing


Building a best-fit exponential smoothing model involves determining the model type (whether the model needs to include trend, seasonality, or both) and then obtaining the best-fit parameters for the chosen model.

The plot of men's clothing sales over time suggested a model with both a linear trend component and a multiplicative seasonality component. This implies a Winters' model. First, however, we will explore a simple model (no trend and no seasonality) and then a Holt's model (incorporates linear trend but no seasonality). This will give you practice in identifying when a model is not a good fit to the data, an essential skill in successful model building.

We'll start with a simple exponential smoothing model.

1. Add a Time Series node and attach it to the Type node. Double-click the node to edit its properties.
2. Under OBSERVATIONS AND TIME INTERVAL, select `date` as the time/date field.
3. Select Months as the time interval.

Figure 1. Setting the time interval

men	
Fields	▼
Observations and Time Interval	^
<p>Observations ⓘ</p> <p><input checked="" type="radio"/> Observations are specified by a date/time field</p> <p><input type="radio"/> Observations are defined as periods or cyclic periods</p> <p>Time/Date Field</p> <p>date ▼</p> <p>Time Interval</p> <p>Months ▼</p>	

- Under BUILD OPTIONS - GENERAL, select Exponential Smoothing for the Method.
- Set Model Type to Simple. Click Save.

Figure 2. Setting the method

Build Options - General	^
<p>Method ⓘ</p> <p>Exponential Smoothing ▼</p> <p>Model Type ⓘ</p> <p>Simple ▼</p>	

- Run the flow to create the model nugget.
- Attach a Time Plot node to the model nugget.
- Under Plot, add the fields `men` and `$TS-men` to the Series list.
- Select the option Use custom x axis field label and select the `date` field.
- Deselect the Display series in separate panel and Normalize options. Click Save.

Figure 3. Setting the plot options

Plot
^

Plot ⓘ

☒ Selected series
☐ Selected Time Series models

Series ⓘ

- Add Columns +

<input type="checkbox"/>	Field Name
<input type="checkbox"/>	men
<input type="checkbox"/>	\$TS-men
<input type="checkbox"/>	

☒ Use custom x axis field label

X axis label

date
v

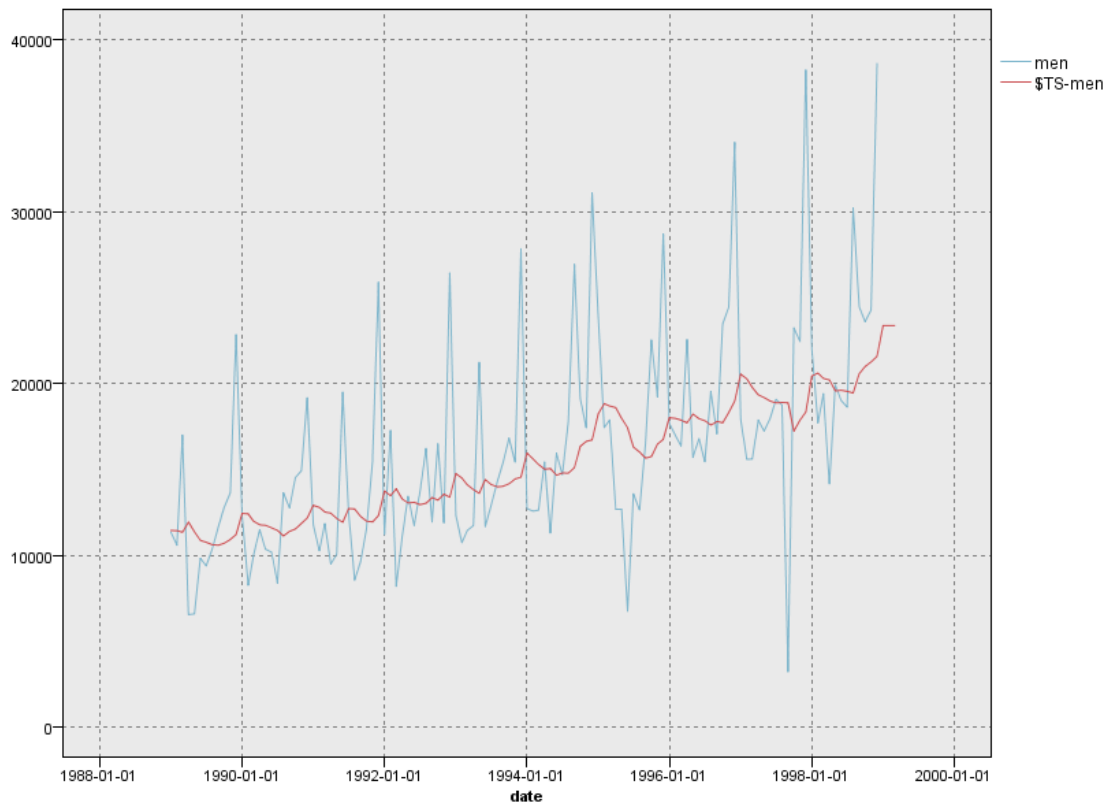
☐ Display series in separate panel

☐ Normalize

11. Run the flow and then open the output.

The men plot represents the actual data, while \$TS-men denotes the time series model.

Figure 4. Simple exponential smoothing model

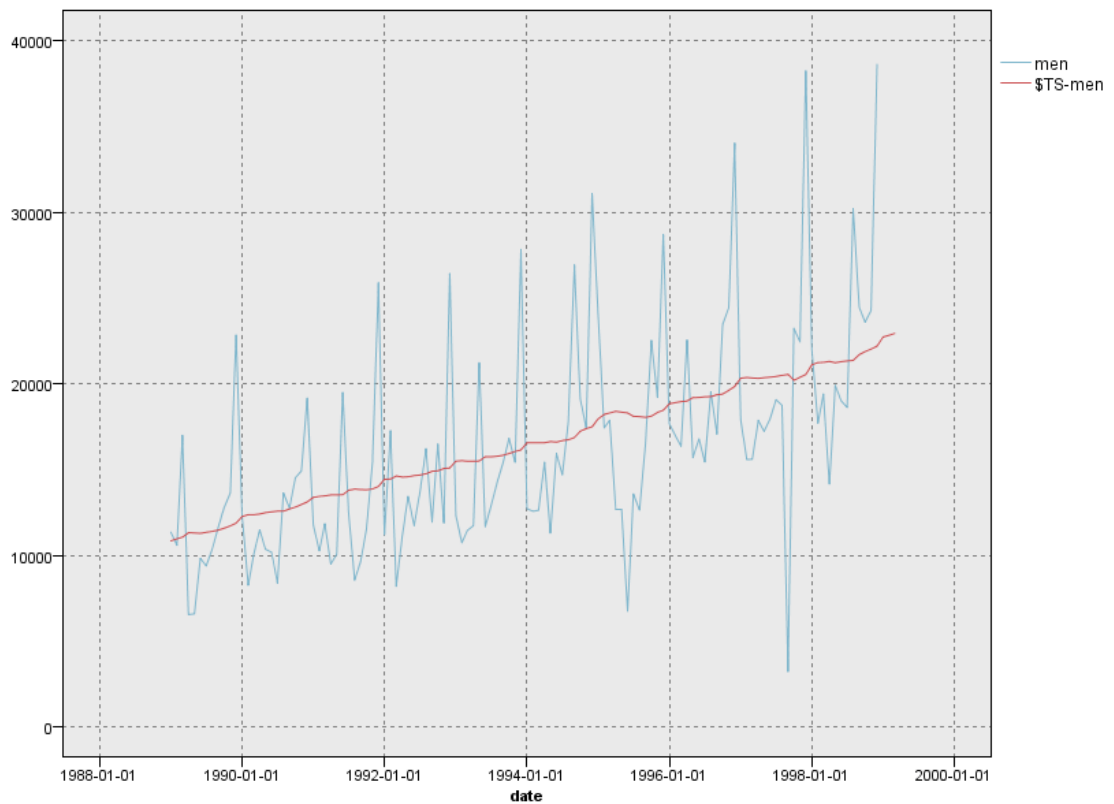


Although the simple model does, in fact, exhibit gradual (and rather ponderous) upward trend, it takes no account of seasonality. You can safely reject this model.

Now let's try a Holt's linear model. This should at least model the trend better than the simple model, although it too is unlikely to capture the seasonality.

12. Double-click the Time Series node. Under BUILD OPTIONS - GENERAL, with Exponential Smoothing still selected as the method, select HoltsLinearTrend as the model type.
13. Click Save and run the flow again to regenerate the model nugget. Open the output.

Figure 5. Holt's linear trend model

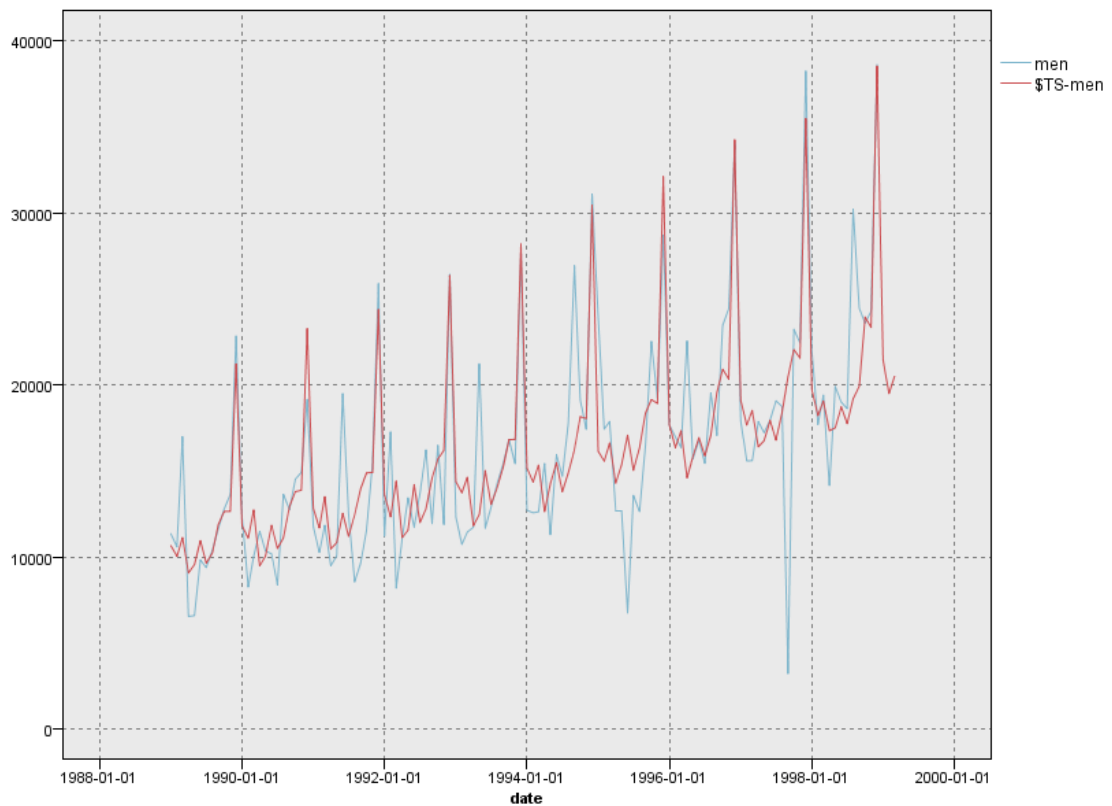


Holt's model displays a smoother upward trend than the simple model, but it still takes no account of the seasonality, so you can disregard this one too.

