IBM SPSS Modeler 18.1.1
Python 脚本编制和自动化指南

注释

在使用本资料及其支持的产品之前，请阅读第 347 页的『声明』中的信息。

产品信息

本版本适用于 IBM SPSS Modeler V18.1.1 及所有后续发行版和修订版，直到在新版本中另有声明为止。

目录

第 1 章 脚本编写和脚本编写语言

脚本编制概述

IBM® SPSS® Modeler 中的脚本编写是用于在用户界面上实现过程自动化的强大工具。您使用鼠标或键盘进行的操作,借助脚本同样可以完成，而且使用脚本可以自动化那些手动执行将造成大量重复操作且高耗时的任务。

脚本的作用包括：

- 限制在流中执行节点的特定顺序。
- 设置节点属性并使用 CLEM （表达式操作控制语言）的子集来执行派生。
- 指定通常包含用户交互的操作的自动执行顺序，例如您可以构建一个模型，然后对其进行测试。
- 设置需要实际用户交互的复杂过程，例如需要重复模型生成和测试的交叉验证步骤。
- 设置流操纵过程 - 例如,您可以提取一个模型训练流，运行它，然后自动生成相应的模型测试流。

本章提供流级脚本、独立脚本以及 IBM SPSS Modeler 用户界面超节点内脚本的高级说明和示例。有关脚本编写语言、语法和命令的更多信息，请参阅章后的章节。

注：

您无法导入和运行在 IBM SPSS Modeler 中的 IBM SPSS Statistics 中创建的脚本。

脚本类型

IBM SPSS Modeler 使用三种类型的脚本：

- **流脚本**存储为流属性然后和指定流一起保存和装入。例如，可以编写自动化训练和应用模型块流程的流脚本。您还可以指定何时执行特定流，脚本应代替流画布内容运行。
- **独立脚本**不与保存在外部文本文件中的所有特定流关联。例如，可以使用独立脚本同时操作多个流。
- **超节点脚本**存储为超节点流属性。超节点只在终端超节点中可用。您可以使用超节点脚本控制超节点内容的执行序列。对于非终端（源或过程）超节点，可以为超节点定义属性或定义这种超节点直接在流脚本中包含的节点。

流脚本

脚本可用于定制特定流中的操作并与该流一起保存。流脚本可用于指定某个流中终端节点的特定执行顺序。可以使用"流脚本"对话框来编辑与当前流一起保存的脚本。

从流属性对话框访问流"脚本"选项卡：

1. 从**工具**菜单中，选择：

   流属性 > **执行**
2. 单击**执行**选项卡以处理当前流的脚本。

使用"流脚本"对话框顶部的工具栏图标可以执行下列操作：

- 将先前存在的独立脚本的内容导入窗口中。

- 将脚本保存为文本文件。
- 打印脚本。
- 追加缺省脚本。
- 编辑脚本（撤销、剪切、复制、粘贴及其他常见的编辑功能）。
- 执行整个当前脚本。
- 执行某个脚本中的选定行。
- 在执行期间停止脚本。（只有在脚本处于运行状态的情况下，才会启用此图标。）
- 检查脚本的语法，如果发现任何错误，就将其显示在对话框的下部窗格中复查。

注：从 V16.0 开始，SPSS Modeler 使用 Python 脚本语言。所有低于 16.0 的版本使用 SPSS Modeler 独有的脚本语言，现在称为旧脚本编制。根据您要处理的脚本的类型，请在**执行**选项卡上选择**缺省（可选脚本）**执行方式，然后选择 **Python** 或**旧**。

可以指定当执行流时是否应运行此脚本。要在每次执行流时按照脚本的执行顺序运行此脚本，可以选择**运行此脚本**。此设置为快速构建模型提供流一级的自动化。但是，缺省设置为在执行流的过程中忽略此脚本。即使选择选项**忽略此脚本**，也可以直接从此对话框运行脚本。

脚本编辑器提供了下列功能，这些功能有助于脚本编写：
- 语法突出显示；将突出显示关键字、文字值（例如字符串和数字）以及注释。
- 行编号。
- 块匹配；当光标处于程序块的开始位置时，还将突出显示相应的结束块。
- 建议的自动补全。

可以使用 IBM SPSS Modeler 显示首选项来定制语法突出显示器使用的颜色和文本样式。要访问显示首选项，请选择**工具** > **选项** > **用户选项**，并选择**语法**选项卡。

