Watson Machine Learning Server 2.0.0-1 User Guide

Note

Before using this information and the product that it supports, read the information in "Notices" on page 11.

Tables of Contents

Watson Machine Learning Server documentation

Welcome to the documentation for Watson Machine Learning Server, a new, single-node server that is part of the IBM Watson Studio family of products.

Take advantage of scalable computational resources and deployment management services when you deploy analytical assets such as SPSS Modeler flows, AutoAI experiments, and machine learning model notebooks from Watson Studio Desktop to the Watson Machine Learning server.

Getting started

- Product overview
- Installation
- What's new?

Common tasks

- Building AutoAI models
- Deploying models

Troubleshooting and support

- Watson Studio Desktop support
- Stack overflow

Other deployment environments and components

- Watson Studio Desktop
- Cloud Pak for Data as a Service
- Cloud Pak for Data
- Deployment environments feature matrix

Overview

IBM Watson Machine Learning Server is a new, single-node server that is part of the IBM Watson Studio family of products.

Analytical assets such as SPSS Modeler flows, AutoAI experiments, and machine learning model notebooks from Watson Studio Desktop can be deployed to the Watson Machine Learning server to get the advantage of scalable computational resources and deployment management services.

Benefits include:

- Simple installation experience of a virtual machine image without the need for Kubernetes or Linux skills
- Seamless user experience for scaling workloads created in Watson Studio Desktop to Watson Machine Learning Server
- Seamless user experience for deploying analytical assets created in Watson Studio Desktop to Watson Machine Learning Server

- No CPU limitation — leverage all available CPUs for more deployments, more training jobs, and more workloads

Services provided

IBM Watson Machine Learning Server services, based on Watson Machine Learning APIs, version 4:

- Repository Service
- Deployment Service
- SPSS and other runtimes

SPSS Modeler service:

- User interface to scale or deploy the SPSS asset on Watson Machine Learning Server. This enables enables the remote execution from SPSS flows to run on Watson Machine Learning Server and allows for scoring SPSS models deployed to the server.

What's new in Watson Machine Learning Server

These features are new in recent releases of Watson Machine Learning Server.

What's new in Watson Machine Learning Server 2.0.0-1

These features are new in Watson Machine Learning Server 2.0.0-1

Security update for Python

Due to a security vulnerability, Python 3.6 is deprecated in favor of Python 3.7.

For notebooks, Python 3.6 is discontinued. You must install Watson Studio Desktop 2.0.0-1 in order to continue to run notebooks.

On Watson Machine Learning Server, Python 3.6 frameworks are deprecated and some are discontinued. If a framework is deprecated, you cannot train new models but you can run existing deployments. If a framework is discontinued, you must retrain and redeploy your asset. For details, see Supported frameworks

PSIRT security fixes

When security issues or vulnerabilities are discovered in IBM Watson Machine Learning Server, the IBM Product Security Incident Response Team (PSIRT) analyzes the issue and applies a fix automatically. The following Common Vulnerabilities and Exposures (CVEs) were addressed in this release.

CVE ID	Description
CVE-2020-11612	The ZlibDecoders in Netty 4.1.x before 4.1.46 allow for unbounded memory allocation while decoding a ZlibEncoded byte stream. An attacker could send a large ZlibEncoded byte stream to the Netty server, forcing the server to allocate all of its free memory to a single decoder.

CVE ID	Description
CVE-2020-14039	In Go before 1.13.13 and 1.14.x before 1.14.5, `Certificate.Verify` may lack a check on the `VerifyOptions.KeyUsages` EKU requirements (if `VerifyOptions.Roots` equals nil and the installation is on Windows). Thus, X.509 certificate verification is incomplete.
CVE-2020-15196	TensorFlow is vulnerable to a heap-based buffer overflow, caused by improper bounds checking by the SparseCountSparseOutput and RaggedCountSparseOutput implementations implementations. By sending a specially-crafted request, a remote authenticated attacker could overflow a buffer and execute arbitrary code on the system.
CVE-2020-15213	TensorFlow is vulnerable to a denial of service, caused by an out of memory allocation in the TFLite implementation of segment sum. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-15192	TensorFlow is vulnerable to a denial of service, caused by a memory leak when passing a list of strings to dlpack.to_dlpack. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-15209	TensorFlow is vulnerable to a denial of service, caused by a NULL pointer dereference flaw in TFLite. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-15205	TensorFlow could allow a remote attacker to obtain sensitive information, caused by a heap-based buffer overflow in the data_splits argument of tf.raw_ops.StringNGrams. By sending a specially-crafted request, an attacker could exploit this vulnerability to obtain contents of the memory, and use this information to launch further attacks against the affected system.
CVE-2020-15191	TensorFlow is vulnerable to a denial of service, caused by improper input validation by the dlpack.to_dlpack function. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-15208	TensorFlow could allow a remote attacker to bypass security restrictions, caused by a data corruption flaw when a dimension mismatch occurs in TFLite. By sending a specially-crafted request, an attacker could exploit this vulnerability to read and write outside of bounds of memory.

CVE ID	Description
CVE-2020-15204	TensorFlow is vulnerable to a denial of service, caused by a NULL pointer dereference flaw when calling `tf.raw_ops.GetSessionHandle` or `tf.raw_ops.GetSessionHandleV2` functions in eager mode. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-15200	TensorFlow is vulnerable to a denial of service, caused by a improper input validation by the `RaggedCountSparseOutput` implementation. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-15195	TensorFlow is vulnerable to a heap-based buffer overflow, caused by improper bounds checking by the SparseFillEmptyRowsGrad implementation. By sending a specially-crafted request, a remote authenticated attacker could overflow a buffer and execute arbitrary code on the system.
CVE-2020-15212	TensorFlow is vulnerable to a denial of service, caused by an out-of-bounds access flaw in the TFLite implementation of segment sum. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a segmentation fault and memory corruption.
CVE-2020-15199	TensorFlow is vulnerable to a denial of service, caused by improper input validation in the RaggedCountSparseOutput function. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-15194	TensorFlow is vulnerable to a denial of service, caused by improper input validation by the `SparseFillEmptyRowsGrad` implementation. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-15211	TensorFlow could allow a remote attacker to bypass security restrictions, caused by an out-of-bounds access flaw in the TFLite operators. By sending a specially-crafted request, an attacker could exploit this vulnerability to read and write from outside the bounds of heap allocated arrays.
CVE-2020-15207	TensorFlow is vulnerable to a denial of service, caused by a segmentation fault data corruption flaw when using negative indexing in TFLite. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition.

CVE ID	Description
CVE-2020-15203	TensorFlow is vulnerable to a denial of service, caused by improper input validation by the fill argument in `tf.strings.as_string`. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-15206	TensorFlow is vulnerable to a denial of service, caused by a flaw when changing the `SavedModel` protocol buffer and altering the name of required keys. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause segmentation fault and data corruption.
CVE-2020-15202	TensorFlow is vulnerable to a denial of service, caused by an integer truncation in Shard API. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a segmentation fault, read or write outside of heap allocated arrays, stack overflows, or data corruption.
CVE-2020-15197	TensorFlow is vulnerable to a denial of service, caused by improper input validation by the SparseCountSparseOutput implementation. By sending a specially-crafted request, a remote authenticated attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-15214	TensorFlow is vulnerable to a denial of service, caused by an out-of-bounds write flaw in the TFLite implementation of segment sum. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition and memory corruption.
CVE-2020-15193	TensorFlow is vulnerable to a denial of service, caused by a memory corruption in the implementation of dlpack.to_dlpack. By sending a specially-crafted request, a remote authenticated attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-15210	TensorFlow is vulnerable to a denial of service, caused by a segmentation fault and data corruption flaw when using an invalid TFLite model. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-15586	Go before 1.13.13 and 1.14.x before 1.14.5 has a data race in some net/http servers, as demonstrated by the `httputil.ReverseProxy` Handler, because it reads a request body and writes a response at the same time.

CVE ID	Description
CVE-2020-15201	TensorFlow is vulnerable to a heap-based buffer overflow, caused by improper bounds checking by the `RaggedCountSparseOutput` implementation. By sending a specially-crafted request, a remote attacker could overflow a buffer and execute arbitrary code on the system.
CVE-2020-15190	TensorFlow is vulnerable to a denial of service, caused by improper input validation by the `tf.raw_ops.Switch` operation in eager mode. By sending a specially-crafted request, a remote attacker could exploit this vulnerability to cause a denial of service condition.
CVE-2020-7238	Netty 4.1.43.Final allows HTTP Request Smuggling because it mishandles Transfer-Encoding whitespace (such as a `[space]Transfer-Encoding:chunked` line) and a later Content-Length header. This issue exists because of an incomplete fix for CVE-2019-16869.

What's new in Watson Machine Learning Server (2.0)

New features make it easier for you to install and configure the Watson Machine Learning Server and provides support for new features for users.

Guided installation

The default for the installation script prompts you for information to guide you through the installation experience.

Automatic generation of a self-signed certificate

If you do not already have a self-signed certificate, you can request one as part of the guided installation process.

Advanced installation option

If you prefer a command line installaton, an advanced option for the installation script supports entering all flags and arguments for the various parameters required for installation in a single command.

Using the advanced mode, you can also specify the port range (of 50 ports) and percentage of total CPU and memory resources to be allocated to the server.

View Watson Machine Learning Server installation text in your language

The language in installation text is determined by the locale of the operating system. If the operating system is not in one of the translated languages, the default language is English.

You can view the installation script in the following languages:

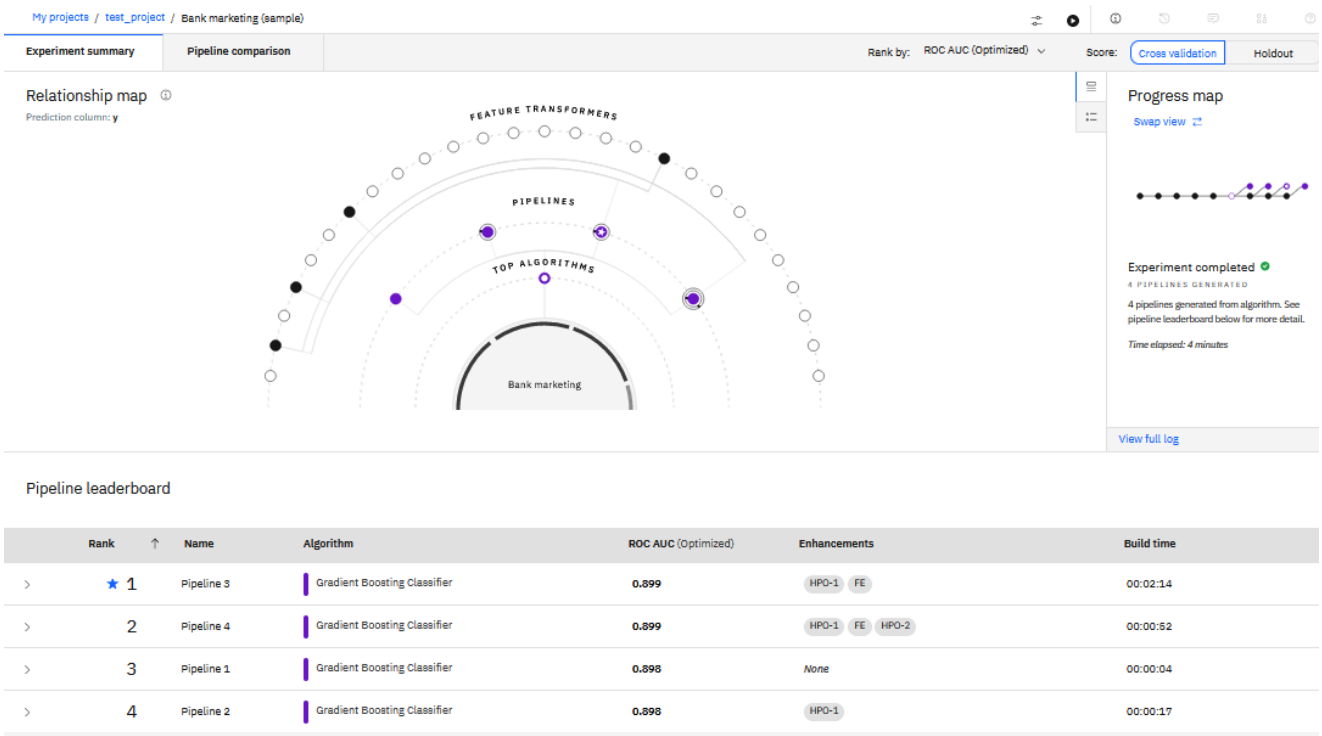
- Simplified Chinese
- Traditional Chinese
- English

- French
- German
- Italian
- Japanese
- Brazilian Portuguese
- Russian
- Spanish

Note: the `help` option for the installation script displays in English only.

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 dataset and discovers data transformations, algorithms, and parameter settings that work best for your problem setting.



Save a pipeline as a model and deploy it to generate predictions.

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 batch jobs.

Support for more types of input data sources for deployments

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/export deployment spaces

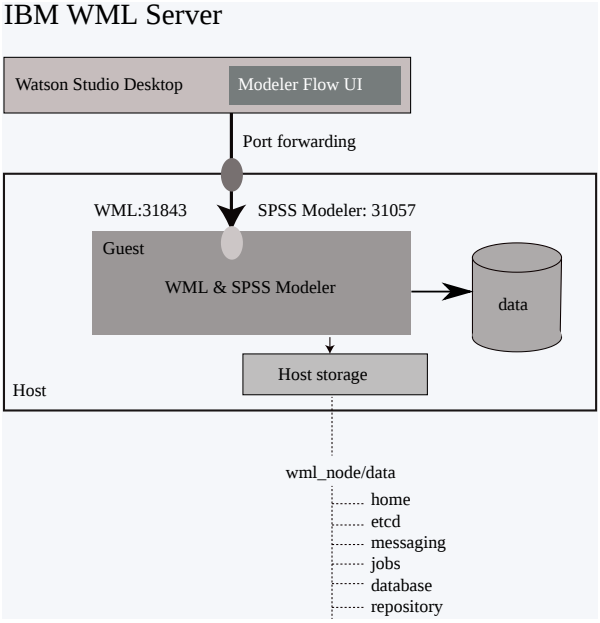
With Watson Machine Learning Server, you can export the contents of a deployment space to a file. You can also create a new space by importing a space from a file. This provides an efficient way to archive a space, use a space as a template, or share a space. See Deployment spaces.

Import/export assets from projects

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.

System architecture

This graphic represents the Watson Machine Learning Server architecture.



Components and terminology

- Vagrant is a command line utility for managing the lifecycle of virtual machines.
- Vagrant Box is a package format for Vagrant environments.
- A bare metal environment is a computer system or network where a virtual machine is installed directly on hardware rather than within the host operating system (OS).
- Virtual machine is an image file that behaves like an actual computer.
- Host is the bare metal machine/server/node and the operating system you install on the server
- Guest is the operating system you install on a virtual machine (VM) that is created on the Host machine
- Host Storage is the physical storage on the Host machine which would be shared with Guest machine

Known issues

These are the known issues for this release of IBM Watson Machine Learning Server.

Known issues for Watson Machine Learning Server 2.0.0-1

Missing assets or loading errors after deployment

If you notice that assets are missing or if you notice errors for loading jobs, functions, or models after you install and deploy Watson Machine Learning Server, run this script.

1. Verify wmlserver status is in ready state using `./wmlserver status`
2. Log into Watson Machine Learning Server virtual machine (VM) by running these steps:

```
cd <wmlserver_package directory>
export VAGRANT_HOME= <vagrant_home_path_as_per_wmlserver_status>
vagrant ssh
sudo su -
```

1. Run the script to create missing assets: `sh /vagrant/asset_recreate.sh`
2. Log out of the Watson Machine Learning Server VM.
3. Verify Watson Machine Learning Server status is in ready state and resume use of the server.

Different behavior depending on how you delete a deployment

Depending on whether you delete a deployment from the UI or using the Python client, the result is different regarding how jobs are handled.

- If you delete a deployment from the UI, deployment jobs persist and you can see them from the Jobs page.
- If you delete a deployment using the Python client, associated jobs are also deleted.

Server fails to start after a re-install

If Watson Machine Learning Server fails to start after a re-installation, the cause can stem from the Vagrant domain needing clean-up. In addition to being unable to start the server, you might see errors trying to remove the server. For example: `'vagrant destroy default -f'... Failed! Failed to delete WML Server`
`exit status 1.`

In this case, clean up the Vagrant domain like this.