You may recall that the initial plot of men's clothing sales over time suggested a model incorporating a linear trend and multiplicative seasonality. A more suitable candidate, therefore, might be Winters' model.

14. Double-click the Time Series node again to edit its properties.
15. Under BUILD OPTIONS - GENERAL, with Exponential Smoothing still selected as the method, select WintersMultiplicative as the model type.
16. Run the flow.

Figure 6. Winters' multiplicative model



This looks better. The model reflects both the trend and the seasonality of the data. The dataset covers a period of 10 years and includes 10 seasonal peaks occurring in December of each year. The 10 peaks present in the predicted results match up well with the 10 annual peaks in the real data.

However, the results also underscore the limitations of the Exponential Smoothing procedure. Looking at both the upward and downward spikes, there is significant structure that's not accounted for.

If you're primarily interested in modeling a long-term trend with seasonal variation, then exponential smoothing may be a good choice. To model a more complex structure such as this one, we need to consider using the ARIMA procedure.

**Parent topic:** [Forecasting catalog sales](#)

## ARIMA

With the ARIMA procedure, you can create an autoregressive integrated moving-average (ARIMA) model that is suitable for finely tuned modeling of time series.

ARIMA models provide more sophisticated methods for modeling trend and seasonal components than do exponential smoothing models, and they have the added benefit of being able to include predictor variables in the model.

Continuing the example of the catalog company that wants to develop a forecasting model, we have seen how the company has collected data on monthly sales of men's clothing along with several series that might be used to explain some of the variation in sales. Possible predictors include the number of catalogs mailed and the number of pages in the catalog, the number of phone lines open for ordering, the amount spent on print advertising, and the number of customer service representatives.

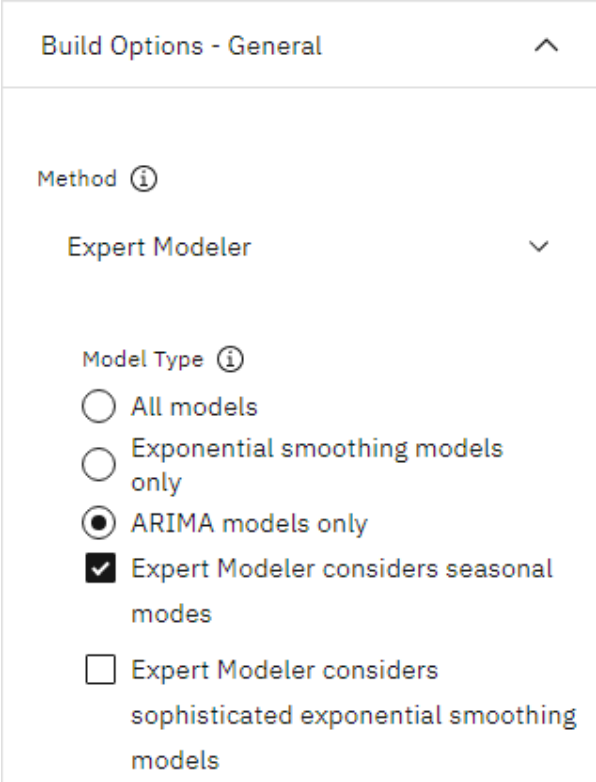
Are any of these predictors useful for forecasting? Is a model with predictors really better than one without? Using the ARIMA procedure, we can create a forecasting model with predictors, and see if there's a significant difference in predictive ability over the exponential smoothing model with no predictors.

With the ARIMA method, you can fine-tune the model by specifying orders of autoregression, differencing, and moving average, as well as seasonal counterparts to these components. Determining the best values for these components manually can be a time-consuming process involving a good deal of trial and error so, for this example, we'll let the Expert Modeler choose an ARIMA model for us.

We'll try to build a better model by treating some of the other variables in the dataset as predictor variables. The ones that seem most useful to include as predictors are the number of catalogs mailed (`mail`), the number of pages in the catalog (`page`), the number of phone lines open for ordering (`phone`), the amount spent on print advertising (`print`), and the number of customer service representatives (`service`).

- 1. Double-click the Type node to open its properties.
- 2. Set the role for `mail`, `page`, `phone`, `print`, and `service` to Input.
- 3. Ensure that the role for `men` is set to Target and that all the remaining fields are set to None.
- 4. Click Save.
- 5. Double-click the Time Series node.
- 6. Under BUILD OPTIONS - GENERAL, select Expert Modeler for the method.
- 7. Select the options ARIMA models only and Expert Modeler considers seasonal models.

Figure 1. Choosing only ARIMA models



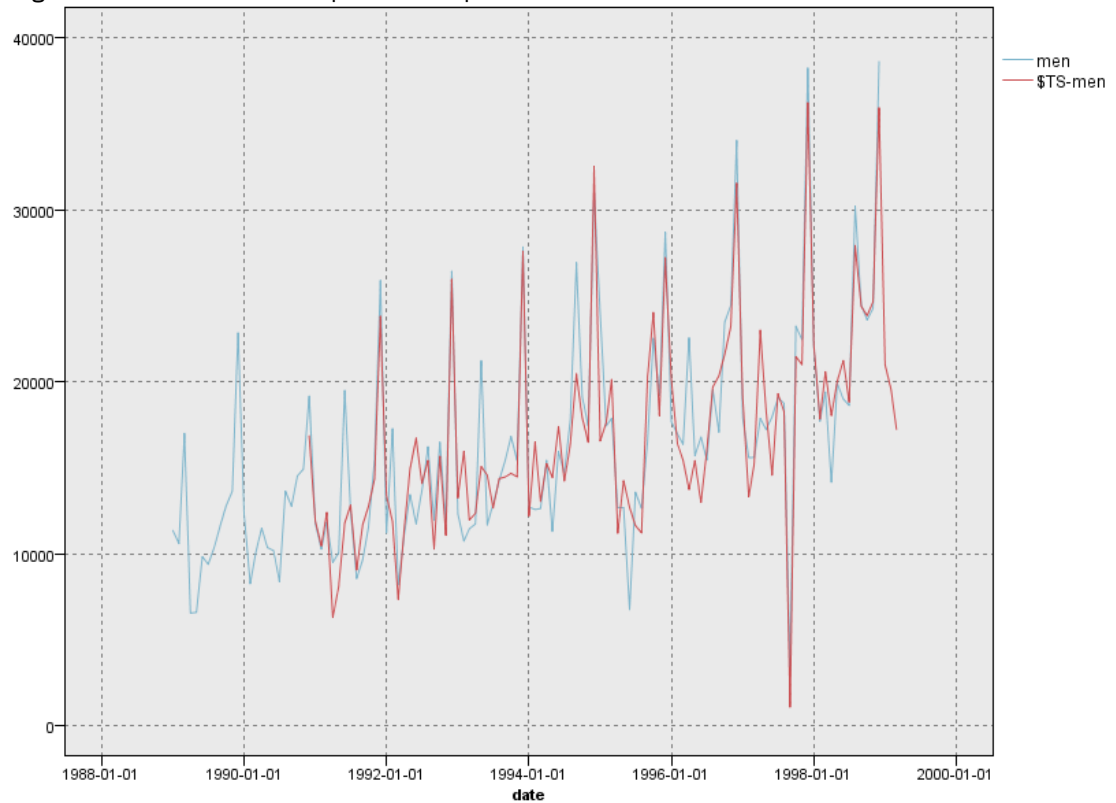
- 8. Click Save and run the flow.
- 9. Right-click the model nugget and select View Model. Click `men` and then click Model information. Notice how the Expert Modeler has chosen only two of the five specified predictors as being significant to the model.

Figure 2. Expert Modeler chooses two predictors

Model Building Method	ARIMA
	Non-seasonal $p=0,d=0,q=0$ ; Seasonal $p=1,d=1,q=0$
Number of Predictors	2

10. Open the latest chart output.

Figure 3. ARIMA model with predictors specified



This model improves on the previous one by capturing the large downward spike as well, making it the best fit so far.

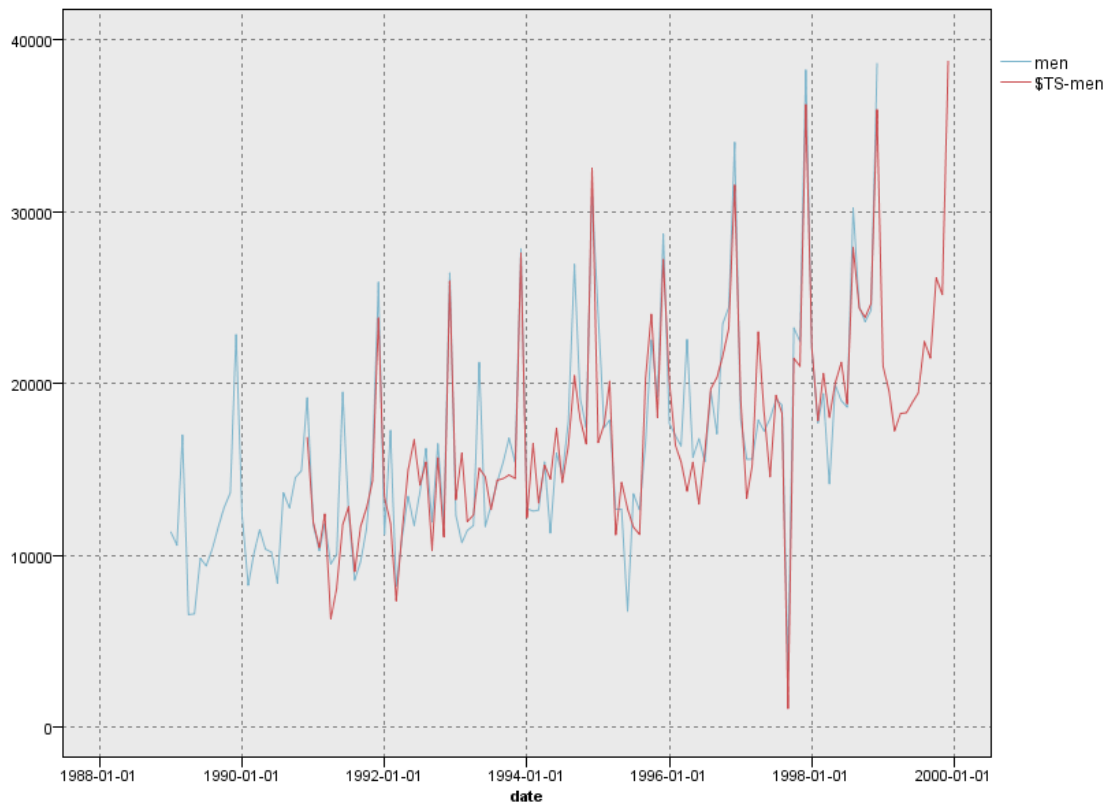
We could try refining the model even further, but any improvements from this point on are likely to be minimal. We've established that the ARIMA model with predictors is preferable, so let's use the model we have just built. For the purposes of this example, we'll forecast sales for the coming year.