通过从上下文菜单中选择**自动建议**或者按 Ctrl + Space，可以访问建议语法补全的列表。使用光标键在列表中上下移动，然后按 Enter 键可插入所选文本。要退出自动建议方式而不修改现有文本，请按 Esc。

调试选项卡显示调试消息，并且可以用于在执行脚本后立即对脚本状态进行评估。**调试**选项卡包含一个只读文本区域和单行输入文本字段。文本区域显示由脚本发送到标准输出或标准错误（例如，通过错误消息文本）的文本。输入文本字段将接收来自用户的输入。然后，将在对话框内最近执行的脚本上下文（称为脚本编制上下文）中对此输入进行评估。文本区域包含命令和生成的输出，以便用户能够查看命令跟踪。文本输入字段始终包含命令提示（对于旧脚本编制，此命令提示符为 -->）。

在下列情况下，将创建新的脚本编制上下文：
- 使用**运行此脚本**或**运行选定的行**来执行脚本。
- 脚本语言会发生更改。

如果创建了新的脚本编制上下文，那么将清除文本区域。

注：在脚本窗格外部执行流将不会修改此脚本窗格的脚本上下文。在脚本对话框中，将无法查看该执行过程中创建的任何变量的值。

流脚本示例：训练神经网络

在执行时，流可用于训练神经网络模型。通常，要检验模型，您可以运行建模节点以便将该模型添加到流中，建立相应的连接，然后执行"分析"节点。

借助 IBM SPSS Modeler 脚本，您可以在创建模型块之后，实现模型块测试过程的自动化。例如，以下流脚本将测试演示流 druglearn.str（在 IBM SPSS Modeler 安装下的 /Demos/streams/ 文件夹中），并可从流属性对话框（**工具 > 流属性 > 脚本**）中运行。

```
stream = modeler.script.stream()
neuralnetnode = stream.findByType("neuralnetwork", None)
results = []
neuralnetnode.run(results)
appliernode = stream.createModelApplierAt(results[0], "Drug", 594, 187)
analysisnode = stream.createAt("analysis", "Drug", 688, 187)
typenode = stream.findByType("type", None)
stream.linkBetween(appliernode, typenode, analysisnode)
analysisnode.run([])
```

以下带着重号的句子说明此脚本示例中的每一行。

- 第一行定义指向当前流的变量。
- 在第二行中，脚本将查找"神经网络"构建器节点。
- 在第三行中，脚本将创建可以在其中存储执行结果的列表。
- 在第四行中，将创建"神经网络"模型块。此模型块存储在第三行中定义的列表内。
- 在第五行中，将为此模型块创建模型应用节点并将此节点放入流画布中。
- 在第六行中，将创建称为 Drug 的分析节点。
- 在第七行中，脚本将查找类型节点。
- 在第八行中，脚本将连接第五行中在类型节点与分析节点之间创建的模型应用节点。
- 最后，执行分析节点以生成分析报告。

可以使用脚本从头开始（从空画布开始）构建并运行流。要了解有关脚本语言的更多一般信息，请参阅脚本编写语言概述。

Jython 代码大小限制

Jython 将每个脚本编译为 Java 字节码，随后由 Java 虚拟机 (JVM) 来执行。但, Java 字节码对单一字节码文件的大小施加了限制。因此，当 Jython 尝试装入字节码时，可能导致 JVM 崩溃。IBM SPSS Modeler 无法阻止发生此问题。

请确保使用正确的编码实践来编写 Jython 脚本（例如，通过使用变量或函数来计算常用中间值以最大限度减少重复代码）。如果有必要，您可能需要将代码拆分到多个源文件中或者使用模块来定义代码，因为这些模块将编译到多个独立的字节码文件中。