1. Check the domains.
   `virsh list --all`

2. Undefine the problem domain.
   `virsh undefine <domain-name>`

Output: `Domain <domain-name> has been undefined`

Following the clean-up of the Vagrant domain, you should be able to remove, then re-install Watson Machine Learning Server.

Association to old deployment space maintained after restore

If you back up Watson Machine Learning Server and then restore to a new server, associations to existing spaces on the original server persist after the restore, as follows:

- Existing Watson Studio Desktop project to space associations are not updated as part of the back-up and restore process. Any project that was linked to a space on a specific Watson Machine Learning Server will still be linked to the same space on that server (it will not automatically change to point to the restored space on the different server).

- The restored deployment space will still appear to be linked to the original desktop project, even though that relationship will not be reflected from the project back to the space (the project is still pointing to the original space on the backed-up server). You will be unable to associate another project with this restored space unless you manually remove the project tag associated with the space via a `PATCH /v4/spaces` API request.

Associated project does not display in the spaces list

Associated spaces are displayed in the Spaces list for the first 100 spaces. If your list of spaces exceeds 100, associated spaces will fail to display.

Scheduled jobs frozen after restore

If you back up and restore Watson Machine Learning Server, your scheduled jobs will be listed but will no longer run on the specified schedule. You can restart the job by editing the job details and updating the schedule.

Unable to restart Watson Machine Learning Server after stopping

If you stop the Watson Machine Learning Server and then are unable to restart it, try restarting the virtual machine (VM) and then restart the server.

AutoAI binary classification training fails with larger data sets

If your AutoAI training experiment fails with an insufficient memory error, choose a larger Compute size and try again.

Unable to update an AutoAI asset with the small deployment configuration

If you create a deployment for any model or function or script and select the small (S) compute configuration option, you will be unable to update the deployed asset to an AutoAI model. To work around this limitation, either patch the hardware spec to higher than “S” before you update the asset, or create a new deployment with a larger compute configuration.

Promoting a large file to a space can result in a time-out error

Promoting large files to a deployment space can fail with a 502 error. The max size of a file you can add or promote to a deployment space depends on your internet capacity and connection speed with the Watson Machine Learning Server.

Deployment

Delete assets from a space before deleting the space

If you delete a deployment space that contains models, those models (and any other assets in the deployment space) will also be deleted. But note that if the deployment space contains deployments, they aren’t deleted. Either delete the deployments before you delete the deployment space, or delete orphaned deployments using the Python client. This will be fixed in a future release.

Saving a Spark model to Watson Machine Learning fails on Windows

Using the Python command `client.repository.store_model()` to save a Spark model to the Watson Machine Learning repository results in an error on Windows. This limitation will be addressed in a future release.

Notices

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution 1.0 Generic license.

- spdx-exceptions-ids v3.0.0
- pandoc v2.2.3.2

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution 2.0 Generic license:

- scikit-learn
- pillow (MSPIMAGEPLUGIN.PY)
- Tensorflow v1.14.0 (IMAGE_RETRAINING.MD)
- scikit-learn v0.21.1

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution 2.5 Generic license:

- gevent v1.4.0
- checker-qual v2.0.0
- iText v2.1.2
- JSR305 v1.3.9
- spyder v3.1.4
- bazel v0.5.2
- JCIP-ANNOTATIONS (HttpCore)

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution 3.0 Generic license:

- Font-Awesome v4.7.0
- pandoc v2.2.3.2
- Gdal v2.3.3
- pip v19.0.1
- pip v19.1.1
- pbr v5.1.3
- Tornado v5.1.1
- Tornado v4.3
- SPDX-EXCEPTIONS v 2.1.0
- spdx-exceptions v2.2.0
- spdx-expression-parse v3.0.0
- JSON3 v3.3.2
- Atob v2.1.2
- xterm v3.13.2
- leaflet-providers v1.1.17
- spyder v3.3.3
- rbokeh v0.6.3
- net
- nltk v3.4.5
- spyder v3.1.4
- OpenCV v3.4.2
- distributed v2.5.2
- Snappy v1.1.3

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution 4.0 Generic license:

- Ipython v7.6.1
- Ipython v7.2.0
- Typescript v3.5.3
- CANIUSE-DB
- caniuse-db v1.0.30000840
- caniuse-db v1.0.30000839
- caniuse-db v1.0.30000833
- CANIUSE-LITE v1.0.30001012
- caniuse-lite v1.0.30000835
- opencpu-server v2.0.8
- z-schema v4.2.2
- IBM-DESIGN-COLORS v1.8.0
- IBM-DESIGN-COLORS v2.1.3
- mlxtend v0.17.0
- future v0.17.1
- future v0.18.0
- Keras v2.3.1
- Keras v2.1.6
- Keras Version v2.2
- Ipython v7.9.0
- gevent v1.4.0
- Tensorflow v1.14.0 (SPEECH COMMANDSDATASET)
- mlxtend v0.17.0

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution Share Alike 1.0 Generic license:

- TEST-TO_LATEX.R
- TO_LATEX.TEX
- text (DEED.RU-TESTTEXT)
- text (HEBREW-TESTTEXT)

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution Share Alike 2.0 Generic license:

- eslint v4.3.0
- text (KR-TESTTEXT)
- text (JP-TESTTEXT)
- pillow v6.2.0

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution Share Alike 2.5 Generic license:

- Tensorflow v1.13.1 (RNN COLORBOT)
- npm v6.13.4
- postcss-reduce-initial v4.0.3
- chardet v3.0.4
- Tensorflow v1.14.0 (RNN COLORBOT)
- Jersey-Json
- text (CN-TESTTEXT)

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution ShareAlike 3.0 Generic license:

- pyzmq v17.1.2
- pyzmq v16.0.2
- jquery-ui v1.9.0
- rbokeh v0.6.3
- krb5 v1.16.1
- hmisc v4.2-0
- Tensorflow v 1.13.1 (MNIST DATASET)
- tk v8.6.8
- posix-character-classes v0.1.1
- leaflet-providers v1.1.17
- spdx-exceptions v1.0.4
- spdx-exceptions v2.1.0
- networkx v2.3
- NumPy v1.17.2
- NumPy v1.16.4
- text (TH-TESTTEXT)
- text (DEED.AR-TESTTEXT)
- text (GR-TESTTEXT)
- text (VIETNAMESE-TESTTEXT)
- readline v7.0
- dracut

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution Share Alike 4.0 Generic license:

- shortid v2.2.14
- Glob v7.1.4
- Glob v7.1.6
- highlight.js v9.12.0
- pylint v2.4.3
- spyder v3.1.4

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution Noncommercial 4.0 International license:

- redux-promise-middleware (Documentation)

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution Noncommercial ShareAlike 2.5 Generic license:

- nltk v3.4
- jsr305 v1.3.9

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution Noncommercial Noderivs 2.0 Generic license:

- gst-plugins-base v1.14.0

The Offering includes some or all of the following that IBM provides under the Creative Commons Attribution Noncommercial Noderivs 2.5 Generic license:

- nltk v3.2.3

The Offering includes some or all of the following that IBM provides under the SIL Open Font License 1.1:

- AMSFONTS
- FONTS (harfbuzz)
- FONT AWESOME 4.2.0 (arrow)
- IBM PLEX TYPEFACE (carbon-components)
- FONT AWESOME (nbconvert)
- FONTAWESOME-FONTS
- HELVETICA-NEUE
- READLINE.PS (Readline)
- FONT-AWESOME (Notebook)
- FONTAWESOME
- FONT AWESOME FONTS
- FONTAWESOME (FONT) (JupyterLab)
- Font-Awesome v4.6.3
- Font-Awesome v4.3.0
- Font-Awesome v4.7.0
- nbconvert v5.2.1
- nbconvertv5.1.1
- qtawesome v3.3.0
- handsontable v0.25.1
- RLUSERMAN.PS (Readline)
- FONT AWESOME - FONT (bazel)
- NOTO-FONTS (pillow)
- CMMI9 (libtasn1)
- FONT AWESOME (Apache ORC)
- READLINE.PS (Readline)
- FONTAWESOME (Tables)
- QTAWESOME-FONTS (qtawesome)

The Offering includes some or all of the following that IBM provides under the UBUNTU FONT LICENCE Version 1.0:

- Font_license (Werkzeug)

Accessibility features in Watson Machine Learning Server documentation

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

IBM Watson Machine Learning 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 IBM Watson Machine Learning 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 IBM Watson Machine Learning, you can do 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 on using assistive technologies with HTML-based information.
- Use screen magnifiers to magnify what is 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 websites, 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 IBM Watson Machine Learning user interfaces do not have content that flashes 2 - 55 times per second.

The IBM Watson Machine Learning 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.

Securing your data

For PID(s): 5737-L65

Notice:

This document is intended to help you in your preparations for GDPR readiness. It provides information about features of IBM Watson Machine Learning Server 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 clients' 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

GDPR

General Data Protection Regulation (GDPR) has been adopted by the European Union (“EU”) and applies 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

- GDPR.eu
- ibm.com/GDPR website

Product Configuration - considerations for GDPR Readiness

Requirement: To help with GDPR readiness, install the latest version of Watson Machine Learning Server.

Offering Configuration

The following sections provide considerations for configuring IBM Watson Machine Learning Server to help your organization with GDPR readiness.

Configuration to support Data Security

To ensure that your data in IBM Watson Machine Learning Server is stored securely, you can encrypt your storage partition. If you use Linux Unified Key Setup-on-disk-format (LUKS) for this purpose, then you must enable LUKS and format the partition with XFS before you install IBM Watson Machine Learning Server. Data Life Cycle

What is the end-to-end process through which personal data go through when using our offering?

For each intended user of your IBM Watson Machine Learning Server, it is required to set up user accounts by creating user IDs and passwords to allow users access to IBM Watson Machine Learning Server. After the user account is set up, IBM Watson Machine Learning Server uses that account to authenticate the user and to determine the type of data and operations that the user can access and interact with.

GDPR recommendations align with the required method to configure access to creating users within an LDAP enterprise directory server. LDAP supports the management of user IDs and passwords at an enterprise level instead of managing this data in IBM Watson Machine Learning Server. IBM Watson Machine Learning Server calls on the LDAP server to provide authentication, which means that IBM Watson Machine Learning Server does not store passwords for individual users. See Manage users and Security in IBM Watson Machine Learning Server to learn more. See User authentication for REST API authentication. See Audit IBM Watson Machine Learning Server records to learn more about access failures recorded in the logs. Authentication to data sources

All data source credentials are encrypted with a global key or a user-specific key, and can only be used inside the IBM Watson Machine Learning Server cluster where the password was originally entered. These encrypted

credentials become unusable if they are pushed to a Git repository or exported outside of a IBM Watson Machine Learning Server cluster using the project or asset export process.

Personal data used for online contact with IBM

IBM Watson Machine Learning Server clients can submit online comments/feedback/requests to contact IBM about IBM Watson Machine Learning Server subjects in a variety of ways, primarily:

- Public comments area on pages of IBM Watson Machine Learning Server documentation in IBM Knowledge Center
- Public comments in the IBM Watson Machine Learning Server space of dWAnswers
- Feedback forms in the IBM Watson Machine Learning Server community

Typically, only the client name and email address are used, to enable personal replies for the subject of the contact, and the use of personal data conforms to the IBM Online Privacy Statement. Data Storage

Ensure your disk volumes and backups of the disk volumes are encrypted. Establish appropriate access controls through the User Management panel. Access failures are recorded in the logs. See Audit IBM Watson Machine Learning Server records for details. Additional considerations

Data Access

Who can access data in your offering?

Only project collaborators can access local data sets and remote data sources. By default, each member of the project needs to have access to the remote data source.

Additional considerations

Multiple log entries from different services are recorded and periodically purged. A IBM Watson Machine Learning Server administrator can configure the log retention period in the Settings.

Data Processing

Disk volumes should be encrypted. To encrypt data storage, LUKS is recomended.

You should only share permalink URLs or REST API endpoints with authorized individuals.

Your configuration of user access privileges determines who can delete personal data. Administrators are given access to delete content from IBM Watson Machine Learning Server, and delete users from the integrated LDAP directory server.

In addition, users who have been granted sufficient privileges can delete content that they have created.

Data inside projects such as CSV files can be deleted individually or by the deletion of the entire project.

Ensure all data repositories are tracked.

Data Monitoring

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

Access failures are recorded in the logs. See Audit IBM Watson Machine Learning Server records for details. Responding to Data Subject Rights

When the IBM Watson Machine Learning Server administrator deletes, updates, or restricts the use of the data, the administrator must also clean the disks that the data is on to completely remove the personal data.

Installation overview

IBM Watson Machine Learning Server is installed using a virtual machine (VM) image that is shipped with prerequisites and a simple script to manage the Guest machine. All administration of guest machine can be managed only from Host machine.

Installing the IBM Watson Machine Learning Server follows this sequence:

1. Plan the installation, making sure you meet the minimum system requirements.
2. Run the installation script and configure the server.

Note: You must be connected to the internet to perform the installation. After the installation you can run Watson Machine Learning Server offline.

Planning the installation

Use this topic to plan for installing IBM Watson Machine Learning Server.

This topic covers:

- System requirements
- Security considerations

System requirements

This is the minimum configuration required for installing Watson Machine Learning Server.

Operating system

- Red Hat Enterprise Linux 7.6, 7.7, or 7.8

Hardware and software

The following resources are required for installing Watson Machine Learning Server 2.0.0-1:

- Memory: 64 GB
- CPU: 16 Core minimum
- Storage: 600 GB total disk space
    - 300 GB for Watson Machine Learning Server (Part of Root filesystem)
    - 300 GB for Watson Machine Learning data
- CPU virtualisation extension: AMD-V or Intel VT-x with nested virtualization enabled if installing on a virtual machine
- Hardware: x86 Intel-based processor

Notes:

- Internet access is required during setup.
- Installation must be performed from Root.
- All IPs in the DNS resolver file `resolv.conf` must be reachable

Security considerations

SSL certificate

Watson Machine Learning Server requires that you use a valid SSL certificate to set up the server. If you use the guided installation feature (available in Watson Machine Learning Server 2.0.0-1) you can request an SSL certificate and key as part of the installation. Use the `update certificate` installation option to update an expired certificate.

Tokens

Watson Machine Learning Server automatically generates a bearer token when a user signs in, and securely stores information in the user's home directory. This token expires based on the policies defined in Watson Machine Learning Server. When the user signs out, the stored bearer token is cleared.

Watson Machine Learning Server provides an encrypted bearer token in the model deployment details that an application developer can use for evaluating models online with REST APIs. The token never expires, and is limited to the model it is associated with.

Each project release uses one bearer token. Each web service deployment provides a deployment bearer token.

Authentication

By default, Watson Studio Local user records are stored in its internal repository database.

Authorization

Watson Machine Learning Server supports Admin and User roles for server access. For details on creating and managing user accounts, see Managing users.

Installing Watson Machine Learning Server

This topic describes installing IBM Watson Machine Learning Server.

Before you install

- Make sure your system meets the minimum requirements listed in Planning the installation.
- From root, create a folder where you will unzip the installation files and run the installation. **Note:** Do not unzip the package under root (/), as `libvirt` might fail and cause errors loading the virtual machine (VM). For this documentation we will use `/wmlserver` as the folder name.
- The installation package is located in Passport Advantage. Search for the product name to locate the download file. The following procedure provides steps for installing IBM Watson Machine Learning Server.

Installing the Watson Machine Learning Server

You have two options for installing Watson Machine Learning Server:

- Install using a guided script that prompts you for the information you need to install the server.
- Install using an advanced option where you provide the required details as part of the installation command

Before you install

1. Change to the folder you created as a prerequisite. For example, change to `/wmlserver`.
2. Download the Watson Machine Learning Server installation file to the installation folder on the designated Host machine.
3. Run `tar -xvzf <installer_file>` to unpack the installer file and extract the following files:
    - `wmlserver` script (used to manage Watson Machine Learning Server)
    - Vagrant Box file (contains VM image)
    - Vagrant file (contains the VM configuration)

Installing using the guided script

1. From the installation folder, for example, `/root/wmlserver`, run:

    ```
    ./wmlserver setup
    ```

When the script runs, it will check for minimum system requirements and indicate success if the requirements are met, or display an error if they are not.

1. When prompted, enter the following information:
    - Data directory path
    - Install Path
    - SSL certificate **Note** If you do not have an SSL certificate, you will have the option to create one as part of the installation process.
    - Private Key for the SSL certificate
    - Host name for the server
2. When the setup is complete, a confirmation message prompts you to start the server.