11. Double-click the Time Series node.
12. Under MODEL OPTIONS, select the option Extend records into the future and set its value to 12.
13. Select the Compute future values of inputs option.
14. Click Save and run the flow.

The forecast looks good. As expected, there's a return to normal sales levels following the December peak, and a steady upward trend in the second half of the year, with sales in general above those for the previous year.

Figure 4. Sales forecast extended by 12 months





**Parent topic:** [Forecasting catalog sales](#)

## Summary

You've successfully modeled a complex time series, incorporating not only an upward trend but also seasonal and other variations. You've also seen how, through trial and error, you can get closer and closer to an accurate model, which you can then use to forecast future sales.

In practice, you would need to reapply the model as your actual sales data are updated - for example, every month or every quarter - and produce updated forecasts.

**Parent topic:** [Forecasting catalog sales](#)

## Making offers to customers (self-learning)

The Self-Learning Response Model (SLRM) node generates and enables the updating of a model that allows you to predict which offers are most appropriate for customers and the probability of the offers being accepted. These sorts of models are most beneficial in customer relationship management, such as marketing applications or call centers.

This example is based on a fictional banking company. The marketing department wants to achieve more profitable results in future campaigns by matching the right offer of financial services to each customer. Specifically, the example uses a Self-Learning Response model to identify the characteristics of customers who are most likely to respond favorably based on previous offers and responses and to promote the best current offer based on the results.

This example uses the flow named Making Offers to Customers - Self-Learning, available in the example project installed with the product. The data files are pm\_customer\_train1.csv, pm\_customer\_train2.csv, and pm\_customer\_train3.csv.

1. On the My Projects screen, click Example Project.
2. Scroll down to the Modeler flows section, click View all, and select the Making Offers to Customers - Self-Learning flow.

## Existing data

The banking company has historical data tracking the offers made to customers in past campaigns, along with the responses to those offers. This data also includes demographic and financial information that can be used to predict response rates for different customers.

Figure 1. Responses to previous offers

	Data	Profile	Visualizations						
	customer_id Decimal	campaign String	response Decimal	response_date Timestamp	purchase Decimal	purchase_date String	product_id Decimal	Rowid Decimal	age Decimal
1	7	Car loan	0		0		NA	1	19
2	13	Car loan	0		0		NA	2	44
3	15	Car loan	0		0		NA	3	45
4	16	Car loan	1	2006-07-05	0		183	761	43
5	23	Car loan	0		0		NA	4	42
6	24	Car loan	0		0		NA	5	39
7	30	Car loan	0		0		NA	6	23
8	30	Savings	0		0		NA	7	23
9	33	Car loan	0		0		NA	8	24
10	42	Savings	0		0		NA	9	35
11	42	Car loan	0		0		NA	10	35
12	52	Car loan	0		0		NA	11	43
13	57	Car loan	0		0		NA	12	51
14	63	Car loan	1	2006-07-14	0		183	1501	67
15	74	Car loan	0		0		NA	13	31
16	74	Savings	0		0		NA	14	31
17	75	Car loan	0		0		NA	15	30
18	82	Car loan	0		0		NA	16	32
19	89	Savings	0		0		NA	17	46
20	89	Car loan	0		0		NA	18	46

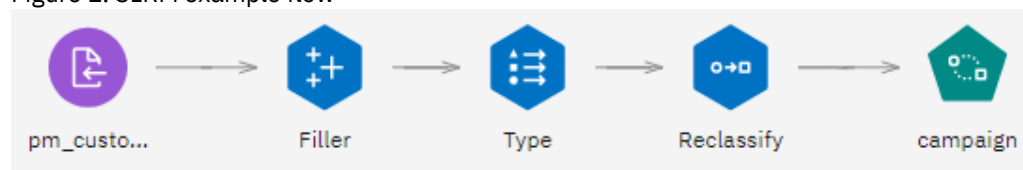
- [Building the flow](#)
- [Browsing the model](#)

Parent topic: [SPSS Modeler tutorials](#)

## Building the flow


1. Add a Data Asset node that points to pm\_customer\_train1.csv.


Figure 1. SLRM example flow





2. Attach a Filler node to the Data Asset node. Double-click the node to open its properties and, under Fill in fields, select `campaign`.
3. Select a Replace type of Always.
4. In the Replace with text box, enter `to_string(campaign)` and click Save.


Figure 2. Derive a campaign field


Settings 


Fill in fields 


 Add Columns 

<input type="checkbox"/> Field Name
<input type="checkbox"/> campaign


Replace 


Always 

Condition 



```
@BLANK (@FIELD)
```

Replace with 



```
to_string(campaign)
```

5. Add a Type node and set the Role to `None` for the following fields:
  - `customer_id`
  - `response_date`
  - `purchase_date`
  - `product_id`
  - `Rowid`
  - `X_random`
6. Set the Role to `Target` for the `campaign` and `response` fields. These are the fields on which you want to base your predictions. Set the Measurement to `Flag` for the `response` field.
7. Click Read Values then click Save. Because the `campaign` field data shows as a list of numbers (1, 2, 3, and 4), you can reclassify the fields to have more meaningful titles.
8. Add a Reclassify node after the Type node and open its properties.

9. Under Reclassify Into, select Existing field.
10. Under Reclassify Field, select `campaign`.
11. Click Get values. The campaign values are added to the ORIGINAL VALUE column.
12. In the NEW VALUE column, enter the following campaign names in the first four rows:
  - Mortgage
  - Car loan
  - Savings
  - Pension
13. Click Save.

Figure 3. Reclassify the campaign names

Mode ⓘ

☒ Single

☐ Multiple

Reclassify Into ⓘ

☐ New field

☒ Existing field

Reclassify Field ⓘ

campaign

New Field Name ⓘ

Reclassify2

Get values Copy Clear new

Automatically Reclassify

Values ⓘ

<input type="checkbox"/>	ORIGINAL VALUE	NEW VALUE
<input type="checkbox"/>	1	Mortgage
<input type="checkbox"/>	2	Car loan
<input type="checkbox"/>	3	Savings
<input type="checkbox"/>	4	Pension

For Unspecified Values Use ⓘ

☒ Original value ☐ Default value

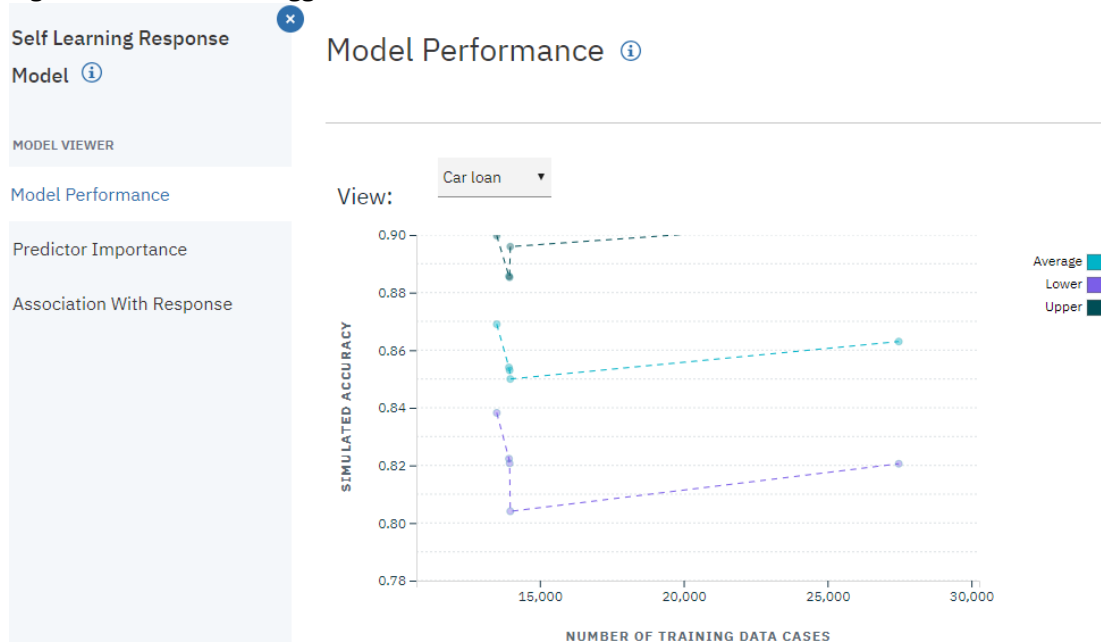
14. Attach an SLRM modeling node to the Reclassify node. Select `campaign` for the Target field, and `response` for the Target response field.
15. Under MODEL OPTIONS, for Maximum number of predictions per record, reduce the number to 2. This means that for each customer there will be two offers identified that have the highest probability of being accepted.
16. Make sure Take account of model reliability is selected, then click Save and run the flow.

**Parent topic:** [Making offers to customers \(self-learning\)](#)

## Browsing the model

1. Right-click the model nugget and select View Model. The initial view shows the estimated accuracy of the predictions for each offer. You can also click Predictor Importance to see the relative importance of each predictor in estimating the model, or click Association With Response to show the correlation of each predictor with the target variable.
2. To switch between each of the four offers for which there are prediction, use the View drop-down.