独立脚本

"独立脚本"对话框用于创建或编辑保存为文本文件的脚本。它显示了文件名称，提供了用于装入、保存、导入和执行脚本的实用程序。

要访问"独立脚本"对话框，请执行以下操作：

在主菜单中，选择：

工具 > **独立脚本**

对流脚本可用的工具栏和脚本语法检查选项对独立脚本同样适用。有关更多信息，请参阅第 1 页的『流脚本』主题。

独立脚本示例：保存和装入模型

独立脚本可用于流操纵。假设有两个流，第一个流创建模型并生成规则集，第二个流则通过现有数据字段，采用图示的方式对规则集进行探索。该方案的独立脚本可能具有如下形式：

```
taskrunner = modeler.script.session().getTaskRunner()

# Modify this to the correct Modeler installation Demos folder.
# Note use of forward slash and trailing slash.
installation = "C:/Program Files/IBM/SPSS/Modeler/18.1.1/Demos/"

# First load the model builder stream from file and build a model
druglearn_stream = taskrunner.openStreamFromFile(installation + "streams/druglearn.str", True)
results = []
druglearn_stream.findByType("c50", None).run(results)

# Save the model to file
taskrunner.saveModelToFile(results[0], "rule.gm")

# Now load the plot stream, read the model from file and insert it into the stream
drugplot_stream = taskrunner.openStreamFromFile(installation + "streams/drugplot.str", True)
model = taskrunner.openModelFromFile("rule.gm", True)
modelapplier = drugplot_stream.createModelApplier(model, "Drug")

# Now find the plot node, disconnect it and connect the
# model applier node between the derive node and the plot node
derivenode = drugplot_stream.findByType("derive", None)
plotnode = drugplot_stream.findByType("plot", None)
drugplot_stream.disconnect(plotnode)
modelapplier.setPositionBetween(derivenode, plotnode)
drugplot_stream.linkBetween(modelapplier, derivenode, plotnode)
plotnode.setPropertyValue("color_field", "$C-Drug")
plotnode.run([])
```

注：要了解有关脚本语言的更多一般信息，请参阅脚本编写语言概述。

独立脚本示例：生成特征选择模型

首先打开一个空画布，在此示例中将构建一个流,该流生成一个特征选择模型，应用此模型并创建一个表,该表包含有对于指定目标而言重要性最高的 15 个字段。

```
stream = modeler.script.session().createProcessorStream("featureselection", True)

statisticsimportnode = stream.createAt("statisticsimport", "Statistics File", 150, 97)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/customer_dbase.sav")

typenode = stream.createAt("type", "Type", 258, 97)
typenode.setKeyedPropertyValue("direction", "response_01", "Target")

featureselectionnode = stream.createAt("featureselection", "Feature Selection", 366, 97)
featureselectionnode.setPropertyValue("top_n", 15)
featureselectionnode.setPropertyValue("max_missing_values", 80.0)
featureselectionnode.setPropertyValue("selection_mode", "TopN")
featureselectionnode.setPropertyValue("important_label", "Check Me Out!")
featureselectionnode.setPropertyValue("criteria", "Likelihood")

stream.link(statisticsimportnode, typenode)
stream.link(typenode, featureselectionnode)
models = []
featureselectionnode.run(models)

# Assumes the stream automatically places model apply nodes in the stream
applynode = stream.findByType("applyfeatureselection", None)
```

```
tablenode = stream.createAt("table", "Table", applynode.getXPosition() + 96,
applynode.getYPosition())
stream.link(applynode, tablenode)
tablenode.run([])
```

此脚本创建了一个用以读入数据的源节点，使用"类型"节点将字段 response_01 的角色（方向）设置为目标，然后创建并执行"特征选择"节点。此脚本还连接流画布上的各个节点和位置以生成可读的布局。然后结果模型块与表节点相连接，"表"节点列出了属性 selection_mode 和 top_n 所确定的 15 个最重要的字段。有关更多信息，请参阅第 192 页的『featureselectionnode 属性』主题。

超节点脚本

通过使用 IBM SPSS Modeler 脚本语言，可以创建和保存所有终端超节点中的脚本。这些脚本只在终端超节点中可用，并且常在创建模板流或用于强制超节点内容以特定顺序执行时使用。使用超节点脚本，您也可以在流中运行多个脚本。

例如，假设需要指定一个复杂流的执行顺序，并且超节点包含若干个包括"设置全局值"节点的节点，而执行"设置全局值"节点又需要在派生用于散点图节点的新字段之前进行。这种情况下，可以创建一个首先执行"设置全局值"节点的超节点脚本。由"设置全局值"节点计算出的值，例如平均差或标准差，可在散点图节点的执行过程中使用。

在超节点脚本中也可以指定节点属性，操作方法与在其他脚本中的进行的操作一样。另外，为所有超节点或直接来自流脚本的超节点的封装节点更改和定义属性。有关更多信息，请参阅第 331 页的第 21 章, 『超节点属性』主题。此方法适用于源和过程超节点以及终端超节点。

注：因为只有终端超节点能够执行自身脚本，所以"超节点"对话框的"脚本"选项卡只在用于终端超节点时可用。

从主画布打开"超节点脚本"对话框：

从流画布选择终端超节点，然后从"超节点"菜单选择：

超节点脚本...