Installing using the advanced option

1. From the installation folder, for example, `/root/wmlserver`, run:

    ```
    ./wmlserver advinstall -d <data directory path> -i <installation path> -l <path for the SSL certificate> -k <private key for SSL certificate> -s <host name for the server>
    ```

    For example:

    ```
    ./wmlserver advinstall -d /root/WML/wmlsdata -i /root/WML/wmlsins -l /root/server.pem -k /root/privkey.pem -s example.com
    ```

2. When the setup is complete, a confirmation message prompts you to start the server.

When the server starts, it launches the Guest machine which contains all Watson Machine Learning services. The Guest machine is reachable on host port 31843 and the SPSS modeler on host port 31057.

Next steps

- After installation, you can start or manage the server using options for the `wmlserver` script. For a complete list of options for the `wmlserver` script, see Installation script options.
- Test your connection to Watson Machine Learning Server

Installation script options

This topic describes the options for the `wmlserver` installation script that you can use to install, remove, or manage IBM Watson Machine Learning Server 2.0.0-1.

View help

To view syntax and descriptions for all of the options for the installation script `wmlserver`, enter:

```
./wmlserver <options>
```

The list of options displays:

Option	Argument	Description
Advanced install	advinstall	WML server advanced installation
Backup	backup	Take backup of data directory specified by user
Capture diagnostic information	diagcollect	Capture diagnostic information of WML Server.
Get resource	get resource	Get CPU & memory allocation for the given resource.
Help	help	Get help about any command's usage.
Precheck	precheck	Checks for required system resources
Remove	remove	Uninstall WML Server.
Restore	restore	Restore the data directory specified by user
Setup	setup	WML guided installation
Start	start	Start WML Server after installation.
Status	status	Check status of WML Server.
Stop	stop	Stop WML Server.
Update certificate	update certificate	Update WML Server SSL certificate and private key.
Update resource	update resource	Update WML Server resources like CPU & memory.
Version	version	Version of WML Server.

Setup with guided install

Use this option to launch a guided installation script. The script checks that your system meets requirements, then prompts you for the following information:

- Data directory path
- Install Path
- SSL certificate **Note** If you do not have an SSL certificate, you will have the option to create one as part of the installation process.
- Private Key for the SSL certificate
- Host name for the server

When the setup is complete, a confirmation message prompts you to start the server.

Syntax

```
./wmlserver setup
```

Example

```
$ ./wmlserver setup
```

Returns a succcess message after installation, and prompts you to start the server using `./wmlserver start`.

Advanced installation

Use the advanced installation option if you already have an SSL and private key and you are comfortable providing all of the options on the command line. Otherwise, use the `setup` option for a guided installation.

Syntax

```
./wmlserver advinstall -d <data directory path> -i <installation path> -l <path for the SSL certificate> -k <private key for SSL certificate> -s <host name for the server>
```

Option	Description
-d	data directory path
-i	installation path
-l	path for the SSL certificate
-s	host name for the server

Example

```
   ./wmlserver advinstall -d /root/WML/wmlsdata -i /root/WML/wmlsins -l /root/server.pem -k /root/privkey.pem -s example.com
```

Check for system requirements

Run this check to see if your system can satisfy the requirements for installing Watson Machine Learning Server. System feedback either confirms that the requirements are met or notifies you about insufficient resources.

Syntax

```
./wmlserver precheck
```

Example

```
$ ./wmlserver precheck
```

Returns the following if successful:

```
To initiate the installation process, minimum resource configuration precheck is required
Minimum resources required: 16 CPU with 62 GB memory.

Resource precheck in progress...

  50% |████████████████████████                                  |

CPU resource requirement ✓ Passed
CPU resource on the host: 16 CPU

  62% |████████████████████████████                              |

Memory resource requirement ✓ Passed
Memory resource on the host: 62.76 GB

  81% |████████████████████████████████████                      |

Hardware virtualization requirement ✓ Passed

  93% |█████████████████████████████████████████                 |

resolv.conf validation ✓ Passed

 100% |████████████████████████████████████████████|

Success! Minimum resource precheck passed.
```

Start the Watson Machine Learning Server

After installation, start the server. The start-up sequence can take 40 - 50 minutes to complete.

Syntax

```
./wmlserver start
```

Example

```
$ ./wmlserver start
```

Returns the following if successful:

```
$ ./wmlserver start
WML Server starting...
WML Server will take 40-50 min to start.
There are 7 steps for the system to successfully start WML Server.

- Step 1 of 7
Enabling Modeler Server ports...
Checking for firewalld binaries... Success!
firewalld binaries are found on host machine
Enabling the Modeler Server ports in firewalld... Skipped
Firewalld is not enabled.

- Step 2 of 7
Editing VM configuration...
Switched the VM configuration 'overcommit_memory' to '2'
Updated the VM configuration 'overcommit_ratio' to '90'
Freed up the buff/cache
Restarting libvirt... Completed

- Step 3 of 7
Vagrant plugin 'vagrant-libvirt' has been installed
Starting the WML Server...
KVM VM has been brough up
Freed up the buff/cache

- Step 4 of 7
Running configured provisioner with start... Completed
WML Server startup has been initiated

- Step 5 of 7
Updating SSL certificate skipped

- Step 6 of 7
Updating Modeler Port...
Executing the vagrant provision with name 'update-modeler-port'... Completed
Modeler Port is updated

- Step 7 of 7
Enabling the RPC ports in firewalld... Completed

Reloading firewall... Completed
```

Stop the Watson Machine Learning Server

Stop the server to perform configuration or maintenance tasks. The process take 5 to 10 minutes to complete.

Syntax

```
./wmlserver stop
```

Example

```
$ ./wmlserver stop
```

Returns the following if successful:

```
$ ./wmlserver stop
Do you want to stop WML Server now?
This may take 5-10 min to complete

Select your option with arrow keys and press Enter
✓ Yes, stop WML Server now

Stopping the WML Server...
WML Server will take 5-10 min to stop.
There are 2 steps for the system to successfully stop WML Server.

- Step 1 of 2
Executing the vagrant provision with name 'stop'... Completed
Openshift cluster has been brought down

- Step 2 of 2
Vagrant plugin 'vagrant-libvirt' has been installed
Executing the command: 'vagrant halt'... Completed

Success! WML Server is successfully stopped
```

Check the status of Watson Machine Learning Server

Check the status of Watson Machine Learning Server. There are 4 possible states:

- **Not found** if the start command has not yet been successfully executed.
- **Starting** when the `start` command is executing and the Watson Machine Learning Server services are loading.
- **Ready** when all of the services are loaded and available for use.
- **Stopped** indicates all the Watson Machine Learning Server services are stopped.

Syntax

```
./wmlserver status
```

Example

```
$ ./wmlserver status
```

Returns the following when the server is ready:

```
$ ./wmlserver status
WML Server Version : 2.0
WML Server Status  : READY
CNAME              : baremetal01.wml-ys1.cloud
Port number        : 31843

To Login to WML Server VM run:
export VAGRANT_HOME=/root/anand/wmls_install/.vagrant.d
vagrant ssh
```

Update the SSL certificate

Update the certificate and private key of a Watson Machine Learning Server installation. This command can only be run after a successful installation and start operation. To upgrade the certificate, you must provide:

- SSL Certificate file
- SSL Private Key file
- Hostname

Note: Make sure that the status of the WML Server installation is `READY` before issuing this command. You can check the status by running the `status` command.

Syntax

```
./wmlserver update certificate [options]
```

Flag	Option	Description
-l	–cert	SSL Server Certificate file
-k	–key	SSL Private Key file
-s	–hostname	Public hostname of WML Server

Example

```
$ ./wmlserver update certificate -l /root/certs/fullchain.pem -k
/root/certs/privkey.pem -s baremetal.ml.test.cloud.example.com
```

Returns the following after a successful update:

```
Validating SSL Certificate and private key...
 100% |████████████████████████████████████████████████████████|

Success! SSL certificate and private key validation completed.

WML Server update certificate has been initiated
Updating SSL certificate...
Executing the vagrant provision with name 'update-cert'... Completed
Removing the SSL certificate file, private key file and hostname file... Completed
SSL Certificate is updated

Success! WML Server is successfully updated with given SSL certificate
```

Get information about system resources

Get the CPU and memory resource allocation for a given resource in a IBM Watson Machine Learning Server installation. This command can only be run after a successful installation and start operation. Currently, only the modeler resource is supported for getting the resource allocation.

You can then update the resource allocation using the `update resource` command.

Syntax

```
./wmlserver get resources [options]
```

Flag	Option	Description
-n	–name	Name of the resource (Supported resource name: ‘modeler’)

Example

```
$ ./wmlserver get resource -n modeler
```

Returns the following:

```
$ ./wmlserver get resource -n modeler
+----------------+-------+-------------+
|    RESOURCE    |  CPU  | MEMORY (GB) |
+----------------+-------+-------------+
| Total          |    48 |     45.7877 |
| Allocated      | 8.215 |      22.418 |
| Modeler server |     2 |           3 |
+----------------+-------+-------------+
```

Update system resources

Update the CPU and memory resource allocation for a given resource in a IBM Watson Machine Learning Server installation. This command can only be run after a successful installation and start operation. Currently, only the modeler resource is supported for updating the resource allocation.

As part of the `update resources` command, you must provide a value for at least one of the CPU or memory resources.

Syntax

```
./wmlserver update resources [options]
```

Flag	Option	Description
-n	–name	Name of the resource (Supported resource name: 'modeler')
-m	–memory	Update memory. Specify memory in GigaBytes(GB) (Optional Parameter)
-c	–cpus	Update the number of CPU cores (Optional Parameter)

Example

```
$ ./wmlserver update resource -n modeler
```

Returns the following:

```
$ ./wmlserver update resource -n modeler -c 3 -m 4
Installing the Vagrant plugin 'vagrant-libvirt'... Completed
Vagrant plugin 'vagrant-libvirt' has been installed
Get the resource allocation for modeler... Completed

Updating Modeler Server resources...
Executing the vagrant provision with name 'update-resource'... Completed

Success! Modeler resource is updated
```

Check the Watson Machine Learning Server version

Check the version of your server.

Syntax

```
./wmlserver version
```

Example

```
$ ./wmlserver version
```

Returns something similar to this:

```
$ ./wmlserver version
WML Server Version : 2.0
-----------------------
wmlserver build info:
Number    : 54
Timestamp : Sun Jun 28 01:05:00 PDT 2020
```

Remove the Watson Machine Learning Server

Uninstall and remove Watson Machine Learning Server.

Syntax

```
./wmlserver remove
```

Example

```
$ ./wmlserver remove
```

You will be prompted to confirm the removal and notified when the process is complete.

Collect diagnostic information about the Watson Machine Learning Server

You can view diagnostic information to help you manage the Watson Machine Learning Server. The following configurations and files are captured as part of this process:

- Output of `lscpu`
- Output of `lsmod`
- All the log files in the `log` directory
- `Vagrantfile`
- The last KVM log file located in `/var/log/libvirt/qemu/`
- Output of vagrant provision `--provision-with diag-collect`

All of this is archived into a tarball file (`.tar.gz`) with a filename with the timestamp in the format: `wmlserver_diag_<yyyy>-<mm>-<dd>_<hh>-<mm>-<ss>.tar.gz` in the same directory in which the Watson Machine Learning Server executable is located.

Syntax

```
./wmlserver diagcollect
```

Example

```
$ ./wmlserver diagcollect
```

Back up the Watson Machine Learning Server

Back up the contents of Watson Machine Learning Server data directory to a folder you specify. You can use the `restore` option to restore the content to the same server or to a different server.

Syntax

```
./wmlserver backup [options]
```

Flag	Option	Description
-d	–host-directory	Host Directory to store WML Server data
-b	–backup-directory	Backup directory path

Example

```
$  ./wmlserver backup -d /root/company/wmls_data -b /root/company/wmls_backup
```

Returns a success notification when the backup is complete. The backup is written to a time-stamped file in the specified backup directory.

Restore the Watson Machine Learning Server

After creating a backup of the contents of Watson Machine Learning Server data directory to a folder you specify, use the `restore` option to restore the contents to the same server or to a different server.

Note: Before you restore, you must stop the server. Follow the steps for Stop the Watson Machine Learning Server.

Syntax

```
./wmlserver restore [options]
```

Flag	Option	Description
-d	–host-directory	Host Directory to store WML Server data
-f	–backup-file	Backup file with absolute path

Example

```
$ ./wmlserver restore -d /root/company/wmls_data -f
/root/company/wmls_backup/backup-wmlserver.example.com-2020-07-05_02-42-01
```

Returns something similar to this:

```
Checking WML Server status is 'STOPPED'...
WML Server is stopped. Proceeding with restore
Extracting backup from tar file... Completed
Restarting the NFS service... Completed

Verifying the NFS service on host machine...
Checking status of NFS processes on this host machine...
NFS processes are running on this host machine
Restarting the NFS service... Completed

Success! Restore finished, start the WML server

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

To start WML Server run: ./wmlserver start
```

Notes

- Use `./wmlserver start` to start the server after the restore.
- If you restore to a different server from where the backup was created, use `./wmlserver update certificate` to apply your SSL certificate and private key.
- If you restore to a different server from where the backup was created, use the `rsync` command to transfer the backup folder from the remote host to the current host machine. If you do not sync the content, you might get errors when trying to create deployments from models you transferred.

  Syntax:

  ```
  rsync -av -e ssh --progress root@<<Remote host
  IP>>:/path/to/remote/backup/folder /path/to/transfer/folder
  ```

  Example:

  ```
  rsync -av -e ssh --progress root@1.23.45.67:/root/company/wmls_backup/backup-
  wmlserver.example.com-2020-07-05_02-42-01 .
  ```

Test the server connection

You can test the server connection using the Python client to connect.

Watson Machine Learning Server requires a bearer token to authenticate the user calling the REST APIs. The token lasts for 13 hours. The following example shows how to retrieve a bearer token from the user management service:

```
curl -k -X GET
<WMLS_URL>/v1/preauth/validateAuth -u joe:joespasswd
```

where 'WMLS_URL' represents the Watson Machine Learning Server URL, joe represents a user with admin access to the server, and joespasswd represents the password.

The call returns a JSON snippet from which the bearer token can be extracted from the accessToken field:

{"username":"joe", "role":"User", "uid":"1003", "accessToken":"eyJhbGc...1AjT_w", "*messageCode*":"success", "message":"success" }

After you retrieve your authentication token, you can connect to the server.

Use the token to authenticate with the server

Use your user access token to connect to the server. This will mask your user credentials.

```
import sys,os,os.path
token = os.environ['USER_ACCESS_TOKEN']

wml_credentials = {
"token": token,
"instance_id" : "wml_local",
"url": "https://111.22.333.444:31843",
"version": "2.0"
}
```

Where:

- Instance id is always "wml_local"
- 'url' represents the Watson Machine Learning Server URL and port number
- version number is the version of Watson Machine Learning Server

Once you verify that you can successfully connect to the server, users with access to the server can connect from IBM Watson Studio Desktop or via the Python client. For details see Connecting to the Watson Machine Learning Server.

Administering the server

Administration of IBM Watson Machine Learning Server involves these tasks:

- Managing server access
- Troubleshooting

Managing users

User IDs for accessing the IBM Watson Machine Learning Server must be created with APIs.

Note: To perform administrative activities on Watson Machine Learning Server such as adding and removing users, the server administrator must use the Watson Machine Learning Server management script `wml.sh` and must have root access to the machine on which the server is installed.

See this sample notebook for how to modify, add or remove users.

This topic describes how to:

- List users
- Add a user
- Remove a user
- View access information about a user
- Change a user's role

Prerequisites

- You must have admin access to IBM Watson Machine Learning Server to perform these tasks.
- Use a bearer token for authentication. For details on getting a token, see Testing the connection.

List users on the server

Use `GET /v1/usermgmt/users` to to list all users with access to Watson Machine Learning.

GET /v1/usermgmt/users

Lists all users for a given server

Notes

Authentication is required. The HTTP request is expected to contain an Authorization header with the format Bearer or a Cookie header.

Sample code

```
curl -k -X GET  https://<WML Server Hostname>:31843/v1/usermgmt/users -H
"Authorization: Bearer <TOKEN>" -H "Content-Type:application/json"
```

Adding a user

Use `POST /v1/USER` to create a new user for IBM Watson Machine Learning Server.