Figure 1. SLRM model nugget



3. Return to the flow.
4. Disconnect the Data Asset node that points to pm\_customer\_train1.csv.
5. Add a new Data Asset node that points to pm\_customer\_train2.csv and connect it to the Filler node.
6. Double-click the SLRM node and select Continue training existing model (under BUILD OPTIONS). Click Save.
7. Run the flow to regenerate the model nugget. Then right-click it and select View Model. The model now shows the revised estimates of accuracy of the predictions for each offer.
8. Add a new Data Asset node that points to pm\_customer\_train3.csv and connect it to the Filler node
9. Run the flow again, then right-click the model nugget and select View Model.

The model now shows the final estimated accuracy of the predictions for each offer. As you can see, the average accuracy fell slightly as you added the additional data sources. However, this fluctuation is a minimal amount and may be attributed to slight anomalies within the available data.

10. Attach a Table node to the generated model nugget, then right-click the Table node and run it. In the Outputs pane, open the table output that was just generated.  
On the far right side of the table, the predictions show which offers a customer is most likely to accept and the confidence that they'll accept, depending on each customer's details. For example, in the first row, there's only a 13.2% confidence rating (denoted by the value 0.132 in the \$SC-campaign-1 column) that a customer who previously took out a car loan will accept a pension if offered one. However, the second and third lines show two more customers who also took out a car loan; in their cases, there is a 95.7% confidence that they, and other customers with similar histories, would open a savings account if offered one, and over 80% confidence that they would accept a pension.

Figure 2. Model output - predicted offers and confidences

\$S- campaign- 1	\$SC- campaign- 1	\$S- campaign- 2	\$SC- campaign- 2
Pension	0.132	Mortgage	0.107
Savings	0.957	Pension	0.844
Savings	0.957	Pension	0.802
Pension	0.132	Mortgage	0.107
Pension	0.805	Savings	0.284
Pension	0.132	Mortgage	0.107
Pension	0.132	Mortgage	0.107
Pension	0.132	Mortgage	0.107
Pension	0.132	Mortgage	0.107
Pension	0.132	Mortgage	0.107
Pension	0.132	Mortgage	0.107
Pension	0.132	Mortgage	0.107
Savings	0.957	Mortgage	0.823
Savings	0.164	Pension	0.132
Savings	0.957	Pension	0.868
Pension	0.132	Mortgage	0.107
Pension	0.132	Mortgage	0.107
Pension	0.132	Mortgage	0.107

Explanations of the mathematical foundations of the modeling methods used in SPSS Modeler are available in the [IBM SPSS Modeler Algorithms Guide](#).

Note that these results are based on the training data only. To assess how well the model generalizes to other data in the real world, you would use a Partition node to hold out a subset of records for purposes of testing

and validation.

**Parent topic:** [Making offers to customers \(self-learning\)](#)

## Retail sales promotion

This example deals with fictitious data that describes retail product lines and the effects of promotion on sales.

Your goal in this example is to predict the effects of future sales promotions. Similar to the [condition monitoring example](#), the data mining process consists of the exploration, data preparation, training, and test phases.

This example uses the flow named Retail Sales Promotion, available in the example project installed with the product. The data files are goods1n.csv and goods2n.csv.

1. On the My Projects screen, click Example Project.
2. Scroll down to the Modeler flows section, click View all, and select the Retail Sales Promotion flow.

- [Examining the data](#)
- [Learning and testing](#)

**Parent topic:** [SPSS Modeler tutorials](#)


## Examining the data

Each record contains:

- **Class.** Product type.
- **Cost.** Unit price.
- **Promotion.** Index of amount spent on a particular promotion.
- **Before.** Revenue before promotion.
- **After.** Revenue after promotion.

The flow is simple. It displays the data in a table. The two revenue fields (**Before** and **After**) are expressed in absolute terms. However, it seems likely that the increase in revenue after the promotion (and presumably as a result of it) would be a more useful figure.

Figure 1. Effects of promotion on product sales



Class	Cost	Promotion	Before	After
Confection	23.990	1467	114957	122762
Drink	79.290	1745	123378	137097
Luxury	81.990	1426	135246	141172
Confection	74.180	1098	231389	244456
Confection	90.090	1968	235648	261940
Meat	69.850	1486	148885	156232
Meat	100.150	1248	123760	128441
Luxury	21.010	1364	251072	268134



The flow also contains a node to derive this value, expressed as a percentage of the revenue before the promotion, in a field called `Increase`. A table shows this field.

Figure 2. Increase in revenue after promotion

Class	Cost	Promotion	Before	After	Increase
Confection	23.990	1467	114957	122762	6.789
Drink	79.290	1745	123378	137097	11.119
Luxury	81.990	1426	135246	141172	4.382
Confection	74.180	1098	231389	244456	5.647
Confection	90.090	1968	235648	261940	11.157
Meat	69.850	1486	148885	156232	4.935
Meat	100.150	1248	123760	128441	3.782
Luxury	21.010	1364	251072	268134	6.796

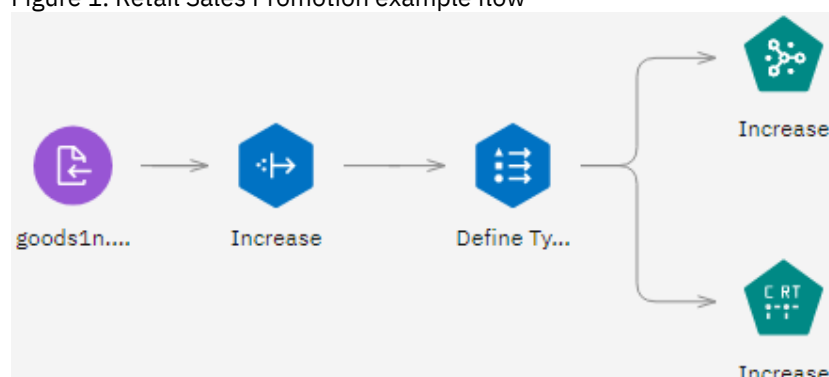
For each class of product, and almost linear relationship exists between the increase in revenue and the cost of the promotion. Therefore, it seems likely that a decision tree or neural network could predict, with reasonable accuracy, the increase in revenue from the other available fields.

**Parent topic:** [Retail sales promotion](#)

## Learning and testing

The flow trains a neural network and a decision tree to make this prediction of revenue increase.

Figure 1. Retail Sales Promotion example flow



After you run the flow to generate the model nuggets, you can test the results of the learning process. You do this by connecting the decision tree and network in series between the Type node and a new Analysis node, changing the Data Asset import node to point to `goods2n.csv`, and running the Analysis node. From the output of this node, in particular from the linear correlation between the predicted increase and the correct answer, you will find that the trained systems predict the increase in revenue with a high degree of success.

Further exploration might focus on the cases where the trained systems make relatively large errors. These could be identified by plotting the predicted increase in revenue against the actual increase. Outliers on this graph could be selected using the interactive graphics within SPSS Modeler, and from their properties, it might be possible to tune the data description or learning process to improve accuracy.

## Condition monitoring

This example concerns monitoring status information from a machine and the problem of recognizing and predicting fault states.

The data is created from a fictitious simulation and consists of a number of concatenated series measured over time. Each record is a snapshot report on the machine in terms of the following:

- **Time.** An integer.
- **Power.** An integer.
- **Temperature.** An integer.
- **Pressure.** 0 if normal, 1 for a momentary pressure warning.
- **Uptime.** Time since last serviced.
- **Status.** Normally 0, changes to an error code if an error occurs (101, 202, or 303).
- **Outcome.** The error code that appears in this time series, or 0 if no error occurs. (These codes are available only with the benefit of hindsight.)

This example uses the flow named Condition Monitoring, available in the example project installed with the product. The data files are cond1n.csv and cond2n.csv.

For each time series, there's a series of records from a period of normal operation followed by a period leading to the fault, as shown in the following table:

Time	Power	Temperature	Pressure	Uptime	Status	Outcome
0	1059	259	0	404	0	0
1	1059	259	0	404	0	0
			...			
51	1059	259	0	404	0	0
52	1059	259	0	404	0	0
53	1007	259	0	404	0	303
54	998	259	0	404	0	303
			...			
89	839	259	0	404	0	303
90	834	259	0	404	303	303
0	965	251	0	209	0	0
1	965	251	0	209	0	0
			...			
51	965	251	0	209	0	0
52	965	251	0	209	0	0
53	938	251	0	209	0	101
54	936	251	0	209	0	101
			...			
208	644	251	0	209	0	101
209	640	251	0	209	101	101

The following process is common to most data mining projects:

- Examine the data to determine which attributes may be relevant to the prediction or recognition of the states of interest.
- Retain those attributes (if already present), or derive and add them to the data, if necessary.
- Use the resultant data to train rules and neural nets.

- Test the trained systems using independent test data.
- [Examining the data](#)
- [Data preparation](#)
- [Learning](#)
- [Testing](#)

**Parent topic:** [SPSS Modeler tutorials](#)

## Examining the data

For the first part of the process, imagine you have a flow that plots a number of graphs. If the time series of temperature or power contains visible patterns, you could differentiate between impending error conditions or possibly predict their occurrence. For both temperature and power, the flow plots the time series associated with the three different error codes on separate graphs, yielding six graphs. Select nodes separate the data associated with the different error codes.

The graphs clearly display patterns distinguishing 202 errors from 101 and 303 errors. The 202 errors show rising temperature and fluctuating power over time; the other errors don't. However, patterns distinguishing 101 from 303 errors are less clear. Both errors show even temperature and a drop in power, but the drop in power seems steeper for 303 errors.

Based on these graphs, it appears that the presence and rate of change for both temperature and power, as well as the presence and degree of fluctuation, are relevant to predicting and distinguishing faults. These attributes should therefore be added to the data before applying the learning systems.