从放大超节点画布打开"超节点脚本"对话框：

右键单击超节点画布，然后从上下文菜单中选择：

超节点脚本...

超节点脚本示例

以下超节点脚本声明超节点中终端节点的执行顺序。此顺序确保首先执行"设置全局值"节点，以便随后执行其他节点时可以使用由此节点计算出的值。

```
execute 'Set Globals'
execute 'gains'
execute 'profit'
execute 'age v. $CC-pep'
execute 'Table'
```

锁定和解锁超节点

以下示例显示了如何锁定和解锁超节点：

```
stream = modeler.script.stream()
superNode=stream.findByID('id854RNTSD5MB')
# unlock one super node
print 'unlock the super node with password abcd'
if superNode.unlock('abcd'):
    print 'unlocked.'
else:
    print 'invalid password.'
# lock one super node
print 'lock the super node with password abcd'
superNode.lock('abcd')
```

流中的循环和条件执行

从 V16.0 开始，通过 SPSS Modeler，您可以选择各个对话框中的值在流中创建一些基本脚本，而无需使用脚本编制语言直接编写指令。可通过此方式创建的两种主要类型的脚本是简单循环以及在满足条件时执行节点的方式。

可以组合流中的循环规则和条件执行规则。例如，您可能具有来自世界各地制造商的汽车销售相关数据。您可以在流中设置一个用于处理数据的循环，从而按制造国家或地区标识详细信息，并将数据输出到各个显示了详细信息（例如，按型号排列的销售量，按制造商和引擎大小排列的排放级别等）的图形。如果您希望仅分析欧洲信息，那么还可以向循环添加条件，以阻止针对总部设在美国和亚洲的制造商创建图形。

注：由于循环和条件执行均以后台脚本为基础，因此它们仅适用于运行的整个流。

- **循环** 使用循环可自动化重复任务。例如，这可能意味着向流添加给定数目的节点，并且每次更改一个节点参数。另外，您可以将流或分支的运行控制为反复运行给定数目的次数，如以下示例所示：
  - 运行流给定数目的次数，并且每次都对源进行更改。
  - 运行流给定数目的次数，并且每次都对变量的值进行更改。
  - 运行流给定数目的次数，并且在每次执行时都输入一个额外的字段。
  - 构建模型给定数目的次数，并且每次都对模型设置进行更改。
- **条件执行** 您可以使用它根据预定义的条件来控制终端节点的运行方式，可能的示例如下：
  - 根据给定值是 true 还是 false，控制是否将运行节点。
  - 定义节点循环将以并行方式运行还是按顺序运行。

循环和条件执行都是在"流属性"对话框中的"执行"选项卡中设置的。任何在条件或循环要求中使用的节点都随附加到这些节点的附加符号一起显示在流画布上，此符号用于指示这些节点将参与循环和条件执行。

您可以通过下列三种方式中的其中一种来访问"执行"选项卡：

- 使用主对话框顶部的菜单：
  1. 从"工具"菜单中，选择：

     流属性 > **执行**
  2. 单击"执行"选项卡以处理当前流的脚本。
- 从流中：
  1. 右键单击节点，然后选择**循环/条件执行**。
  2. 选择相关子菜单选项。
- 从主对话框顶部的图形工具栏中，单击流属性图标。

如果这是您第一次设置循环或条件执行详细信息，请在"执行"选项卡上选择**循环/条件执行**执行方式，然后选择**条件**或**循环**子选项卡。

流中的循环

通过循环，您可以自动化流中的重复任务；可能的示例如下：

- 运行流给定数目的次数，并且每次都对源进行更改。
- 运行流给定数目的次数，并且每次都对变量的值进行更改。
- 运行流给定数目的次数，并且在每次执行时都输入一个额外的字段。
- 构建模型给定数目的次数，并且每次都对模型设置进行更改。