Provide the user's details and if LDAP is setup, also select if the user is to be authenticated by LDAP. If External LDAP is picked as an authenticator for the user, there is no password promt. If not, the user password is set to an initial default which the user can change after they login. URL

POST /v1/user

Parameter	Description
username (required)	User's full name
displayName (required)	Users email
role (required)	User role, can be 'Admin' or 'User'
message (optional)	Message to send to use on create
email (optional)	Users email
password (optional)	Users password

Notes

Authentication is required. The HTTP request is expected to contain an Authorization header with the format Bearer or a Cookie header.

Sample code

```
curl -X POST  https://<WML Server Hostname>:31843/v1/user  -d
'{"username":"wmluser1","displayName":"wml
user1","role":"Admin","password":"password"}' -H "Authorization: Bearer <TOKEN>" -H
"Content-Type:application/json"
```

Removing a user

Use `DELETE /v1/user/:username` to delete a user from access rights to Watson Machine Learning. Note that you cannot delete the user information for the server administrator.

DELETE /v1/user/:username

The only parameter this command accepts is the full name of the user being deleted.

Parameter	Description
username (required)	User's full name

Notes

Authentication is required. The HTTP request is expected to contain an Authorization header with the format Bearer or a Cookie header.

Sample code

```
curl -X DELETE https://<WML Server Hostname>:31843/v1/user/wmluser1   -H
"Authorization: Bearer <TOKEN>" -H "Content-Type:application/json"
```

Getting information about a user

Use `GET /v1/user/:username` to get information about users with access to Watson Machine Learning.

GET /v1/user/:username

Accepts an optional parameter to show users in all states.

Parameter	Description
includeAll(optional)	Determines whether to include user entries that are in pending or denied state

Notes

Authentication is required. The HTTP request is expected to contain an Authorization header with the format Bearer or a Cookie header.

Sample code

```
curl -X GET https://<WML Server Hostname>:31843/v1/user/wmluser1  -H "Authorization:
Bearer <TOKEN>" -H "Content-Type:application/json"
```

Changing a user's details

PUT /v1/user/(username) Changing a user's details

You can change a user's role and other details. Changing a user's role expands or limits access to Watson Machine Learning. The available roles are admin or user. An admin can manage the server and add or remove users. Users can only post to the server. Other details such as `displayName` can also be changed with the `PUT` command.

PUT /v1/user/

Current user:

Parameter	Description
username (required)	The name that needs to be updated
displayName (optional)	Users email
role (optional)	User role, can be 'Admin' or 'User'
message (optional)	Message to send to use on create
email (optional)	Users email
password (optional)	Users password

New User:

Parameter	Description
username (optional)	Users full name
displayName (optional)	Users email
role (optional)	User role, can be 'Admin' or 'User'
message (optional)	Message to send to use on create
email (optional)	Users email
password (optional)	Users password

Notes

Authentication is required. The HTTP request is expected to contain an Authorization header with the format Bearer or a Cookie header.

Sample code

```
curl -X PUT  https://<WML Server Hostname>:31843/v1/user  -d
'{"username":"wmluser1","displayName":"wml
user1_UPDATED","role":"User","message":"Welcome to WMLS
UPDATED","email":"wmluser1_UPDATED@ibm.com","password":"password_UPDATED"}' -H
"Authorization: Bearer <TOKEN>" -H "Content-Type:application/json"
```

Troubleshooting

In the event of an installation failure, or if issues arrive post-installation with the Watson Machine Learning services, you can collect diagnostic information to share with IBM support.

To do so, run the installation script with the (diagcollect> parameter. This command will produce a file with the format `wmlserver_diag_<timestamp>.tar.gz` in the current directory. You can share this file with IBM Support to help troubleshoot issues.

Example:

```
sh wml.sh diagcollect
```

Sample output: wmlserver_diag_2019-09-17-08.54.55.tar.gz

Deploying assets (Version 2.0.0-1)

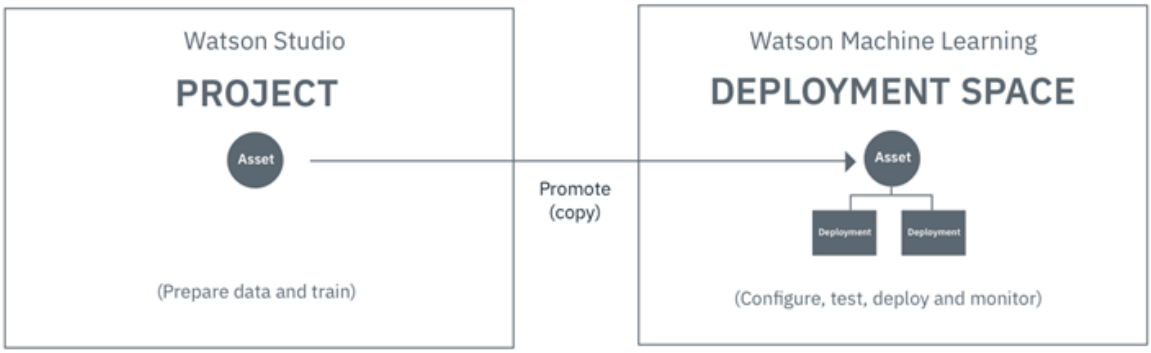
Deployment is the final stage of the lifecycle of a model or script, where you run your models and code. Watson Machine Learning provides the tools you need to deploy an asset, such as an SPSS modeler flow, a machine learning model, or a function, depending on what tools are configured for your system.

Following deployment, you can use model management tools to evaluate your models. IBM Watson OpenScale tracks and measures outcomes from your AI models, and helps ensure they remain fair, explainable, and compliant. Watson OpenScale also detects and helps correct the drift in accuracy when an AI model is in production

Service The Watson Studio, Watson Machine Learning, Watson OpenScale, and other supplemental services are not available by default. An administrator must install these services on the IBM Cloud Pak for Data platform. To determine whether a service is installed, open the Services catalog and check whether the service is enabled

Deployment overview

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 an asset to a space, components required for a successful deployment, such as a training library, model definition, other dependent assets, 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.

For models and function code, you can also create a deployment programmatically 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.

Connecting to the Watson Machine Learning Server (2.0.0-1)

You can connect to Watson Machine Learning Server from the Watson Studio Desktop interface, or from a notebook, using the Python client.

Connecting from Watson Studio Desktop

After you install IBM Watson Machine Learning Server, IBM Watson Studio Desktop users 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.

Provide the information required to connect to the server to your Watson Studio Desktop users. When Watson Studio Desktop users choose to add IBM Watson Machine Learning Server for a runtime for their SPSS model flow or as a deployment space, they will need to provide the following to connect to the server:

- Server hostname
- Port number
- Username and password
- (Optional) A self-signed certificate to support an SSL connection with Watson Machine Learning Server. To add a user, see Managing users.

Connecting to the Watson Machine Learning Server

Watson Studio Desktop users can then connect to Watson Machine Learning Server as follows:

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**.

For details on deploying assets to IBM Watson Machine Learning Server, see the documentation for IBM Watson Studio Desktop.

Connecting using the Python client

You can deploy a model from a notebook using the Python client.

If you create a notebook in Watson Studio Desktop, the Python client is installed automatically. To manually install the Python library:

```
!pip install --upgrade watson-machine-learning-client-V4
```

Supply the server location and the user credentials to connect to the server and authenticate, as follows:

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient
client = WatsonMachineLearningAPIClient( wml_credentials )

wml_credentials = {
   "instance_id": "wml_local",
   "url" : "https://<WML Server Hostname>:31843",
   "username":"<USER NAME>",
   "password": "<PASSWORD>",
```

```
    "version": "2.0"
}
```

Where:

- Instance id is always “wml_local”
- ‘url’ represents the Watson Machine Learning Server URL and port number
- username and password are the credentials for a user with admin access to the server
- version number is the version of Watson Machine Learning Server

Specifying a self signed certificate

If Watson Machine Learning Server is using self-signed certificates to support SSL connections, you can perform Python requests API call in one of the following ways:

1. You can specify the local certificate file as a path using the `verify` parameter of the `request` function as shown in this example:

```
import requests
resp = requests.post(WML_SERVICE_URL + "/v4/deployments/heartbeat",
                     headers = { 'Content-Type': 'application/json', 'Authorization': token},
                     verify='/Jupyter_Notebooks/cacert.pem')
print(resp.content)
```

```
b'{"version":"0.1.781-201912182257","build":"2019-12-19T09:09:02Z","hostname":"wml-os-manager-0"}'
```

2. If you want to disable SSL certificate verification, pass “False” to the `verify` parameter of the request function. This is not recommended as it will create an insecure connection.

```
import requests
resp = requests.post(WML_SERVICE_URL + "/v4/deployments/heartbeat",
                     headers = { 'Content-Type': 'application/json', 'Authorization': token},
                     verify=False)
print(resp.content)
```

```
b'{"version":"0.1.781-201912182257","build":"2019-12-19T09:09:02Z","hostname":"wml-os-manager-0"}'
```

Supported frameworks

You can use popular tools, libraries, and frameworks to train and deploy machine learning models and functions using IBM Watson Machine Learning. This topic lists supported versions and features.

Supported machine learning frameworks

The following table lists the machine learning frameworks that are certified for use on the Watson Machine Learning Server. Note the following:

- For security reasons, Python 3.6 is deprecated in favor of Python 3.7.
- If a framework is marked as deprecated, then support for the framework will be removed in a future release. For details on migrating a model or function, see Upgrading from a deprecated framework.
- Models built using an unsupported framework can be saved to the Watson Machine Learning repository, but they cannot be deployed. To successfully save and deploy a model, use a framework from this table of supported frameworks.

Supported frameworks for Watson Machine Learning Server

Framework	Versions	Online	Batch	CoreML

Framework	Versions	Online	Batch	CoreML
Spark	2.3 2.4	Yes	Yes Inline payload only	No
PMML	3.0 to 4.3	Yes	Programmatic only Inline payload only	No
Hybrid/AutoML	0.1	Yes	Yes	No
SPSS	 18.1 18.2	Yes	Yes	No
Scikit-learn	0.20 (deprecated) 0.22 (deprecated) 0.23	Yes	Yes	Yes
XGBoost	0.80 (deprecated) 0.90	Yes	Yes	Yes
TensorFlow	1.15 (deprecated) 2.1 (deprecated)	Yes	Yes	No
Keras	2.2.5 (deprecated)	Yes	Yes	No
Caffe	1.0 (deprecated)	Yes	Yes	No
PyTorch	1.1 (deprecated) 1.2 (deprecated) 1.3	Yes	Yes	No
Decision Optimization	12.9 12.10	No	Yes	No
Python function	0.1	yes	Programmatic only Inline payload only	no
Python scripts	1.0	No	Yes	No

Specifying a model type and configuration

The environment for a machine learning model or Python function is made up of the hardware and software specifications.

Software specifications and runtimes

Software specifications have replaced runtimes as the way to define the language and version you use for a model or function. They enable you to better configure the software used for running your models and functions. Previously, you could only choose among predefined and non-configurable runtimes. Now software specifications allow you to precisely define not only the software version to be used, but also include additional extensions (such as using conda .yml files or custom libraries).

Defining a software specification

Use this format for specifying the language and version used for a model or function:

```
meta_props={
     client.repository.ModelMetaNames.NAME: "skl_pipeline_heart_problem_prediction",
     client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid,
     client.repository.ModelMetaNames.TYPE: "scikit-learn_0.23"
}
```

To get the list of predefined software specifications, use:

```
client.software_specifications.list()
```

which returns a list of software specifications.

Save a specification to a variable:

```
software_spec_uid = client.software_specifications.get_uid_by_name("scikit-
learn_0.23-py3.7")
software_spec_uid
```

Choosing a predefined software specification

This table lists the model types and software specifications that map to the supported frameworks for deployments.

This table lists the predefined, or base, model types and software specifications.

Python deprecation notice

Due to a security vulnerability, Python 3.6 is deprecated in favor of Python 3.7. If a framework is marked as deprecated, then support for the framework will be removed in a future release.

Framework	Versions	Model Type	Default software_specification
Spark	2.4	mllib_2.4	spark-mllib_2.4
Spark	2.3	mllib_2.3	spark-mllib_2.3
PMML	3.0 to 4.3	pmml_. (or) pmml_..*3.0 - 4.3	spark-mllib_2.4
Hybrid/AutoML	0.1	wml-hybrid_0.1	hybrid_0.1
SPSS	17.1	spss-modeler_17.1	spss-modeler_17.1
SPSS	18.1	spss-modeler_18.1	spss-modeler_18.1
SPSS	18.2	spss-modeler_18.2	spss-modeler_18.2
Scikit-learn	0.23	scikit-learn_0.23	default_py3.7
Scikit-learn	0.22	scikit-learn_0.22	scikit-learn_0.22-py3.6 (deprecated) default_py3.6 (deprecated)
Scikit-learn	0.20	scikit-learn_0.20	scikit-learn_0.20-py3.6 (deprecated)
XGBoost 0.90	xgboost_0.90 with Python 3.7	If model is trained with sklearn wrapper (XGBClassifier or XGBRegressor) use scikit-learn_0.23	default_py3.7
XGBoost 0.90	xgboost_0.90 with Python 3.6	If model is trained with sklearn wrapper (XGBClassifier or XGBRegressor) use scikit-learn_0.22	scikit-learn_0.22-py3.6(deprecated) default_py3.6(deprecated) xgboost_0.90-py3.6 (deprecated)
XGBoost 0.82	xgboost_0.82	If model is trained with sklearn wrapper (XGBClassifier or XGBRegressor) use scikit-learn_0.20	scikit-learn_0.20-py3.6 (deprecated)
Tensorflow	1.15	tensorflow_1.15	tensorflow_1.15-py3.6 (deprecated) default_py3.6 (deprecated)

Framework	Versions	Model Type	Default software_specification
Tensorflow	2.1	tensorflow_2.1	default_py3.7 tensorflow_2.1-py3.7
Keras	2.2.5	keras_2.2.5	tensorflow_1.15-py3.6 (deprecated) default_py3.6 (deprecated)
PyTorch	1.1	pytorch-onnx_1.1	pytorch-onnx_1.1-py3.6 (deprecated)
PyTorch	1.2	pytorch-onnx_1.2	pytorch-onnx_1.2-py3.6 (deprecated) pytorch-onnx_1.2-py3.6-edt (deprecated) default_py3.6 (deprecated)
PyTorch	1.3.1	pytorch-onnx_1.3.1	default_py3.7
Python Functions	0.1	NA	default_py3.7 ai-function_0.2-py3.6 (deprecated) default_py3.6 (deprecated)
Python Scripts	1.0	NA	default_py3.7 default_py3.6 (deprecated)

Managing assets that refer to discontinued software or frameworks

Note these guidelines for updating assets that rely on discontinued software specifications or frameworks. In some cases, the update is seamless. In other cases, your action is required to retrain or redeploy assets.

Managing assets that refer to discontinued software specifications

- During migration, assets that refer to the discontinued software specification will be mapped to a comparable supported default software specification. This will be done in cases where the model type is not discontinued.
- When you create new deployments of the migrated assets, the updated software specification in the asset's metadata will be used.
- Existing deployments of the migrated assets will be updated to use the new software specification. If there is a failure during deployment or scoring due to framework or library version incompatibilities, follow the manual steps for recreating the asset with the supported framework versions.

Migrating assets that refer to discontinued framework versions

- During migration, the model type will not be updated.
- The existing deployments post migration will be removed and new deployments for this framework will not be allowed.

Notes about upgrading from a deprecated framework

Update models or functions built with deprecated frameworks.

Upgrading a machine learning model

Follow these steps to update a model built with a deprecated framework.

Option 1: Save the model with a compatible framework

1. Download the model from the Watson Machine Learning repository.
2. Save the model back to the Watson Machine Learning repository with a model type and version supported in the current release.
3. Deploy the model.
4. Score the model to generate predictions.

If there is a failure when you deploy or score the model, it means that the model is not compatible with the new version used for saving the model. In this case, use Option 2.

Option 2: Retrain the model with a compatible framework

1. Retrain the model with a model type and version supported in the current version.
2. Save the model to the Watson Machine Learning repository with the supported model type and version.
3. Deploy and score the model.

Upgrading a Python function

Follow these steps to update a function built with a deprecated framework.

Option 1: Save the python function with a compatible runtime or software specification:

1. Download the Python function from the Watson Machine Learning repository.
2. Save the Python function back to the Watson Machine Learning repository with a supported runtime or software specification version.
3. Deploy the Python function.
4. Score the Python function to generate predictions.

If there is a failure when you score the Python function, it means that the function is not compatible with the new runtime or software specification version used for saving the Python function. In this case, use Option 2.

Option 2: Modify the function code and save it with a compatible runtime or software specification

- Modify the Python function code to make it compatible with the new runtime or software specification version. This could involve updating the dependent libraries installed within the python function code.
- Save the Python function to the Watson Machine Learning repository with the new runtime or software specification version.
- Deploy and score the Python function.

Creating a custom software specification in a project

If your Scikit-learn, XGBoost and Tensorflow model requires custom components such as user-defined transformers, estimators, or user-defined tensors, you can create a custom software specification derived from a base, or predefined specification.Python functions and Python scripts also support custom software specifications.

You can use custom software specification to reference any third-party libraries, user-created python packages, or both. Third-party libraries or user-created python packages must be specified as package extensions which can then be referenced in a custom software specification.

Custom software specifications are promoted with the assets that use them. Python functions and scripts, and models created with the Scikit-learn, XGBoost and Tensorflow frameworks can use the software specifications for deployments.

Note that software specifications are also included when you import a project or space that includes one.

Overview of creating a custom software specification

The high-level steps to create a custom software specification that uses third-party libraries and user-created python packages.

1. Create a custom software specification
2. Create a package extension to save a conda YAML file that contains a list of 3rd party libraries. **Note:** This step is not required if the model does not have any dependency on a third-party library.
3. Create a package extension to save a user-created Python package. **Note:** This step is not required if the model does not require any user-defined transformers, estimators, or user-defined tensors.
4. Add a reference of the package extensions to the custom software specification you created.

Creating a custom software specification in a notebook

You can use the Watson Machine Learning APIs or Python client to define a custom software specification that is derived from a base specification.

These notes are applicable when specifying software specification for Scikit-Learn, XGBoost, Tensorflow, Keras, PyTorch, or Caffe models trained in Watson Studio notebooks, or Python functions developed in a notebook.

- If you have created the models or Python functions in a Watson Studio notebook with “Default Python 3.7” environment, then save the model or function with a reference to the software specification using the name “default_py3.7”. To get the corresponding software specification id, use:

```
 softwareSpecId =
wml_client.software_specifications.get_uid_by_name('default_py3.7')
```

- You can create custom environments in Watson Studio to install 3rd party libraries required to execute your Python scripts. If you have trained a model or developed a Python function in a Watson Studio notebook with acustomized environment, specify the reference to the custom software specification when you save the model or function. The name of the custom software specification will be the same as the name of the custom environment. For example, if the custom environment is called “my_cust_model1_env”:

```
  softwareSpecId =
wml_client.software_specifications.get_uid_by_name('my_cust_model1_env')
```

To create a custom software specification

This code template illustrates how to create a custom software specification using the Python client.

1. Authenticate and create the client.

2. Create and set the default deployment space, then list available software specifications.

```
metadata = {
            wml_client.spaces.ConfigurationMetaNames.NAME: 'examples-create-
software-spec',
            wml_client.spaces.ConfigurationMetaNames.DESCRIPTION: 'For my
models'
            }
space_details = wml_client.spaces.store(meta_props=metadata)
```

```
space_uid = wml_client.spaces.get_uid(space_details)

# set the default space
wml_client.set.default_space(space_uid)

# see available meta names for software specs
print('Available software specs configuration:',
wml_client.software_specifications.ConfigurationMetaNames.get())
wml_client.software_specifications.list()

asset_id = 'undefined'
pe_asset_id = 'undefined'
```

3. Create the metadata for package extensions to add to the base specification.

```
 # create the metadata for package extensions
 pe_metadata = {
     wml_client.package_extensions.ConfigurationMetaNames.NAME: 'My custom
library',
     # wml_client.software_specifications.ConfigurationMetaNames.DESCRIPTION:
'...', # optional
     wml_client.package_extensions.ConfigurationMetaNames.TYPE: 'conda_yml'
 }
```

4. Create the package extension to save to a conda yaml file.

```
    # name: testing1
    # dependencies:
    #   - regex
 pe_asset_details = wml_client.package_extensions.store(meta_props=pe_metadata,
file_path='resources/customlibrary.yaml')
 pe_asset_id = wml_client.package_extensions.get_uid(pe_asset_details)

 base_id = wml_client.software_specifications.get_uid_by_name('default_py3.7')
```

5. Create the metadata for the software specification and store the software specification.

```
 # create the metadata for software specs
 ss_metadata = {
     wml_client.software_specifications.ConfigurationMetaNames.NAME: 'Python
3.7 with pre-installed ML package',
     wml_client.software_specifications.ConfigurationMetaNames.DESCRIPTION:
'Adding some custom libraries like regex', # optional

wml_client.software_specifications.ConfigurationMetaNames.BASE_SOFTWARE_SPECIFI
CATION: {'guid': base_id}
 }
 # store the software spec
 ss_asset_details =
wml_client.software_specifications.store(meta_props=ss_metadata)
 # get the id of the new asset
 asset_id = wml_client.software_specifications.get_uid(ss_asset_details)
```

6. Associate the package extension with the software specification.

```
 wml_client.software_specifications.add_package_extension(asset_id,
pe_asset_id)

 ss_asset_details = wml_client.software_specifications.get_details(asset_id)
 print('Package extensions', pp.pformat(ss_asset_details['entity']
['software_specification']['package_extensions']))
```

Propagating software specification and package extension from projects to deployment spaces

Custom software specifications and package extensions created in a Watson Studio project can be exported to a deployment space using the “Promote” option in the Watson Studio interface. Further, after you promote a custom software specification and package extension from a project to a space, follow these steps after making any changes to the software specification and package extension in the project, to trabsfer the changes to the deployment space.

1. Delete the software spec, package extensions, associated models(optional) in the space using the Watson Machine Learning Python Client.
2. In a Watson Studio project, promote the model, function, or script that is associated with the changed custom software specification and package extension to the space.

Using a hardware specification

You can include an existing hardware specification using these steps.

1. To get details of the available hardware specifications, use GET v2/hardware_specifications. For example:

```
curl -s -k -X GET https://example.com/v2/hardware_specifications?name=S -H
"content-type: application/json" -H "Authorization: Bearer $token"

{"total_results":1,"resources":[{"metadata":{"created_at":"2020-02-
25T09:41:57.779Z","updated_at":"2020-02-
25T09:41:57.779Z","name":"S","description":"A hardware specification providing
2 CPU cores and 8 GiB of
memory.","asset_type":"hardware_specification","asset_id":"bb69c3be-e441-4b59-
a193-846491de7b9a","href":"/v2/hardware_specifications/bb69c3be-e441-4b59-a193-
846491de7b9a"},"entity":{"hardware_specification":{"nodes":{"cpu":
{"units":"2","model":""},"mem":{"size":"8Gi"},"num_nodes":1}}}}]}
```

2. Deploy the batch job with the hardware specification in the payload. For example, to specify hardware by name:

```
    {
         "asset”: {
            "href":"/v4/models/2d0c85e9-a593-46aa-89a0-241bcdbe45ba?
space_id=d21487e9-9f6f-4be7-9017-73e6e5eb2011"
         },
         "space": {
            "href":"/v4/spaces/$spaceId"
         },
         "batch": {},
         "hardware_spec": {
             "name": "S",
             "num_nodes" : 1
         }
    }
```

   To specify hardware by ID:

```
  {
      "asset": {
             "href":"/v4/models/2d0c85e9-a593-46aa-89a0-241bcdbe45ba?
space_id=d21487e9-9f6f-4be7-9017-73e6e5eb2011"
        },
      "space": {
            "href": "/v4/spaces/d21487e9-9f6f-4be7-9017-73e6e5eb2011"
       },
       "batch":{},
       "hardware_spec": {
              "id": "bb69c3be-e441-4b59-a193-846491de7b9a",
```

```
                "num_nodes":1
            }
    }
```

3. Create the payload for the job. If you specify a hardware configuration, it overrides and replaces the hardware_spec in the jobs payload and the job will run with the hardware_spec mentioned in the jobs payload.

   For example, specifying a payload with a hardware name:

```
 {
    "deployment": {
         "href":"/v4/deployments/c6ca18af-1d6a-4c6a-ad78-93606144ec00"
      },
    "scoring" : {
                    "input_data":[{
                            "id": "data1",
                            "fields" :
["Age","Sex","BP","Cholesterol","Na","K"],
                             "values" :
[[23,"F","HIGH","HIGH",0.792535,0.031258]]
                           }]
     },
      "hardware_spec": {
                "name": "S",
                "num_nodes":1
      }
   }
```

   Specifying hardware by ID:

```
 {
    "deployment": {
         "href":"/v4/deployments/c6ca18af-1d6a-4c6a-ad78-93606144ec00"
      },
    "scoring" : {
                    "input_data":[{
                            "id": "data1",
                            "fields" :
["Age","Sex","BP","Cholesterol","Na","K"],
                             "values" :
[[23,"F","HIGH","HIGH",0.792535,0.031258]]
                           }]
     },
      "hardware_spec": {
                "id": "bb69c3be-e441-4b59-a193-846491de7b9a",
                "num_nodes":1
      }
   }
```

Persisting a Tensorflow model with custom layers

If you build a custom layer model with Tensorflow 2.1 and the `tf.keras` library, follow these steps to persist the model to the Watson Machine Learning repository so you can successfully deploy the model.

Requirements for V4 GA endpoints

Note these requirements if you created a model that uses a V4 GA endpoint, on Cloud Pak for Data as a Service, or Cloud Pak for Data 3.5.

- Save the source code containing the custom layer or other custom object definitions in `/ml/v4/model_definitions`.
- The model metadata should be referenced in the `model_definition.id` for the `model_definition`.
- The model metadata should have metadata `user_defined_objects` populated with key-value pairs as its value. The key references the name of custom layer object and the corresponding value points to its class.

Example

For a model which uses `CustomLayer tf.keras Layer` class in the file `custom_layers.py` in source code, the `user_defined_objects` metadata will look as follows:

```
{
        "name": "mytfmodel",
        "type": "tensorflow_2.1",
        "software_spec": default_py3.7,
        "user_defined_objects": {
                "CustomLayer": "custom_layers.CustomLayer",
        }
}
```

Requirements for V4-beta API endpoints

Note these requirements if you created a model that uses a V4-beta endpoint, such as on Watson Machine Learning Server 2.0.0.1, or Cloud Pak for Data 3.0.1 with Patch 5 applied.

- Save the source code containing the custom layer or other custom object definitions in `wml_model_definition` asset (present as `model_definitions` in Python client).
- The model metadata should be referenced in the `model_definition.id` for the `model_definition`.
- The archive file in which the .h5 model is stored must have a `custom_object.json` defined in the same directory which will take the key-value pairs as its value. The key references the name of custom layer object and the corresponding value points to its class.

Example

For a model which uses `CustomLayer tf.keras Layer` class present in the file `custom_layers.py` in source code, the contents of `custom_object.json` metadata will look like this:

```
{
                "CustomLayer": "custom_layers.CustomLayer",
}
```

AutoAI Overview (Version 2.0.0-1)

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

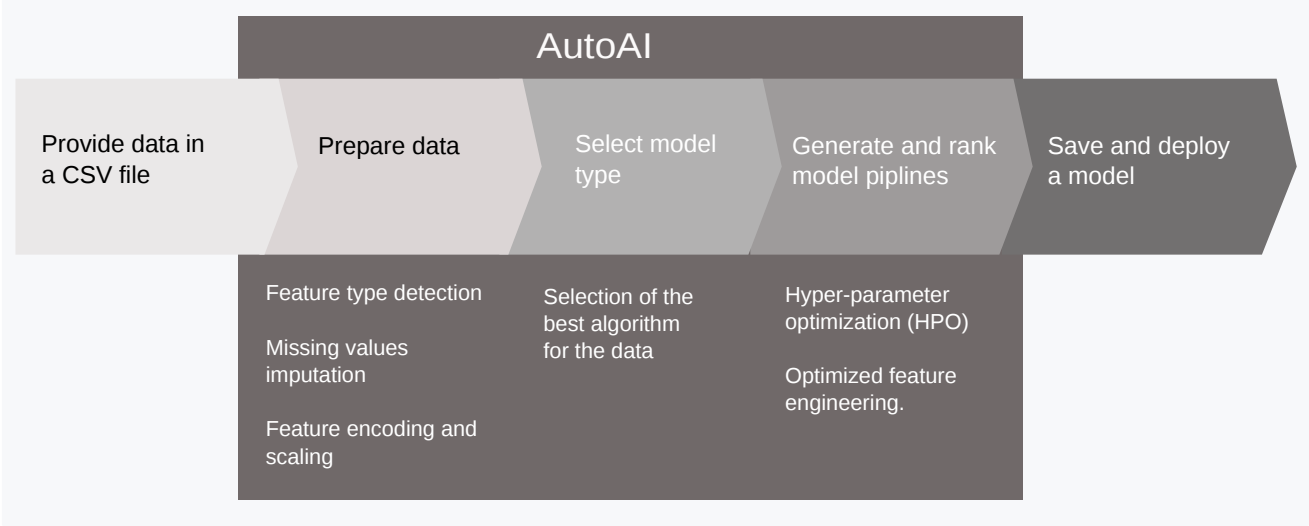
Data format
Tabular: CSV files, with comma (,) delimiter

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 2.0.0-1)

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.

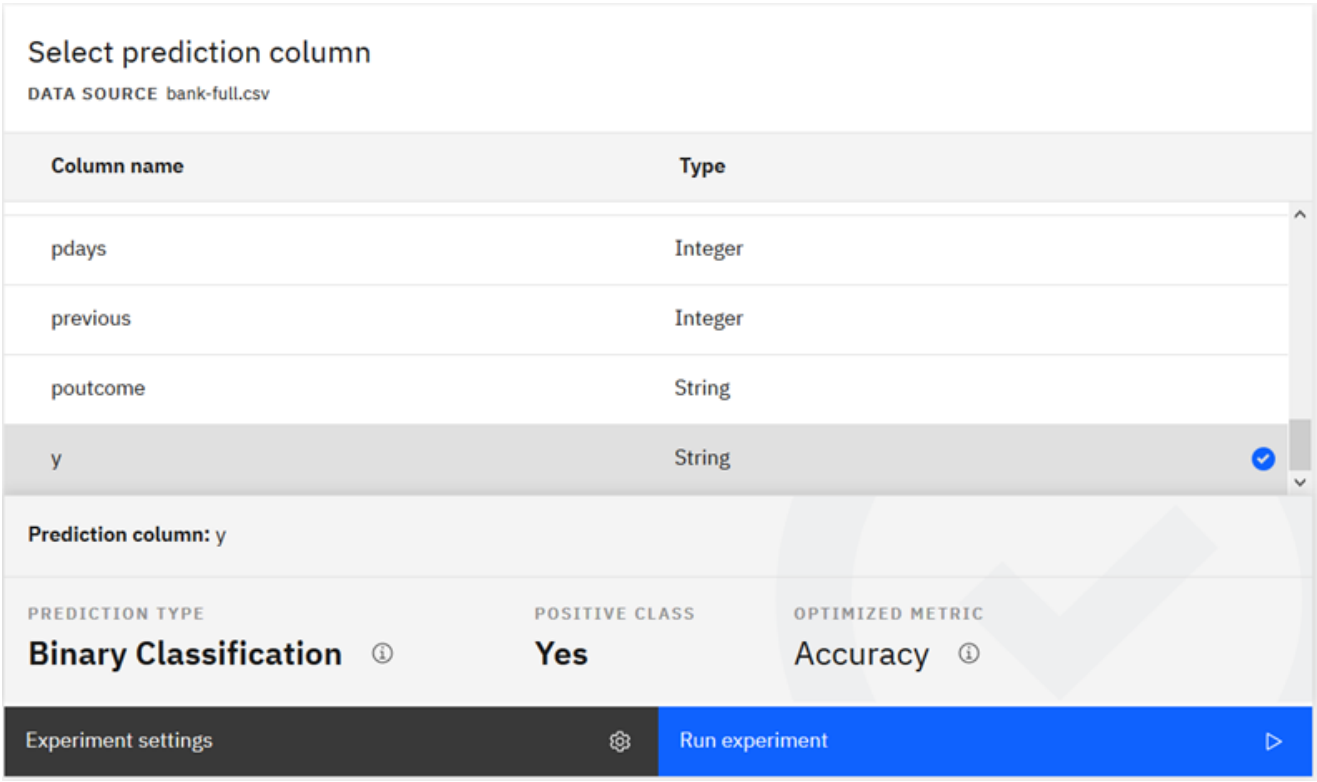
Define the experiment

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**.

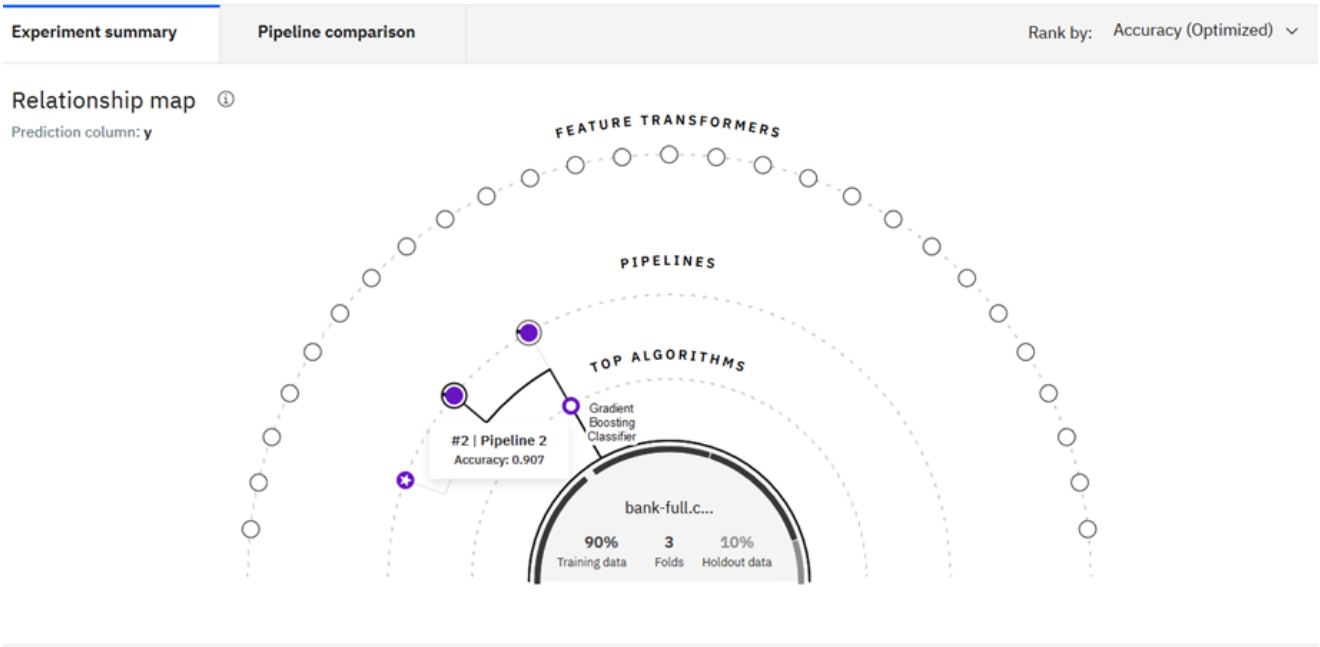
1. Select **y** as the column to predict. You can see that when you choose a column to predict, AutoAI selects a model type that matches the data. AutoAI analyzes your data and determines that the **y** column contains Yes/No information, making this data suitable for a binary classification model.
2. Click **Run experiment**.



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

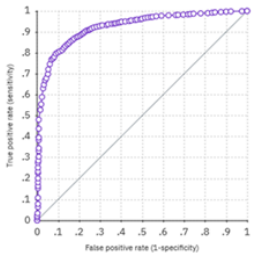
Expand a pipeline to view details about it.

	Rank ↑	Name	Algorithm	Accuracy (Optimized)	Enhancements	Build time
Collapse	★ 1	Pipeline 1	Gradient Boosting Classifier	0.907	*None*	00:01:04

Model evaluation measures

	Cross validation score	Holdout score
Accuracy	0.907	0.911
Average precision	0.602	0.614
F1	0.519	0.537
Log loss	0.213	0.211
Normalized Gini Coefficient	0.405	0.411
Precision	0.654	0.687
Recall	0.431	0.440
ROC AUC	0.924	0.923

ROC Curve



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:

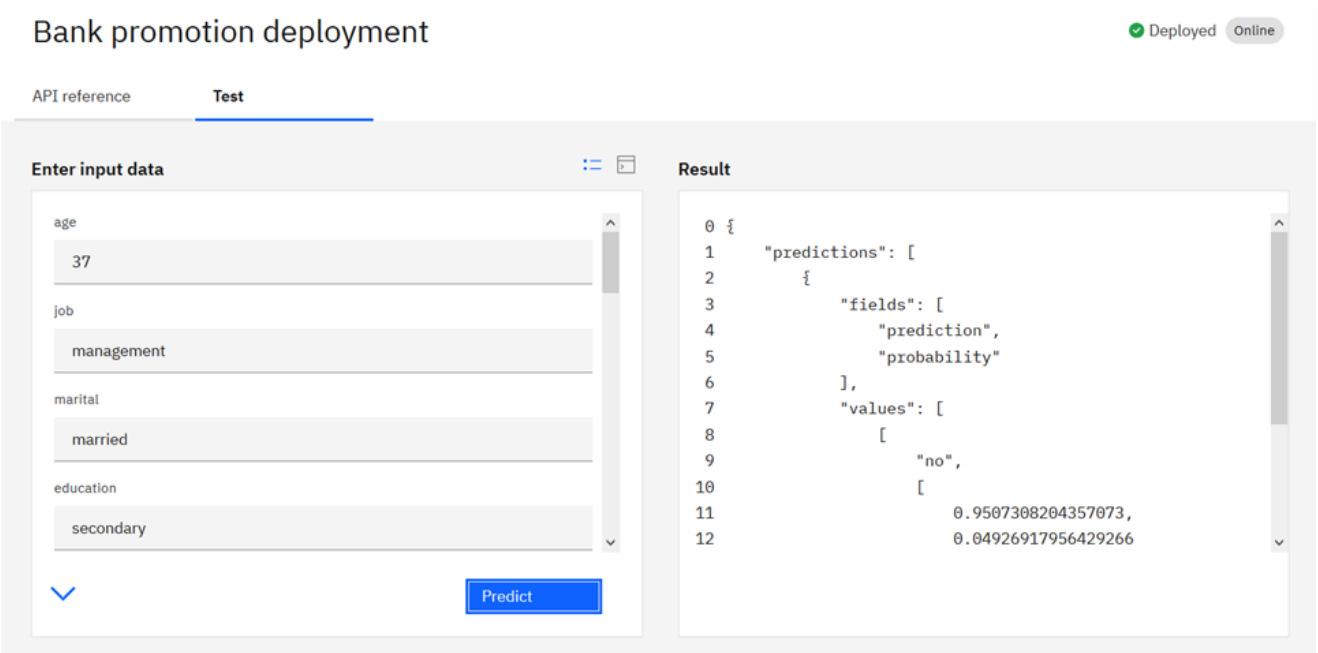
1. Click the deployment icon  next to the model name.
2. In the page that opens, fill in the fields:
   - Specify a name for the deployment.
   - Select “Online” as the **Deployment type**.
   - Click **Create**.

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:



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 2.0.0-1)

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 or files. Where possible, AutoAI will transform the data and impute missing values.

Notes:

- 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 and click **Add to project**.
2. 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.
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 2.0.0-1)

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.

The notebook allows you to review the Scikit-Learn source code for the trained model in a notebook.

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** from the action menu for the pipeline, then **Save as notebook**.
3. Name your notebook, add an optional description, and save it.

Notes:

- The notebook uses Python 3.7 and `ibm_watson_machine_learning` software development kit.

- Notebook code generated using AutoAI will execute successfully. If code is modified or reordered, there is no guarantee it will successfully execute.

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 a Scikit-learn library. In the notebook, you can view:

- An authentication template so you can enter your own credentials.
- 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.

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.

Create a sample notebook

To see for yourself what an AutoAI-generated notebook looks like:

1. Follow the steps in AutoAI tutorial.
2. After the experiment runs, choose a pipeline, then click **Save** and **Save as notebook**.
3. Name and save the notebook.
4. Open the resulting notebook in the notebook editor and review the code.

Reviewing the code

For details on the methods used in the code, see Autoai-libs.

Using autoai-lib for Python (Version 2.0.0-1)

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 cat_imp_strategy=’most_frequent’
`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.
`missing_values_reference_list`	reference list of missing values in the input numpy array X
`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.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

Option	Description
missing_values	reference list of missing values
filling_values	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)
```

known_values_list	reference list of lists of known values for each column
filling_values	special value assigned to unknown values
missing_values_reference_list	reference list of missing values

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:

```
autoai_libs.transformers.exportable.boolean2float(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.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

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 compatibiity 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 compatibiity 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.)

Option	Description
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
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	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
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.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 2.0.0-1)

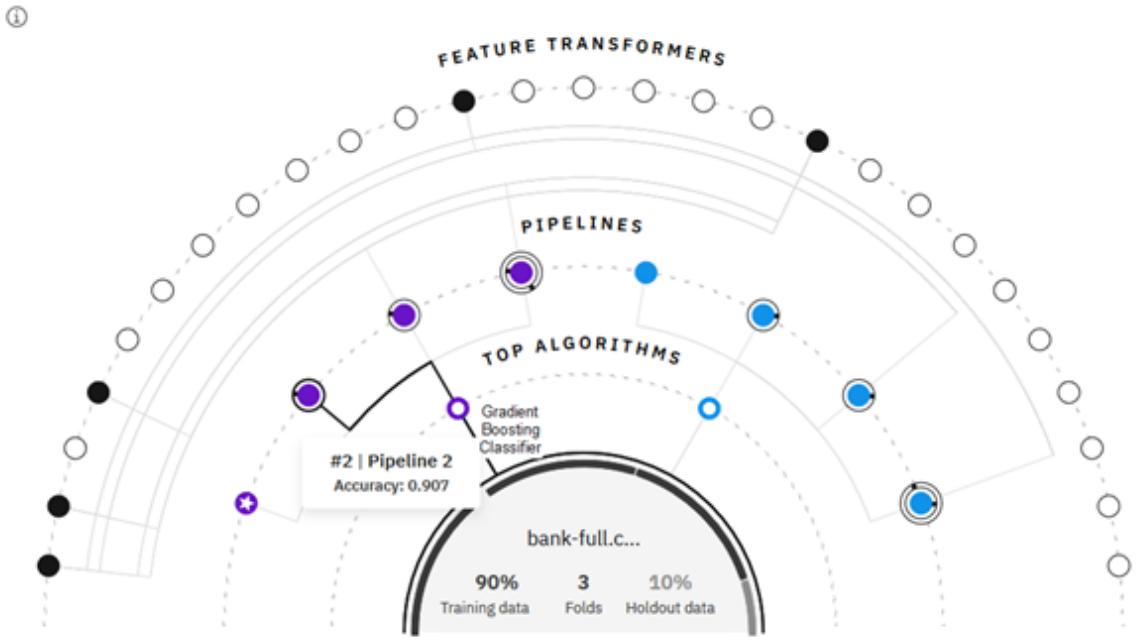
AutoAI automatically prepares data, applies algorithms, and attempts to build model pipelines best suited for your data and use case. This topic describes how to evaluate the model pipelines.

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 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

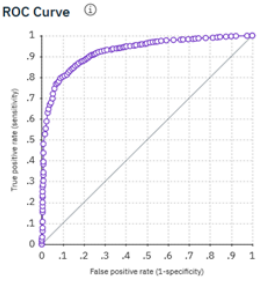
You can evaluate the pipelines as follows:

- 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.

2 Pipeline 2 Gradient Boosting Classifier **0.907** HPO-1 00:02:15 Save as

Model evaluation measures

	Cross validation score	Holdout score
Accuracy	0.907	0.911
Average precision	0.602	0.614
F1	0.519	0.537
Log loss	0.213	0.211
Normalized Gini Coefficient	0.405	0.411
Precision	0.654	0.687
Recall	0.431	0.440
ROC AUC	0.924	0.923



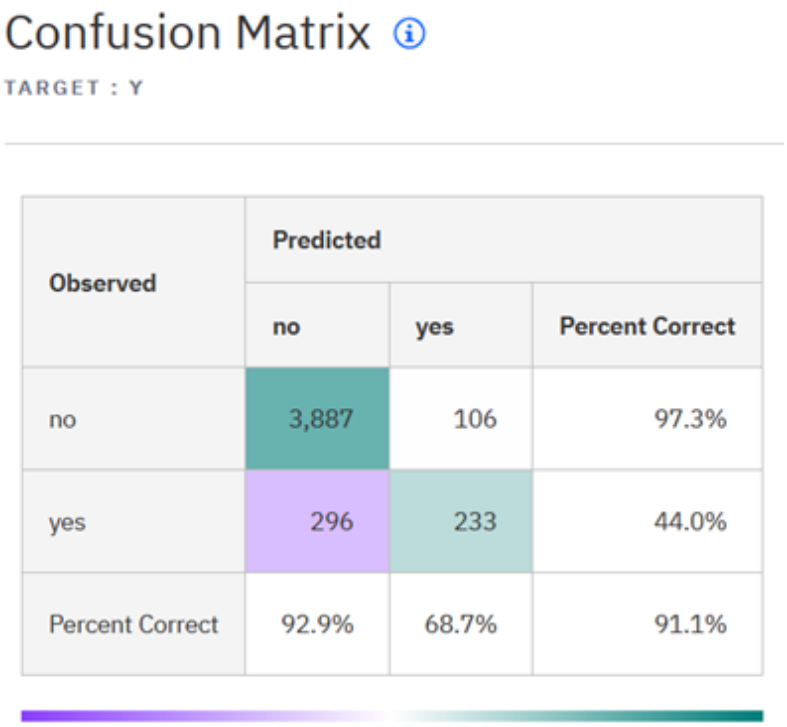
Viewing the confusion matrix

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:



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 2.0.0-1)

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.
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 *micro* 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 algortihms, 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
Principle 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.

Name	Code	Function
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	cbrt	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	sqrt	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	square	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.
Feature Agglomeration	featureagglomeration	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.

Name	Code	Function
Isolation Forest	isoforestanomaly	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 2.0.0-1)

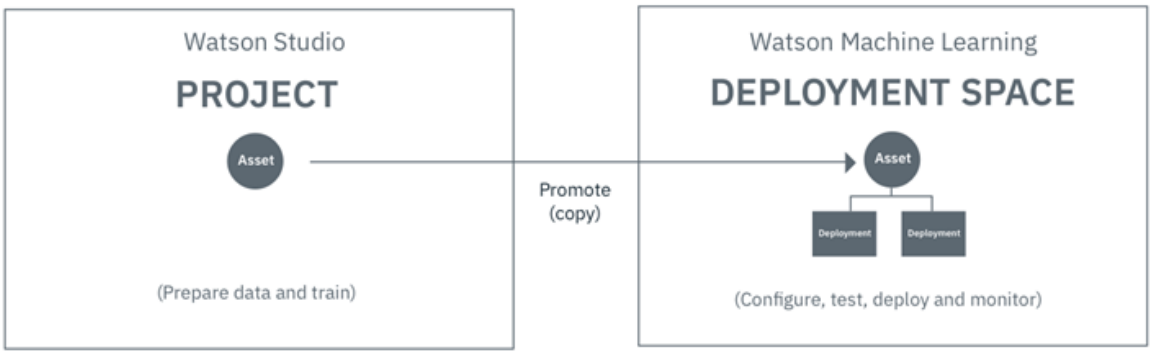
Deployment is the final stage of the lifecycle of a model or script, where you run your models and code. Watson Machine Learning provides the tools you need to deploy an asset, such as an SPSS modeler flow, a machine learning model, or a function, depending on what tools are configured for your system.

Following deployment, you can use model management tools to evaluate your models. IBM Watson OpenScale tracks and measures outcomes from your AI models, and helps ensure they remain fair, explainable, and compliant. Watson OpenScale also detects and helps correct the drift in accuracy when an AI model is in production

Service The Watson Studio, Watson Machine Learning, Watson OpenScale, and other supplemental services are not available by default. An administrator must install these services on the IBM Cloud Pak for Data platform. To determine whether a service is installed, open the Services catalog and check whether the service is enabled

Deployment overview

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 an asset to a space, components required for a successful deployment, such as a training library, model definition, other dependent assets, 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.

For models and function code, you can also create a deployment programmatically 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.

Deployment spaces (Version 2.0.0-1)

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

Deployment spaces allow you to create deployments for machine learning models and functions and view and manage all of the activity and assets for the deployments, including data connections, data refinery flows, 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 **Model from file**.
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 **Create 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 2.0.0-1)

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.

Notes:

- Depending on your configuration and the type of data connection, large data sets, typically more than 2GB, can time-out when you promote them to a space or catalog.
- Although you can promote any kind of data connection to a space, where you can 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, 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 Watson Studio 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 acessing 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 2.0.0-1)

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.

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		✓	✓
Online score	✓	✓	✓
Update assets		✓	✓
Import assets		✓	✓
Deploy assets		✓	✓
Remove assets		✓	✓
Remove deployments			✓
View spaces/members	✓	✓	✓
Delete space			✓

Creating deployments (Version 2.0.0-1)

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 Also called Web service, 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 2.0.0-1)

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 **Create 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 (Version 2.0.0-1)

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:

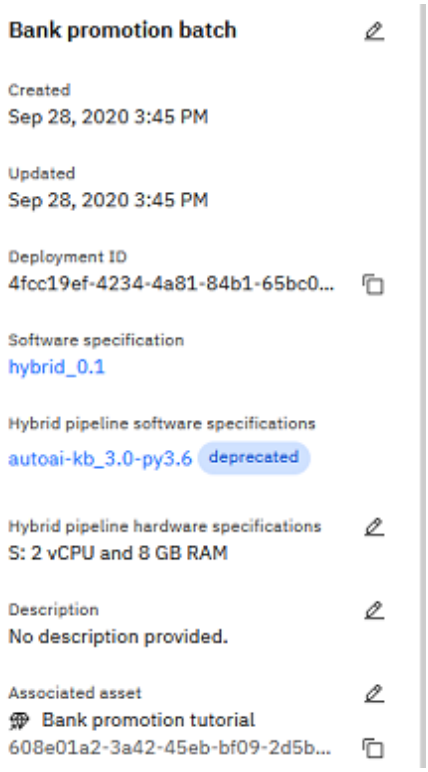
```
{"input_data":[{
        "fields":
["PassengerId","Pclass","Name","Sex","Age","SibSp","Parch","Ticket","Fare","Cabin","
Embarked"],
        "values": [[1,3,"Braund, Mr. Owen Harris",0,22,1,0,"A/5
21171",7.25,null,"S"],
                   [4,1,"Winselt, Mr. Leo Brown",1,65,1,0,"B/5
200763",7.50,null,"S"]]
}]}
```

Creating a batch deployment

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*, the deployment creation is complete.

Viewing deployment details

Click the name of a deployment to view the details.



You can view the configuration details such as hardware and software specifications. You can also get the deployment ID, which you can use in API calls from an endpoint. For details, see Looking up a deployment endpoint.

Create a batch job from a deployment

1. Click the name of a deployment, then click **Create job** to configure how to run the deployment.
2. Define the details for the job, such as name and an optional description for the job.
3. (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.
4. 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.
5. 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.

6. (Optional) If you are deploying a Python script, you can enter environment variables to pass parameters to the job.
7. 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.

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 or a Storage volume.

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 a supported database 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 an SPSS model using a database connection, make sure the modeler stream Input and Output nodes are Data Asset nodes. In SPSS Modeler, the Data Asset nodes must be configured with the table names that will be used later for job predictions. Set the nodes and table names before you save the model to Watson Machine Learning. While configuring the Data Asset nodes, choose the table name from the Connections; choosing a Data Asset that is created in your project is currently not supported.

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 2.0.0-1)

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.

Using data from a Cloud Object Storage connection

1. Create a connection to IBM Cloud Object Storage by adding a **Connection** to your project or space and selecting Cloud Object Storage (infrastructure) as the connection type. Provide the secret key, access key and login URL.
2. Add input and output files to the deployment space as connected data using the COS connection you created.

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 sepcifications 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
- Pytorch
- Python function

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
  - DB2

File Formats: csv,xls,sas,sav

Notes:

- SPSS jobs support multiple data source inputs and a single output. If the schema is not provided in the model metadata at the time of saving the model, you must enter “id” manually and select data asset in the Watson Studio UI for each connection. If the schema is provided in model metadata, “id” names are auto populated using metadata. You just select the data asset for the corresponding “id”s in Watson Studio. For details, see Using multiple data sources for an SPSS job.
- 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 supported databases can be created in the output_data_references only when the input_data_references also refers to one of these sources.
- Note that table names provided in input and output data references are ignored. Table names referred in SPSS model stream will be used during the batch deployment.

- The `environment_variables` parameter of deployment jobs is not applicable

- 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
            }
        }
    }
```