**Parent topic:** [Condition monitoring](#)

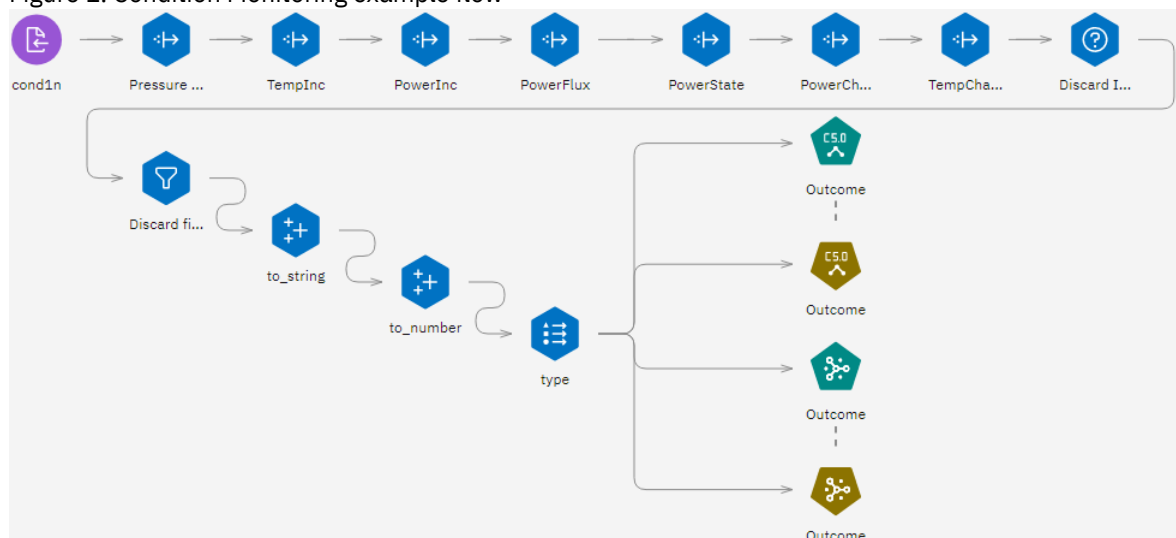
## Data preparation

Based on the results of exploring the data, the following flow derives the relevant data and learns to predict faults.

This example uses the flow named Condition Monitoring, available in the example project installed with the product. The data files are cond1n.csv and cond2n.csv.

1. On the My Projects screen, click Example Project.
2. Scroll down to the Modeler flows section, click View all, and select the Condition Monitoring flow.

Figure 1. Condition Monitoring example flow



The flow uses a number of Derive nodes to prepare the data for modeling.

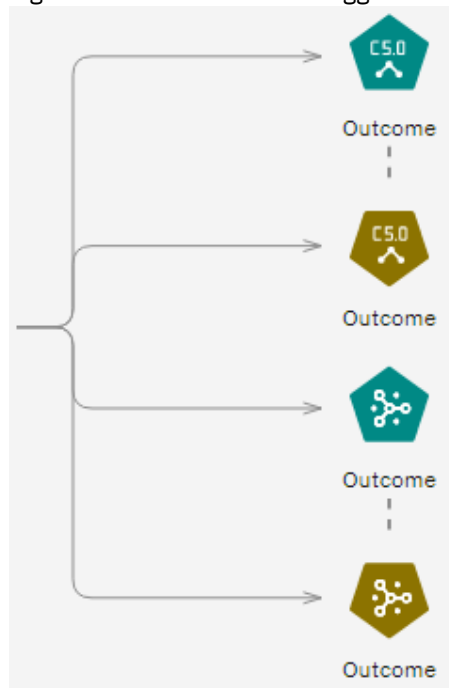
- Data Asset import node. Reads data file cond1n.csv.
- Pressure Warnings (Derive). Counts the number of momentary pressure warnings. Reset when time returns to 0.
- TempInc (Derive). Calculates momentary rate of temperature change using @DIFF1.
- PowerInc (Derive). Calculates momentary rate of power change using @DIFF1.
- PowerFlux (Derive). A flag, true if power varied in opposite directions in the last record and this one; that is, for a power peak or trough.
- PowerState (Derive). A state that starts as *Stable* and switches to *Fluctuating* when two successive power fluxes are detected. Switches back to *Stable* only when there hasn't been a power flux for five time intervals or when Time is reset.
- PowerChange (Derive). Average of *PowerInc* over the last five time intervals.
- TempChange (Derive). Average of *TempInc* over the last five time intervals.
- Discard Initial (Select). Discards the first record of each time series to avoid large (incorrect) jumps in *Power* and *Temperature* at boundaries.
- Discard fields (Filter). Cuts records down to *Uptime*, *Status*, *Outcome*, *Pressure Warnings*, *PowerState*, *PowerChange*, and *TempChange*.
- Type. Defines the role of *Outcome* as *Target* (the field to predict). In addition, defines the measurement level of *Outcome* as *Nominal*, *Pressure Warnings* as *Continuous*, and *PowerState* as *Flag*.

**Parent topic:** [Condition monitoring](#)

## Learning

Running the flow trains the C5.0 rule and neural network (net). The network may take some time to train, but training can be interrupted early to save a net that produces reasonable results. Once the learning is complete, model nuggets are generated: one represents the neural net and one represents the rule.

Figure 1. Generated model nuggets



These model nuggets enable us to test the system or export the results of the model. In this example, we will test the results of the model.

**Parent topic:** [Condition monitoring](#)

## Testing

Both of the generated model nuggets are connected to the Type node.

1. Reposition the nuggets as shown, so the Type node connects to the neural net nugget, which connects to the C5.0 nugget.
2. Attach an Analysis node to the C5.0 nugget.
3. Edit the Data Asset node to use the file cond2n.csv (instead of cond1n.csv), which contains unseen test data.
4. Right-click the Analysis node and select Run. Doing so yields figures reflecting the accuracy of the trained network and rule.

## Hotel satisfaction example for Text Analytics

The Text Analytics nodes offer powerful text analytic capabilities, which use advanced linguistic technologies and Natural Language Processing (NLP) to rapidly process a large variety of unstructured text data and, from this text, extract and organize the key concepts. Text Analytics can also group these concepts into categories.

In this example, a hotel manager is interested in learning what customers think about the hotel.

Figure 1. Chart of positive opinions

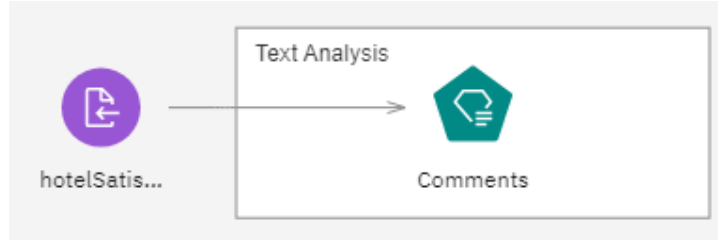


Figure 2. Chart of negative opinions



## Text Mining node

Figure 1. Text Mining node to analyze comments from hotel guests



1. Add a Data Asset node that points to hotelSatisfaction.csv.
2. From the Text Analytics category on the node palette, add a Text Mining node and double-click it to open its properties.
3. Under FIELDS, select `id` for the ID field and `Comments` for the Text field. Note that only the Text field is required.

Figure 2. Text Mining node properties

Fields	
ID field ⓘ	id
Text field ⓘ	Comments

4. Under MODEL, make sure Build interactively (category model nugget) is selected. Later when you run the node, this option will launch an interactive interface (known as the *Interactive Workbench*) in which you can extract concepts and patterns, explore and fine-tune the extracted results, build and refine categories, and build category model nuggets.
5. Select Exploring text link analysis (TLA) results. The default option Using extraction results to build categories extracts only concepts, whereas TLA extraction outputs both concepts and text links that are connections between topics (service, personnel, food, etc.) and opinions.
6. Under Copy Resources From, select Text analysis package, click Select Resources, and then load Hotel Satisfaction (English).tap (with `Current category set(s) = Topic + Opinion`).

A text analysis package (TAP) is a predefined set of libraries and advanced linguistic and nonlinguistic resources bundled with one or more sets of predefined categories. If no text analysis package is relevant for your application, you can instead start by selecting Resource template under Copy Resources From. A resource template is a predefined set of libraries and advanced linguistic and nonlinguistic resources that have been fine-tuned for a particular domain or usage.

Figure 3. Text Mining node properties

Build mode ⓘ

- ☐ Generate directly (concept model nugget)
- ☒ Build interactively (category model nugget)
  - ☐ Use session work (categories, TLA, resources, etc.) from last node update

Begin session by: ⓘ

- ☐ Using extraction results to build categories
- ☒ Exploring text link analysis (TLA) results

Copy Resources From ⓘ

- ☐ Resource template
- ☒ Text analysis package

Hotel Satisfaction (English)

7. Under EXPERT, select Accommodate spelling for a minimum word character length of. This option applies a fuzzy grouping technique that helps group commonly misspelled words or closely spelled words under one concept. The fuzzy grouping algorithm temporarily strips all vowels (except the first one) and strips double/triple consonants from extracted words and then compares them to see if they're the same (so, for example, `location` and `locatoin` are grouped together).

Figure 4. Text Mining node properties



Expert

Limit extraction to concepts with a global frequency count of at least ⓘ

1

☒ Accommodate punctuation errors

☒ Accommodate spelling for a minimum word character length of

Spelling limit ⓘ

5

☒ Extract uniterms

☒ Extract nonlinguistic entities

☒ Uppercase algorithm

☐ Group partial and full person names together when possible

Maximum nonfunction word permutation ⓘ

3

☒ Use derivation when grouping compound nouns

8. Click Save. Right-click the Text Mining node and run it to open the Interactive Workbench and proceed to the next section of this tutorial.

**Parent topic:** [Hotel satisfaction example for Text Analytics](#)

## Using the Interactive Workbench

The Interactive Workbench contains the extraction results and the category model contained in the Text Analytics Package.

Figure 1. Interactive Workbench

## Categories and concepts

### Category data

Build Extend Score Display Collapse all

<input type="checkbox"/> Category	Descriptors	Docs
<input type="checkbox"/> All	-	820
<input type="checkbox"/> Uncategorized	-	↻
<input type="checkbox"/> No concepts extracted	-	↻
<input type="checkbox"/> ▶ Hotel Amenities	104	↻
<input type="checkbox"/> ▶ General Satisfaction	34	↻
<input type="checkbox"/> ▶ Service	32	↻
<input type="checkbox"/> ▶ Location	11	↻
<input type="checkbox"/> ▶ Budget	10	↻

### Concepts data

Extract Display Concept

<input type="checkbox"/> Concept	In	Global	Docs	Type
<input type="checkbox"/> excellent	</>	167	147	<Positive>
<input type="checkbox"/> room	</>	148	144	<Room>
<input type="checkbox"/> good	</>	117	112	<Positive>
<input type="checkbox"/> staff		90	90	<Personnel>
<input type="checkbox"/> friendly	</>	76	76	<PositiveAttitude>
<input type="checkbox"/> hotel	</>	72	72	<Unknown>
<input type="checkbox"/> service	</>	60	60	<Services>
<input type="checkbox"/> clean	</>	53	53	<PositiveFeeling>
<input type="checkbox"/> no	</>	47	47	<NO>

## Category data

In the Categories pane you can build and manage your categories. This pane is located in the upper left corner of the Categories and concepts view. After extracting the concepts and types from your text data, you can begin building categories automatically using techniques such as concept inclusion, semantic network (in English only), or manually.

Since this example flow is built using a text analysis package (TAP), the category model is already populated.

Each time a category is created or updated, you can see whether any text matches a descriptor in a given category by clicking Score to score the documents or records. If a match is found, the document or record is assigned to that category. The end result is that most, if not all, of the documents or records are assigned to categories based on the descriptors in the categories.

Figure 2. Interactive Workbench - category data

Build	Extend	Score	Display	Collapse all		
<input type="checkbox"/> Category					Descriptors	Docs
<input type="checkbox"/> All					-	820
<input type="checkbox"/> Uncategorized					-	5
<input type="checkbox"/> No concepts extracted					-	0
<input type="checkbox"/> ▶ Hotel Amenities					104	492
<input type="checkbox"/> ▶ General Satisfaction					34	158
<input type="checkbox"/> ▶ Service					32	260
<input type="checkbox"/> ▶ Location					11	71
<input type="checkbox"/> ▶ Budget					10	91

You can expand each category sub-category, select them or a descriptor, and click Display to see the source data:

Figure 3. Interactive Workbench - category source data

Build	Extend	Score	Display	Collapse all		
<input type="checkbox"/> Category					Descriptors	Docs
<input type="checkbox"/> ▶ Hotel Amenities					104	492
<input type="checkbox"/> ▶ Comfort					36	235
<input type="checkbox"/> ▶ Cleanliness					15	118
<input checked="" type="checkbox"/> ▶ Neg					10	49
<input type="checkbox"/> </> [* smell * & <Negative>   <NegativeFeeling>   <Contextual> & !{no}]					-	8
<input checked="" type="checkbox"/> not cleaned					-	9
<input type="checkbox"/> musty					-	1
<input type="checkbox"/> dirty					-	14
<input type="checkbox"/> </> [deteriorated]					-	3

Extract	Display			Concept		
<input type="checkbox"/> Concept	In	Global	Docs	Type		
<input type="checkbox"/> excellent	</>	167	147	<Positive>		
<input type="checkbox"/> room	</>	148	144	<Room>		
<input type="checkbox"/> good	</>	117	112	<Positive>		
<input type="checkbox"/> staff	</>	90	90	<Personnel>		
<input type="checkbox"/> friendly	</>	76	76	<PositiveAttitude>		
<input type="checkbox"/> hotel	</>	72	72	<Unknown>		
<input type="checkbox"/> service	</>	60	60	<Services>		
<input type="checkbox"/> clean	</>	53	53	<PositiveFeeling>		
<input type="checkbox"/> no	</>	47	47	<NO>		

Rank	ID#	Comments	Categories
<input type="checkbox"/> 1	111	...Very overrated. Room could have been cleaned better before our arrival...	Hotel Amenities/Co... Hotel Amenities/Cl... General Satisfactio...
<input checked="" type="checkbox"/> 2	57	...Room was not cleaned daily...	Hotel Amenities/Co... Hotel Amenities/Cl...
<input type="checkbox"/> 3	159	...bathroom wasn't cleaned after previous guest. Checking in and out was a nightmare....	Hotel Amenities/Co... Hotel Amenities/Cl... Service/CheckInOu...
<input type="checkbox"/> 4	204	...Room not cleaned before check-in...	Hotel Amenities/Co... Hotel Amenities/Cl... Service/CheckInOu...
<input type="checkbox"/> 5	128...	...bathroom not cleaned after previous guest. Checking in was a nightmare....	Hotel Amenities/Co... Hotel Amenities/Cl... Service/CheckInOu...

Room was not cleaned daily
----------------------------

## Concepts data




During the extraction process, the text data is analyzed to identify interesting or relevant single words such as airport, location, or locatoin, and word phrases such as airport pick-up. These words and phrases are collectively referred to as *terms*. Using the linguistic resources, the relevant terms are extracted, and similar terms are grouped together under a lead term called a *concept*.

In this way, a concept might represent multiple underlying terms depending on your text and the set of linguistic resources you're using.

You can also use a Filter to select a subset of concepts:

Figure 4. Interactive Workbench - filtering concepts

Concepts data

Extract Display   Concept 

<input type="checkbox"/> Concept	Filter	Global	Docs	Type
<input type="checkbox"/> restaurant		17	17	<FoodPlaces>
<input type="checkbox"/> small	</>	17	17	<Contextual>

And you can select from different options:

Figure 5. Interactive Workbench - filter options

## Filter

Display results where

Global value exceeds ...and Docs exceed

0 0

Filter by type

☒ All types

☐ Selected types

Type

- ☐ Positive
- ☐ Unknown
- ☐ Room
- ☐ Negative
- ☐ PositiveFeeling
- ☐ Personnel
- ☐ PositiveAttitude
- ☐ RoomAmenities

---

Filter by match text

Type here

Match condition

Contains

Cancel Filter

If you want to remove the Filter and display all concepts, select Clear Filter:

Figure 6. Interactive Workbench - clear filter

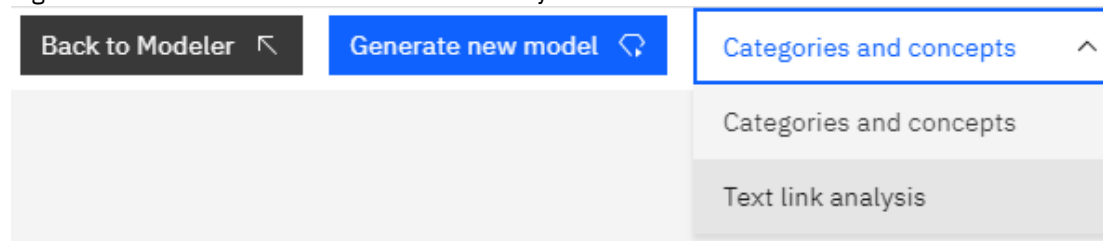
Concepts data

Extract	Display			Concept	
<input type="checkbox"/>	Concept	Clear Filter	Docs	Type	
<input type="checkbox"/>	excellent	</>	167	147	<Positive>
<input type="checkbox"/>	room	</>	148	144	<Room>
<input type="checkbox"/>	good	</>	117	112	<Positive>
<input type="checkbox"/>	staff		90	90	<Personnel>
<input type="checkbox"/>	friendly	</>	76	76	<PositiveAttitude>
<input type="checkbox"/>	hotel	</>	72	72	<Unknown>
<input type="checkbox"/>	service	</>	60	60	<Services>
<input type="checkbox"/>	clean	</>	53	53	<PositiveFeeling>
<input type="checkbox"/>	no	</>	47	47	<NO>

## Text link analysis

1. Access the text links from the Text link analysis pane:

Figure 7. Interactive Workbench - text link analysis



2. Select a Type Pattern from the top pane (for example, <Services> <Positive>) to display the corresponding Concepts Patterns in the bottom pane. To see the corresponding text, click Display. As the text in the Comments cell may be truncated, you can click on a cell to display the entire text in the Highlights cell.

Figure 8. Interactive Workbench - patterns

Type Patterns

Extract	Display				
<input type="checkbox"/>	Global	In	Type 1	Type 2	Type 3
<input type="checkbox"/>	124		<Unknown>	<Positive>	
<input type="checkbox"/>	75		<Personnel>	<PositiveAttitude>	
<input type="checkbox"/>	73		<Unknown>		
<input type="checkbox"/>	61		<Room>	<PositiveFeeling>	
<input checked="" type="checkbox"/>	58		<Services>	<Positive>	
<input type="checkbox"/>	43		<Room>	<Positive>	
<input type="checkbox"/>	43		<Unknown>	<PositiveFeeling>	
<input type="checkbox"/>	41		<Positive>		
<input type="checkbox"/>	38				

Concept Patterns

Display					
<input type="checkbox"/>	#	Global	In	Docs	Concept 1
<input checked="" type="checkbox"/>	1	20	</>	20	service
<input type="checkbox"/>	2	14	</>	14	service
<input type="checkbox"/>	3	6	</>	6	room service
<input type="checkbox"/>	4	2	</>	2	service
<input type="checkbox"/>	5	2	</>	2	service
<input type="checkbox"/>	6	1	</>	1	cleaning service
<input type="checkbox"/>	7	1	</>	1	housekeeping
<input type="checkbox"/>	8	1	</>	1	pick up service
<input type="checkbox"/>	9	1	</>	1	

Full Path

Rank	ID#	Comments
<input type="checkbox"/>	1	145 ...Great service...
<input type="checkbox"/>	2	429... ..Excellent Services...
<input type="checkbox"/>	3	429... ..Great service...
<input type="checkbox"/>	4	858... ..Outstanding services...
<input type="checkbox"/>	5	128... ..The service was great then...
<input type="checkbox"/>	6	92 ...The service is excellent...
<input type="checkbox"/>	7	146 ...Great services and amenities...