Supported combinations of input and output sources

You must specify compatible sources for the SPSS Modeler flow input, the batch job input, and the output. If you specify an incompatible combination of types of data sources, you will get an error trying to execute the batch job.

These combinations are supported for batch jobs:

SPSS model stream input/output	Batch deployment job input	Batch deployment job output
File	Local/managed or referenced data asset (file)	Remote data asset (file) or name
Database	Remote data asset (database)	Remote data asset (database)

Specifying multiple inputs

If you are specifying multiple inputs for an SPSS model stream deployment with no schema, specify an ID for each element in input_data_references.

For details, see Using multiple data sources for an SPSS job.

In this example, when you create the job, provide three input entries with ids: “sample_db2_conn”, “sample_teradata_conn” and “sample_googlequery_conn” and select the required connected data for each input.

```
{
"deployment": {
    "href": "/v4/deployments/<deploymentID>"
  },
  "scoring": {
          "input_data_references": [{
               "id": "sample_db2_conn",
               "name": "DB2 connection",
               "type": "data_asset",
               "connection": {},
               "location": {
                     "href": "/v2/assets/<asset_id>?space_id=<space_id>"
               },
           },
           {
               "id": "sample_teradata_conn",
               "name": "Teradata connection",
               "type": "data_asset",
               "connection": {},
               "location": {
                     "href": "/v2/assets/<asset_id>?space_id=<space_id>"
               },
           },
           {
               "id": "sample_googlequery_conn",
               "name": "Google bigquery connection",
               "type": "data_asset",
               "connection": {},
               "location": {
                     "href": "/v2/assets/<asset_id>?space_id=<space_id>"
               },
           }],
          "output_data_references": {
                "id": "sample_db2_conn"
                 "type": "data_asset",
                 "connection": {},
                 "location": {
```

```
                    "href": "/v2/assets/<asset_id>?space_id=<space_id>"
                },
            }
}
```

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

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

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.

Type: inline

Notes:

- `environment_variables` parameter of deployment jobs is not applicable
- Make sure the output is structured to match the output schema described in Execute a synchronous deployment prediction.

Using multiple data sources for an SPSS Modeler job

In an SPSS Modeler flow, it's common to have multiple import and export nodes nodes, where multiple import nodes can be fetching data from one or more relational databases. This section describes the process for using Watson Machine Learning to create an SPSS Modeler batch job with multiple data sources from relational databases.

Note: This capability is available starting in IBM Cloud Pak for Data 3.0.1. The examples in this section use IBM Db2 and IBM Db2 Warehouse, referred to in examples as *dashdb*.

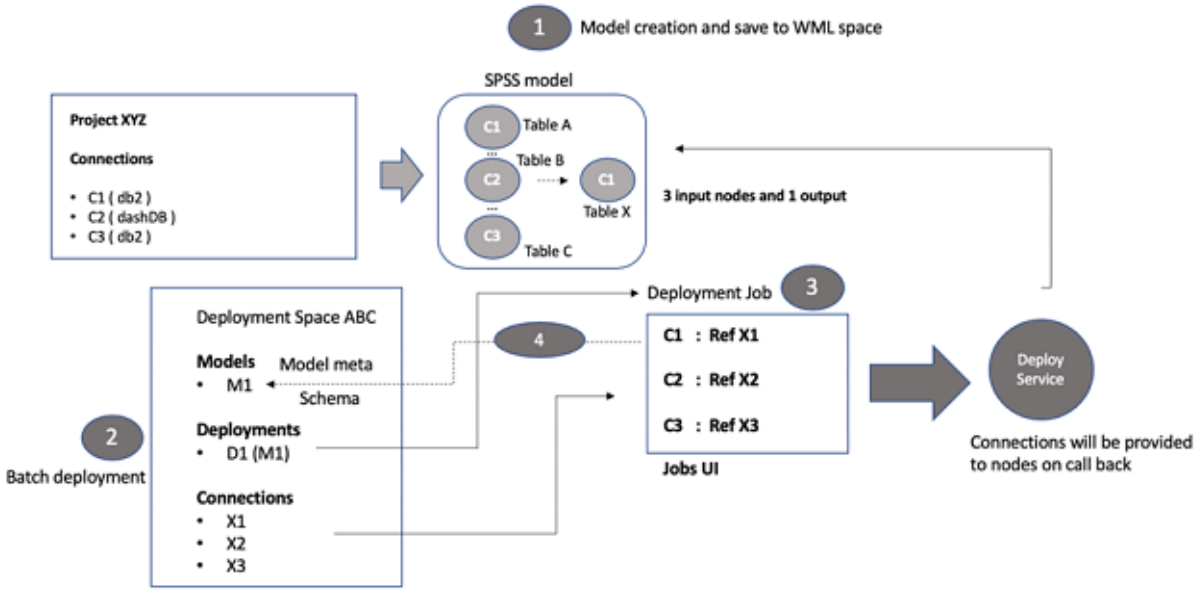
Connecting to multiple relational databases as input to a batch job

The number of import nodes in an SPSS Modeler flow can vary. You might use as many as 60 or 70. However, the number of unique connections to databases in these cases will normally be just a few, though the table names accessed through the connections will vary. Rather than specifying the details for every table connection, the approach described here focuses on the database connections. Therefore, the batch jobs will accept a list of data connections or references by *node name* that are mapped to connection names in the SPSS Modeler flow's import nodes.

For example, assume that if a flow has 30 nodes, only 3 unique database connections are used to connect to 30 different tables. In this case, you submit 3 connections (C1, C2, and C3) to the batch job. C1, C2, and C3 are connection names in the import node of the flow and the *node name* in the input of the batch job.

When a batch job runs, the data reference for a node is provided by mapping the *node name* with the *connection name* in the import node. This example illustrates the steps for creating the mapping.

The following diagram shows the flow from model creation to job submission:

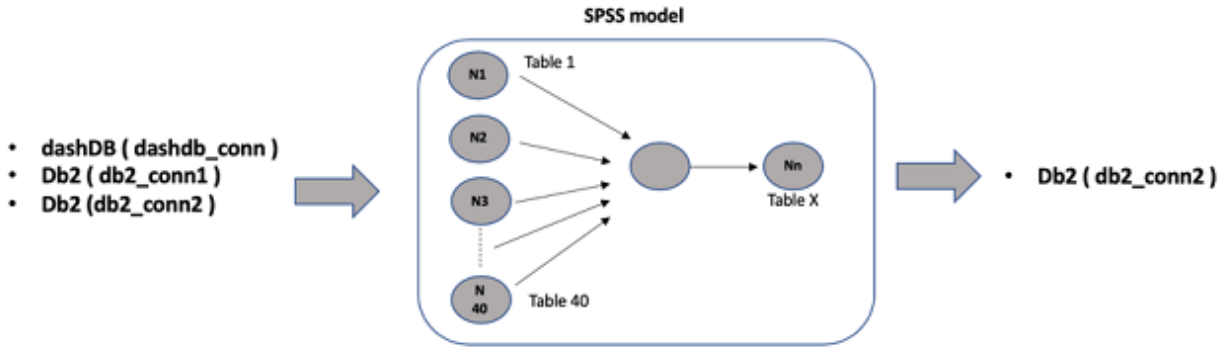


Limitation: Although the connection reference for a node in a flow will be overridden by the reference received from the batch job, the table name in the import or export node will not be overridden.

Deployment scenario with example

In this example, an SPSS model is built using 40 import nodes and a single output. The model has the following configuration:

- Connections to 3 databases: 1 Db2 Warehouse (dashDB) and 2 Db2.
- The import nodes are read from 40 tables (30 from Db2 Warehouse and 5 each from the Db2 databases).
- A single output table is written to a Db2 database.



Example

These steps demonstrate how to create the connections and identify the tables.

1. Create connected data assets in your Watson Studio project.

   To run the SPSS Modeler flow, you start in your Watson Studio project and create a connection for each of the three databases your model connects to. You then configure each import node in the flow to point

to a table in one of the connected databases.

For this example, the database connections in the project are named `dashdb_conn`, `db2_conn1` and `db2_conn2`.

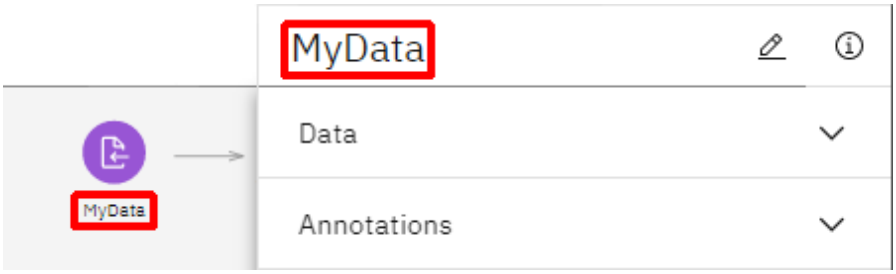
2. Configure Data Asset import nodes in your SPSS Modeler flow with connected data assets.

   Configure each node in the flow to reference one of the three connected data assets you created (`dashdb_conn`, `db2_conn1`, and `db2_conn2`), then specify a table for each node.

   Note: You can change the name of the connection at the time of the job run, but the table names you select in the flow are referenced when the job runs; you can't overwrite or change them.

3. Save the SPSS model to the Watson Machine Learning repository.

   For this example, it's helpful to provide the input and output schema when saving the model, as it simplifies the process of identifying each input when you create and submit the batch job in the Watson Studio user interface. Note that connections referenced in the Data Asset nodes of the SPSS Modeler flow must be provided in the *node name* field of the input schema. To find the *node name*, double-click the Data Asset import node in your flow to open its properties:



   Note: SPSS models saved without schemas are still supported for jobs, but you must enter *node name* fields manually and provide the data asset when you submit the job.

   This code sample shows how to save the input schema when you save the model.

```
(Endpoint: POST /v4/models)
 {
       "name": "SPSS Drug Model",
       "label_column": "label",
       "type": "spss-modeler_18.1",
       "runtime": {
               "href": "/v4/runtimes/spss-modeler_18.1"
        },
        "space": {
               "href": "/v4/spaces/<space_id>"
        },
        "schemas": {
                  "input":  [ { "id": "dashdb_conn", "fields": [] },
                                    { "id": "db2_conn1 ", "fields": [] } ,
                                    { "id": "db2_conn2", "fields": [] } ],
                   "output": [{ "id": "db2_conn2 ","fields": [] }]
          }
 }
```

   Note: The number of fields in each of these connection doesn't matter. They're not validated or used. What's important is the number of connections that are used.

4. Create the batch deployment for the SPSS model.

The creation of the batch deployment job has not changed; there's nothing specific to this use case while the SPSS batch deployment is created. You can submit the deployment request with the model created in the previous step.

5. Submit SPSS batch jobs.

   You can submit a batch job from the Watson Studio user interface or by using the REST API. If the schema is saved with the model, the Watson Studio user interface makes it simple to accept input from the connections specified in the schema. Because you already created the data connections, you can select a connection for each *node name* field that displays in the Watson Studio user interface as you define the job.

   Note that the names of the connection created at the time of job submission can be different from the one used at the time of model creation, but the respective one must be assigned to the *node name* field

Submitting a job when there is no schema

If the schema isn't provided in the model metadata at the time the model is saved, you must enter the *import node name* manually and select the data asset in the Watson Studio user interface for each connection. Connections referenced in the Data Asset import nodes of the SPSS Modeler flow must be provided in the *node name* field of the import/export data references.

This code sample demonstrates how to specify the connections for a job submitted using the REST API (Endpoint: /v4/deployment_jobs).

```
```
            {
                        "deployment": {
                                    "href": "/v4/deployments/<deploymentID>"
                         },
                         "scoring": {
                                      "input_data_references": [
                                        {
                                               "id": "dashdb_conn",
                                              "name": "dashdb_conn",
                                              "type": "data_asset",
                                              "connection": {},
                                              "location": {
                                                        "href":
"/v2/assets/<asset_id>?space_id=<space_id>"
                                               },
                                               "schema": {}
                                    },
                                    {
                                               "id": "db2_conn1 ",
                                               "name": "db2_conn1 ",
                                               "type": "data_asset",
                                               "connection": {},
                                               "location": {
                                                         "href":
"/v2/assets/<asset_id>?space_id=<space_id>"
                                                },
                                                "schema": {}
                                    },
                                    {
                                               "id": "db2_conn2 ",
                                               "name": "db2_conn2",
                                               "type": "data_asset",
                                               "connection": {},
                                               "location": {
                                                         "href":
```
```

```
"/v2/assets/<asset_id>?space_id=<space_id>"
                                                    },
                                                    "schema": {}
                                            }],
                                            "output_data_reference": {
                                                    "id": "db2_conn2"
                                                   "name": "db2_conn2",
                                                    "type": "data_asset ",
                                                    "connection": {},
                                                    "location": {
                                                              "href":
"/v2/assets/<asset_id>?space_id=<space_id>"
                                                    },
                                                    "schema": {}
                                            }
                    }     ```
```

Managing deployment jobs (Version 2.0.0-1)

A job is a way of running a batch deployment, Data Refinery flow, Streams flow, 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
- Deleting jobs

You create a Data Refinery job from the Assets page:

1. Click **Create job** from the Actions menu next to the name of the Data Refinery flow.
2. Complete the job details and run the job. Output is saved to the specified output file.