Docs

The service is excellent. Perfect location

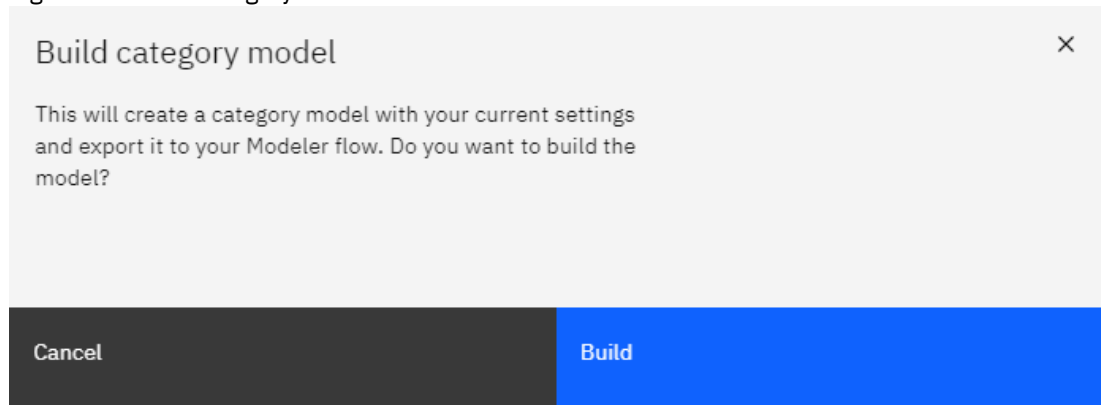
## Building and deploying the model

1. When your model is ready, click Generate new model to generate a text nugget.

Figure 1. Generate a new model

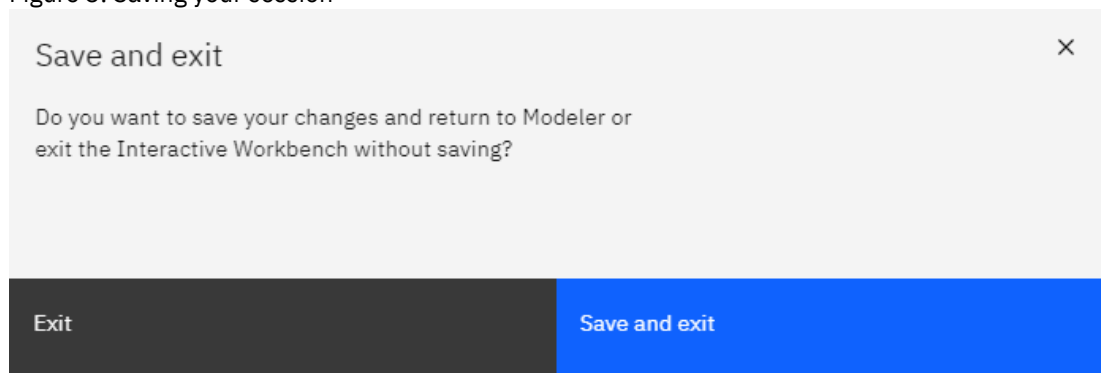


Figure 2. Build a category model



2. If you want to save the Interactive Workbench session, click Back to Modeler and then Save and exit.

Figure 3. Saving your session



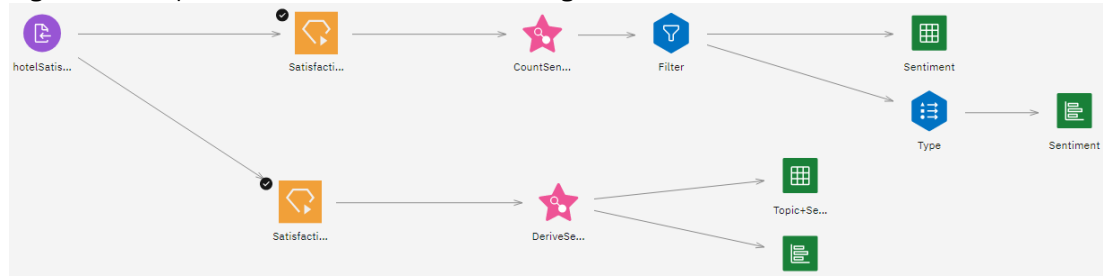
The generated text nugget appears on your flow canvas.

Figure 4. Generated text nugget



After the category model has been validated and generated in the Interactive Workbench, you can deploy it in your flow and score the same data set or score a new one.

Figure 5. Example flow with two modes for scoring



This example flow illustrates the two modes for scoring:

- Categories as fields. With this option, there are just as many output records as there were in the input. However, each record now contains one new field for every category that was selected on the Model tab. For each field, enter a flag value for true and for false, such as `True/False`, or `1/0`. In this flow, values are set to 1 and 0 to aggregate results and count the number of positive, negative, mixed (both positive and negative), or no score (no opinion) answers.

Figure 6. Model results - categories as fields

My Projects / Example Project with Text Analytics / HotelSatisfaction / Sentiment (8 fields, 820 records) ...							
id	Comments	Gender	Reason	Neg	Pos	Cont	Sentiment
1.000	"Rooms were clean."	"female"	"leisure"	0	1	0	positive
5.000	"Comfortable rooms, outstanding breakfast, nice service"	"male"	"business"	0	3	0	positive
9.000	"Beautiful location"	"female"	"business"	0	1	0	positive
13.000	"Friendly room service, attentive staff"	"female"	"leisure"	0	1	0	positive
17.000	"Very quiet, but very expensive"	"male"	"leisure"	1	1	0	mixed
21.000	"Service, quiet location, room facilities, but no parking facilities"	"male"	"leisure"	1	2	0	mixed
25.000	"I enjoyed the good food, the friendly staff, the room, the cleanliness, the view, the proximity to the city center, beach, sea, the transport, the accessibility"	"male"	"business"	0	3	0	positive
29.000	"Small room but cosy. Great restaurant."	"male"	"business"	1	2	0	mixed
33.000	"Price-performance ratio is right. Location is fantastic."	"female"	"leisure"	0	2	0	positive
37.000	"The rooms were newly renovated, quite spacious and clean. Prices are reasonable."	"female"	"leisure"	0	3	0	positive
41.000	"Highly recommended"	"male"	"business"	0	1	0	positive
45.000	"The rooms are extremely clean and spacious."	"female"	"business"	0	2	0	positive
49.000	"We immediately felt comfortable, like at home."	"male"	"business"	0	1	0	positive
53.000	"very quiet Hhotel"	"male"	"business"	0	1	0	positive

- Categories as records. With this option, a new record is created for each category, document pair. Typically, there are more records in the output than there were in the input. Along with the input fields, new fields are also added to the data depending on what kind of model it is.

Figure 7. Model results - categories as records

My Projects / Example Project with Text Analytics / HotelSatisfaction / Topic+Sentiment (6 fields, 1,256 ...					
id	Comments	Gender	Reason	Category	Sentiment
1.000	"Rooms were clean."	"female"	"leisure"	Hotel Amenities/Cleanliness	Pos
5.000	"Comfortable rooms, outstanding breakfast, nice service"	"male"	"business"	Hotel Amenities/Comfort	Pos
5.000	"Comfortable rooms, outstanding breakfast, nice service"	"male"	"business"	Hotel Amenities/Restaurant	Pos
5.000	"Comfortable rooms, outstanding breakfast, nice service"	"male"	"business"	Service/Competence	Pos
9.000	"Beautiful location"	"female"	"business"	Location	Pos
13.000	"Friendly room service, attentive staff"	"female"	"leisure"	Service/Attitude	Pos
17.000	"Very quiet, but very expensive"	"male"	"leisure"	Hotel Amenities/Quietness	Pos
17.000	"Very quiet, but very expensive"	"male"	"leisure"	Budget	Neg
21.000	"Service, quiet location, room facilities, but no parking facilities"	"male"	"leisure"	Hotel Amenities/Quietness	Pos
21.000	"Service, quiet location, room facilities, but no parking facilities"	"male"	"leisure"	Hotel Amenities/Car Park	Neg
21.000	"Service, quiet location, room facilities, but no parking facilities"	"male"	"leisure"	Location	Pos

- You can add a Select node after the DeriveSentiment SuperNode, include `Sentiments=Pos`, and add a Charts node to gain quick insight about what guests appreciate about the hotel:

Figure 8. Chart of positive opinions

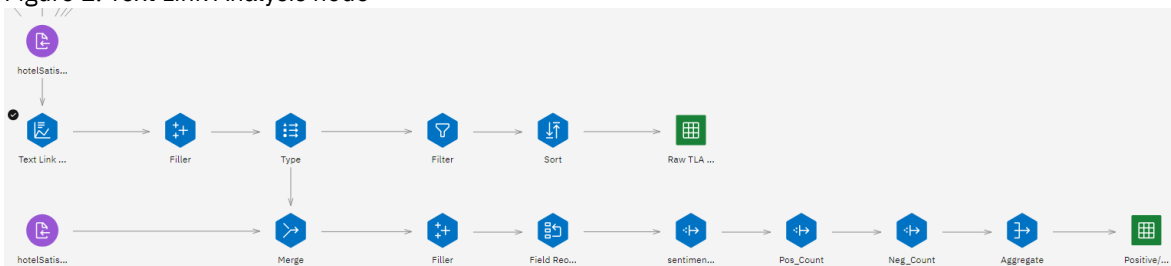


**Parent topic:** [Hotel satisfaction example for Text Analytics](#)

## Text Link Analysis node

In some cases, you may not need to create a category model to score. The Text Link Analysis (TLA) node adds a pattern-matching technology to text mining's concept extraction. This identifies relationships between the concepts in the text data based on known patterns. These relationships can describe how a customer feels about a product, which companies are doing business together, or even the relationships between genes or pharmaceutical agents.

Figure 1. Text Link Analysis node



1. Add a Text Link Analysis node to your canvas and connect it to the Data Asset node that points to hotelSatisfaction.csv. Double-click the node to open its properties.
2. Select `id` for the ID field and `Comments` for the Text field. Note that only the Text field is required.
3. For Copy resources from, select the Hotel Satisfaction (English) template.

Figure 2. Text Link Analysis node FIELD properties



Text Link Analysis

Fields

^

ID field

id

▼

Text field

Comments

▼

Language field

...

▼

Document Type

☒ Full Text

☐ Structured Text

Structured text formatting

Textual Unity

☒ Document Mode

☐ Paragraph Mode

Paragraph mode settings

Minimum

1

Maximum

300

Copy resources from

Hotel Satisfaction (English)

4. Under EXPERT, select Accommodate spelling for a minimum word character length of.

Figure 3. Text Link Analysis node EXPERT properties

Text Link Analysis

Fields

Expert

Limit extraction to concepts with a global frequency count of at least

1

☒ Accommodate punctuation errors

☒ Accommodate spelling for a minimum root character limit of

Spelling limit

5

☒ Extract uniterms

☒ Extract nonlinguistic entities

☒ Uppercase algorithm

☐ Group partial and full person names together when possible

Maximum nonfunction word permutation

3

☒ Use derivation when grouping compound nouns

The resulting output is a table (or the result of an Export node).

Figure 4. Raw TLA output

My Projects / Example Project with Text Analyt... / HotelSatisfaction / Raw TLA output (10 fields, 1,446...)

Concept1	Type1	Concept2	Type2	Concept3	Type3	Concept4	Type4	id	Matched Text
room	Room	clean	PositiveFeeling	Null	Null	Null	Null	1	<*Rooms were clean*>.
nothing	Uncertain	Null	Null	Null	Null	Null	Null	2	<*nothing*>
value for money	Budget	excellent	Positive	Null	Null	Null	Null	3	<*Excellent value for money*>
parking	HotelAmenities	too small	Negative	Null	Null	Null	Null	4	<*Parking too small*>
wifi	Internet	not free	NegativeBudget	room	Room	Null	Null	4	<*No free wifi in rooms*>
crib	RoomAmenities	no	NO	children	Unknown	Null	Null	4	<*No crib for our child*>.
room	Room	comfortable	PositiveFeeling	Null	Null	Null	Null	5	<*Comfortable rooms*,>, outstanding breakfast, nice service
service	Services	good	Positive	Null	Null	Null	Null	5	rooms, outstanding breakfast, <*nice service*>

Figure 5. Counting sentiments on a TLA node

My Projects / ... / HotelSatisfaction / Positive/Negative (4 fields, 820 r...

id	Comments	Pos_Count_Sum	Neg_Count_Sum
1	Rooms were clean.	1	0
2	nothing	0	0
3	Excellent value for money	1	0
4	Parking too small. No free wifi in rooms. No crib for our child.	0	3
5	Comfortable rooms, outstanding breakfast, nice service	3	0
6	Quiet location right on the beach.	1	0
7	Pleasant service	1	0

**Parent topic:** [Hotel satisfaction example for Text Analytics](#)

## Troubleshooting Watson Studio Desktop

Watson Studio Desktop provides log levels that help you troubleshoot problems.

- **Viewing or exporting the log files**  
You can view or export the Watson Studio Desktop log files.
- **Setting the log level at runtime**  
The log level that you specify defines the amount of diagnostic information that is provided by Watson Studio Desktop.

## Viewing or exporting the log files

You can view or export the Watson Studio Desktop log files.

### Access the log files from the Help menu

To view the log files, go to Help > Open Log Folder.

To export the log files, go to Help > Export Logs. The logs will be exported as a compressed file to your computer's desktop.

### Log files

By default, log files are stored in the following paths:

Windows

C:\Users\username\AppData\Roaming\IBM Watson Studio\logs\

macOS

/Users/username/Library/Application Support/IBM Watson Studio/logs/

**Parent topic:** [Troubleshooting Watson Studio Desktop](#)

## Setting the log level at runtime

The log level that you specify defines the amount of diagnostic information that is provided by Watson Studio Desktop.

Use an environment variable to set the log level. See the list of available log levels below.

- [IBM Watson Studio Subscription installation](#)
- [IBM Watson Studio Desktop 1.1 installation](#)
- [IBM Watson Studio Desktop 2.0 installation](#)

## IBM Watson Studio Subscription installation

---

### Windows

Enter the following command at the command prompt, where *loglevel* is the log level.

```
set LOG_LEVEL={loglevel}&&"C:\Program Files\IBM Watson Studio\IBM Watson Studio.exe"
```

### macOS

Open Terminal and enter the following command, where *loglevel* is the log level.

```
LOG_LEVEL={loglevel} open -a "/Applications/IBM Watson Studio.app"
```

## IBM Watson Studio Desktop 1.1 installation

---

### Windows

Enter the following command at the command prompt, where *loglevel* is the log level.

```
set LOG_LEVEL={loglevel}&&"C:\Program Files\IBM Watson Studio
Desktop\IBM Watson Studio
Desktop.exe"
```

### macOS

Open Terminal and enter the following command, where *loglevel* is the log level.

```
LOG_LEVEL={loglevel} open -a "/Applications/IBM Watson Studio
Desktop.app"
```

## IBM Watson Studio Desktop 2.0 installation

---

### Windows per-machine installation

Enter the following command at the command prompt, where *loglevel* is the log level.

```
set LOG_LEVEL={loglevel}&&"C:\Program Files\IBM Watson Studio
Desktop\IBM Watson Studio
Desktop.exe"
```

### Windows per-user installation

Enter the following command at the command prompt, where *loglevel* is the log level and *username* is the user's name.

```
set LOG_LEVEL={loglevel}&&"C:\Users\{username}\AppData\Local\Programs\IBM Watson
Studio
Desktop\IBM Watson Studio
Desktop.exe"
```

### macOS

Open Terminal and enter the following command, where *loglevel* is the log level.

```
LOG_LEVEL={loglevel} open -a "/Applications/IBM Watson Studio
Desktop.app"
```

## Available log levels

---

The following log levels are listed according to their diagnostic information. Each level includes the contents of the following log levels in descending order. If you select *info*, for example, the diagnostic information includes log levels *warn* and *error*. The log levels are not case-sensitive. You can select one of the following log levels:

- *all*

- `trace`
- `debug`
- `info` (This level is the default selection.)
- `warn`
- `error`

**Parent topic:** [Troubleshooting Watson Studio Desktop](#)

## Glossary

---

This glossary provides terms and definitions for Watson Studio deployment environments. (Not all terms and definitions apply to Watson Studio Desktop.)

### A

---

**algorithm**

Formula applied to data to determine optimal ways to solve analytical problems.

**analytical asset**

An asset that runs code to analyze data. See also *asset*.

**asset**

An item in a project or catalog that contains metadata about data or data analysis. See also *analytical asset*, *data asset*.

### B

---

**batch deployment**

A method to deploy models that processes input data from a file and writes the output to a file.

### C

---

**cleanse**

To ensure that all values in a data set are consistent and correctly recorded.

**collaborator**

A member of a group of people who are working together toward a common goal.

**confusion matrix**

A table that provides a detailed numeric breakdown of annotated document sets. The table is used to compare the annotations that were added by a machine learning model to the annotations in the ground truth. The table reports the number of false positives, false negatives, true positives, and true negatives.

**connected data asset**

A pointer to data that is accessed through a connection to an external data source.

**connection**

The information required to connect to a database. The actual information required varies according to the DBMS and connection method.

**connection asset**

An asset that contains information that enables connecting to a data source.

### D

---

**data asset**

An asset that points to data, for example, to an uploaded file. Connections and connected data assets are also considered data assets. See also *asset*.

**data mining**

The process of collecting critical business information from a data source, correlating the information, and uncovering associations, patterns, and trends. See also *predictive analytics*.

**Data Refinery flow**

A data source, a chain of one or more operations that refine and shape that data source, and a target that the data moves to.

Data Refinery flow asset

An asset that is based on an ordered set of steps to cleanse, shape, and enhance data.

data science

The analysis and visualization of structured and unstructured data to discover insights and knowledge.

data set

A collection of data, usually in the form of rows (records) and columns (fields) and contained in a file or database table.

data source

A repository, queue, or feed for reading data, such as a Db2 database or IBM MQ.

data table

A collection of data, usually in the form of rows (records) and columns (fields) and contained in a table.

deployment

A model or application package that is available for use.

deployment space

A workspace where models are deployed and deployments managed.

## E

---

endpoint URL

A network destination address that identifies resources, such as services and objects. For example, an endpoint URL is used to identify the location of a model or function deployment when a user sends payload data to the deployment.

## F

---

field ops node

A node in an SPSS Modeler flow that performs operations on data fields, such as filtering, deriving new fields, and determining the measurement level for given fields.

flow

A collection of nodes that define a set of steps for processing data or training a model.

flow canvas

See *flow editor*.

flow editor

A tool for creating flows.

## G

---

graphical canvas

A tool for creating analytical assets by visually coding. A canvas is an area on which to place objects or nodes that can be connected to create a flow.

graph node

A node in an SPSS Modeler flow that displays data before or after modeling. Some common graphs include plots, histograms, web nodes, and evaluation charts.

## H

---

HPO

See *hyperparameter optimization*.

hyperparameter

In machine learning, a parameter whose value is set before training as a way to increase model accuracy.

hyperparameter optimization (HPO)

The process for setting hyperparameter values to the settings that provide the most accurate model.

## I

---

import node

A node that pulls data into an SPSS Modeler flow, and always appears at the start of the flow.

## J

---

Jupyter notebook

See *notebook*.

Jupyter notebook editor

The standard notebook editor in a project. Locks the notebook during editing to prevent conflicts.

## L

---

## M

---

machine learning framework

The libraries and runtime for training and deploying a model.

model

In a machine learning context, a set of functions and algorithms that have been trained and tested on a data set to provide predictions or decisions

modeler flow asset

An asset that is based on a graphical representation of a data model or a neural network design.

modeling node

A node in SPSS Modeler that represents a statistical algorithm such as neural nets, decision trees, clustering algorithms, and data sequencing.

## N

---

node

The graphical representation of a data operation in a stream or flow. Different types of nodes have different shapes to indicate the type of operation that they perform.

notebook

An interactive document that contains executable code, descriptive text for that code, and the results of any code that is run.

notebook asset

An asset that is based on a Jupyter notebook file.

notebook kernel

The part of the Jupyter notebook editor that executes code and returns the computational results.

## O

---

online deployment

Method of accessing a deployment through an API endpoint, providing a real-time score or solution on new data.

output node

A node in an SPSS Modeler flow that produces various choices of output for data, charts, and model results. Output nodes usually appear as the last node in a flow or a branch of a flow.

## P

---

payload

The data passed to a deployment to get back a score, prediction, or solution.

placeholder

A field or variable to be replaced with a value.

predictive analytics

A business process and a set of related technologies that are concerned with the prediction of future possibilities and trends. Predictive analytics applies such diverse disciplines as probability, statistics, machine learning, and artificial intelligence to business problems to find the best action for a given situation. See also *data mining*.

project

A workspace to organize resources and collaboratively work on data.

Python

A programming language used in data science and AI.

## R

---

R

An extensible scripting language used in data science and AI that offers a wide variety of analytic, statistical, and graphical functions and techniques.

record ops node

A node in an SPSS Modeler flow that performs operations on data records, such as selecting, merging, and appending.

refine

To cleanse and shape data.

## S

---

scoring

1. The process of computing how closely the attributes for an incoming identity match the attributes of an existing entity.
2. In machine learning, measures the confidence of a predicted outcome.

shape

To customize data by filtering, sorting, removing columns; joining tables; performing operations that include calculations, data groupings, hierarchies and more.

SPSS Modeler

A tool for creating flows that build and train predictive models.

SQL pushback

In SPSS Modeler, the process of performing many data preparation and mining operations directly in the database through SQL code.

supernode

An SPSS Modeler node that shrinks a data stream by encapsulating several nodes into one.

## T

---

text classification

A type of model that automatically identifies and classifies text into specified categories.

trained model

A model that is ready to be deployed.

training

The initial stage of model building, involving a subset of the source data. The model can then be tested against a further, different subset for which the outcome is already known.

## U

---

unstructured data

Any data that is stored in an unstructured format rather than in fixed fields. Data in a word processing document is an example of unstructured data.

## V

---



visualization

A graph, chart, plot, table, map, or any other visual representation of data.