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.

Deploying a script

Once a script is copied to a deployment space, you can deploy it for use.

Notes for deploying a script

Supported types for scripts are Python scripts.

- Batch deployment is the only supported deployment type.
- Your software specification is included when the script is promoted from a project. For details on software specifications, see Specifying a model type and software specification.

For details on supported input and output types and setting environment variables, see Batch deployment details.

Deploying a script from a space

A batch deployment processes input data from a file and writes the output to a file.

Before you begin

1. Save or promote to deployment space.
2. Add the input file for the batch deployment to the space.

For details on making assets available in a space, see Deployment spaces.

Creating the batch deployment

1. From the deployment space, click the name of the script you want to deploy.

2. From the **Deployments** tab, click **Create deployment**.
3. Choose **Batch** as the deployment type and enter a name for your deployment.
4. Choose a configuration 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. Select the input data source you promoted or uploaded to the space and provide a name for the output file that will contain the results.
8. Choose the environment for running the script. **Attention:** The default environment is `Default spark python3.7`. You must manually override this with the correct environment for your script.
9. (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.
10. 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 Core ML deployment (Version 2.0.0-1)

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. For details, see Supported frameworks.

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 **Create 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.

Managing deployed assets

After you deploy assets, you can manage and update them to make sure they are performing well and to monitor their accuracy.

Here are some ways you can manage or update a deployment:

- Get the deployment endpoint URL to interact with the deployment.
- Update a deployed asset. For example, you can replace a model with a better-performing version without having to create a new deployment.
- Scale a deployment to increase availability and throughput by creating replicas of the deployment.
- Delete a deployment to remove a deployment and free up resources.

Looking up the endpoint URL of a deployment

To send payload data to a model or function deployment for analysis (for example, to classify the data, or make a prediction from the data) you need to know the endpoint URL of the deployment. This topic describes how to look up the endpoint URL of a deployment in IBM Watson Machine Learning.

There are two ways to look up the endpoint URL of a deployment:

- IBM Watson Studio deployment space
- Watson Machine Learning Python client

Option 1: From a deployment space

From the **Deployments** tab of your space, click a deployment name to look up the deployment ID for the endpoint URL.

For details, see Endpoint URLs.

Option 2: Using the Watson Machine Learning Python client

1. List the deployments by calling the function `client.deployments.list()`
2. In the row of the desired deployment, the deployment endpoint URL is listed in the “url” column

Updating a deployed asset

After you create an online or batch 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.

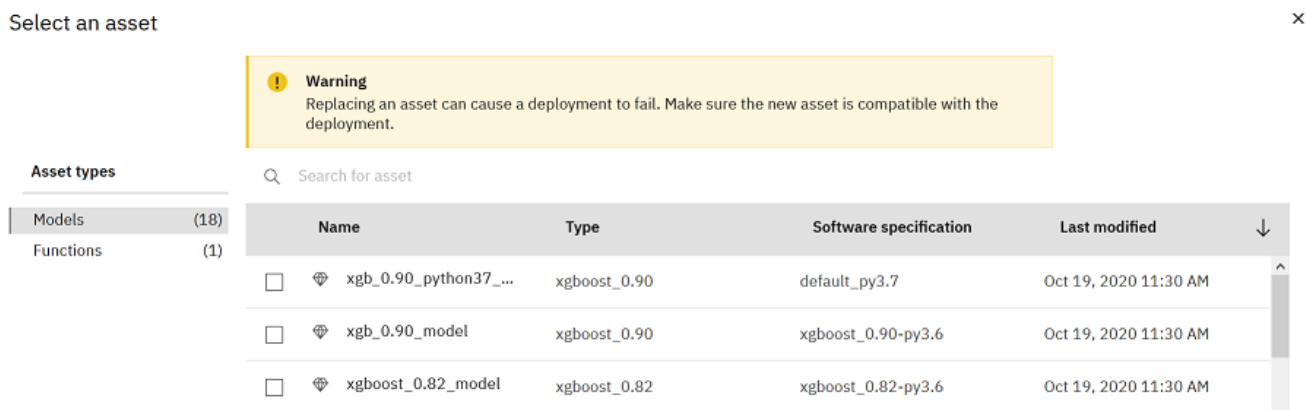
Before you update an asset, make sure these conditions are true:

- 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.

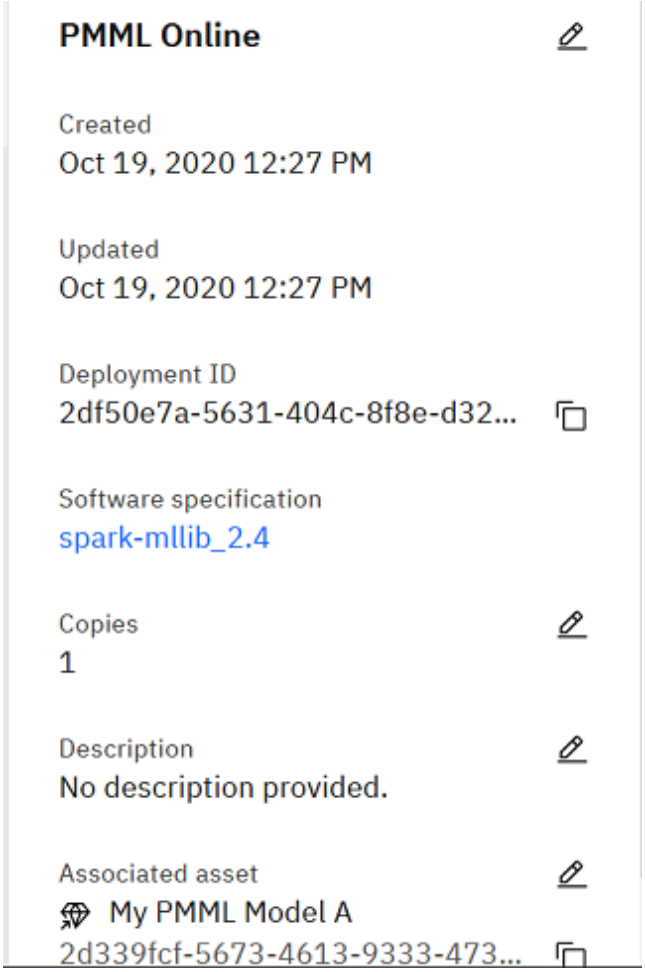
Update an asset from the deployment space

1. From the **Deployments** tab of your deployment space, click the action menu for the deployment and choose **Replace asset**.
2. Select the asset you want to replace and click **Replace**.
3. Drag and drop or upload a new asset to replace the deployed asset.
4. Wait for the notification message that confirms a successful update, then test the new deployment.



Tip: You can also update a deployment from the information sheet for the deployment.

1. Click the deployment name to view the details.
2. Click the Edit icon in the information sheet to edit the deployment.



Update an asset using the Patch API command

Use the Watson Machine Learning API `Patch` command to update any supported asset except for RShiny apps.

Use the Watson Machine Learning API `Patch` command to update any supported asset except for RShiny apps.

Use this method to patch a model for an online deployment.

```
curl -X PATCH 'https://us-south.ml.cloud.ibm.com/ml/v4/models/:model_id?space_id=
<string>&project_id=<string>&version=2020-09-01' \n--data-raw '[
    {
        "op": "<string>",
        "path": "<string>",
        "value": "<object>"
    },
    {
        "op": "<string>",
        "path": "<string>",
        "value": "<object>"
    }
]'

For example, patch a model 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.”
    }, ```
```

Scaling a deployment

When you create an online deployment for a model, function, or Shiny app from a deployment space or programmatically, a single copy of the asset is deployed by default. To increase scalability and availability, you can increase the number of copies (replicas) by editing the configuration of the deployment. Additional replicas allow for a larger volume of scoring requests.

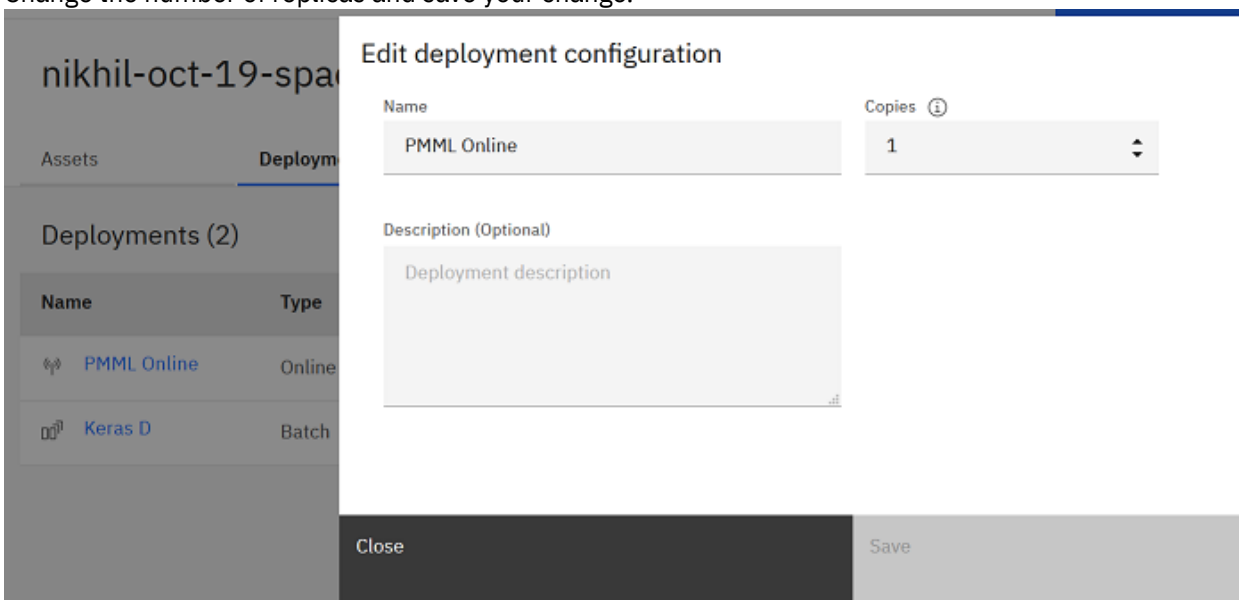
Deployments can be scaled in the following ways:

- Update the configuration for a deployment in a deployment space.
- Programmatically, using the Watson Machine Learning Python client library, or the Watson Machine Learning REST APIs.

Increase the number of replicas of an online deployment from a space

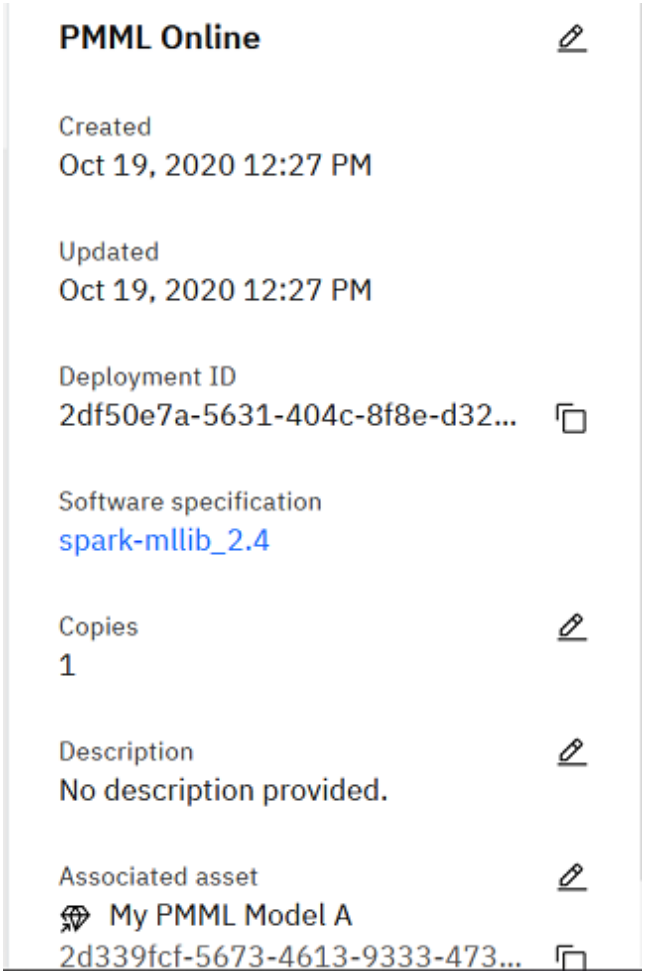
1. Click to open the **Deployment** tab of your deployment space.
2. Click **Edit configuration** on the action menu for the deployment name.

3. Change the number of replicas and save your change.



Tip: You can also update the number of replicas from the information sheet for the deployment.

1. Click the deployment name to view the details.
2. Click the Edit icon in the information sheet to edit the number of copies.



Increase the number of replicas of a deployment programmatically

To view or run a working sample of scaling a deployment programmatically, you can view a notebook example.

Python example

This example uses the Python client to set the number of replicas 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.

REST API example

```
curl -k -X PATCH -d '[ { "op": "replace", "path": "/hardware_spec", "value": {
"name": "S", "num_nodes": 2  } } ]' <Deployment end-point URL>
```

You must specify a name for the `hardware_spec` value, but the argument is not applied for scaling.

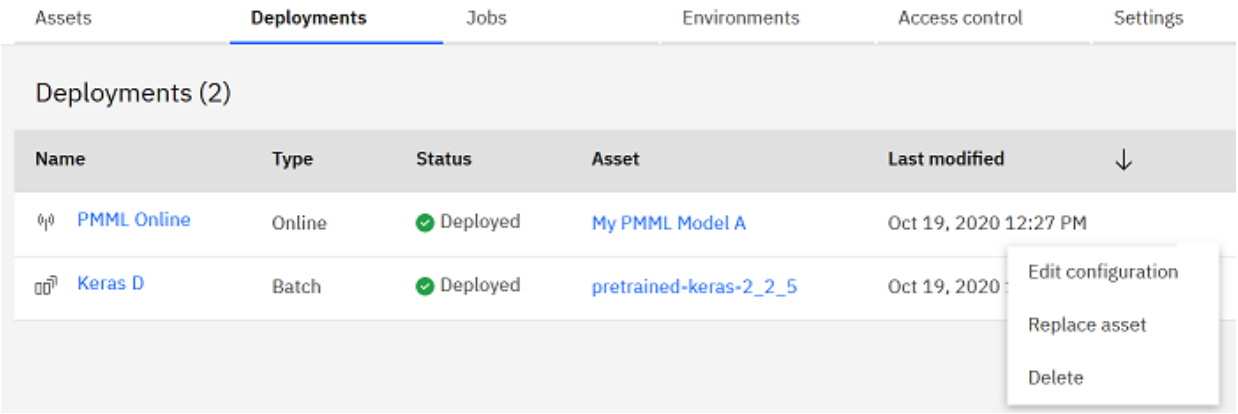
Deleting a deployment

When you are done with a deployment, it is a best practice to delete it to free up resources. You can delete a deployment from a deployment space, or programmatically, using the Python client or Watson Machine Learning APIs.

Deleting a deployment from a space

To remove a deployment:

1. Open the **Deployments** page of your deployment space.
2. Choose **Delete** from the action menu for the deployment name.



Deleting a deployment using the Python client

Use the following method to delete the deployment.

```
client.deployments.delete(deployment_uid)
```

Returns a `SUCCESS` message. To check that the deployment was removed, you can list the deployments and make sure it is no longer listed.

```
client.deployments.list()
```

Returns:

```
----  ----  -----  -------  -------------
GUID  NAME  STATE  CREATED  ARTIFACT_TYPE
----  ----  -----  -------  -------------
```

Deleting a deployment using the REST API

Use the Delete method for deleting a deployment:

```
DELETE /ml/v4/deployments/{deployment_id}
```

For example:

```
curl --location --request DELETE 'https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id?space_id=
<string>&version=2020-09-01'
```