可以在流"执行"选项卡的**循环**子选项卡上设置要满足的条件。要显示该子选项卡，请选择**循环/条件执行**执行方式。

如果设置了**循环/条件执行**执行方式，那么在您运行流时，您定义的所有循环要求都将生效。（可选）您可以针对您的循环要求生成脚本代码，并通过单击"循环"子选项卡右下角的**粘贴...** 将此代码粘贴到脚本编辑器中；主要"执行"选项卡将显示此更改以显示**缺省（可选脚本）**执行方式，并将脚本显示在此选项卡的顶部。这意味着，您可以先使用多个循环对话框选项来定义循环结构，然后再生成可在脚本编辑器中进行进一步定制的脚本。请注意，当您单击**粘贴...** 时，您定义的所有条件执行要求也会显示在生成的脚本中。

要点：如果您在 IBM SPSS Collaboration and Deployment Services 作业中运行某个 SPSS Modeler 流，那么可以覆盖您在此流中设置的循环变量。这是因为，IBM SPSS Collaboration and Deployment Services 作业编辑器条目将覆盖 SPSS Modeler 条目。例如，如果您在流中设置了某个循环变量以便为每个循环创建不同的输出文件名称，那么这些文件将在 SPSS Modeler 中正确命名，但由 IBM SPSS Collaboration and Deployment Services Deployment Manager 的"结果"选项卡中输入的固定条目覆盖。

要设置循环，请完成下列步骤：

1. 创建迭代关键字以定义将在流中执行的主要循环结构。有关更多信息，请参阅创建迭代关键字。
2. 在需要时，定义一个或多个迭代变量。有关更多信息，请参阅创建迭代变量。
3. 您创建的迭代和所有变量都将显示在该子选项卡的主要部分中。缺省情况下，将按显示顺序执行迭代；要在列表中上下移动迭代，请单击迭代以将其选中，然后使用该子选项卡右侧的向上或向下箭头更改顺序。

创建用于流中的循环的迭代关键字

使用迭代关键字可以定义将在流中执行的主要循环结构。例如，如果要对汽车销售进行分析，那么可以创建流参数制造国家或地区，并将其用作迭代关键字；在运行流时，此关键字将设置为各个迭代过程中您的数据中的各个不同的国家或地区值。使用"定义迭代关键字"对话框可以设置关键字。

要打开此对话框，请选择"循环"子选项卡左下角的**迭代关键字...** 按钮，或者右键单击流中的任何节点，然后选择**循环/条件执行** > **定义迭代关键字（字段）**或**循环/条件执行** > **定义迭代关键字（值）**。如果是从流中打开此对话框，那么系统可能会自动为您填写一些字段，例如，节点的名称。

要设置迭代关键字，请填写下列字段：

迭代依据。您可以选择下列其中一个选项：

- **流参数 - 字段**。使用此选项可创建一个循环，用于将现有流参数的值依次设置为各个指定字段。
- **流参数 - 值**。使用此选项可创建一个循环，用于将现有流参数的值依次设置为各个指定值。
- **节点属性 - 字段**。使用此选项可创建一个循环，用于将节点属性的值依次设置为各个指定字段。
- **节点属性 - 值**。使用此选项可创建一个循环，用于将节点属性的值依次设置为各个指定值。

设置内容。选择将在每次执行循环时设置其值的项。您可以选择下列其中一个选项：

- **参数**。仅当您选择**流参数 - 字段**或**流参数 - 值**时才可用。从可用列表中选择所需参数。

- **节点**。仅当您选择**节点属性 - 字段**或**节点属性 - 值**时才可用。选择要对其设置循环的节点。单击浏览按钮以打开"选择节点"对话框并选择所需节点；如果列出的节点过多，那么可以通过选择下列其中一个类别对显示结果进行过滤，以仅显示特定类型的节点："源"、"进程"、"图形"、"建模"、"输出"、"导出"或"应用模型"节点。
- **属性**。仅当您选择**节点属性 - 字段**或**节点属性 - 值**时才可用。从可用列表中选择节点的属性。

要使用的字段。仅当您选择**流参数 - 字段**或**节点属性 - 字段**时才可用。选择节点中要用于提供迭代值的字段。您可以选择下列其中一个选项：

- **节点**。仅当您选择**流参数 - 字段**时才可用。选择要对其设置循环且包含详细信息的节点。单击浏览按钮以打开"选择节点"对话框并选择所需节点；如果列出的节点过多，那么可以通过选择下列其中一个类别对显示结果进行过滤，以仅显示特定类型的节点："源"、"进程"、"图形"、"建模"、"输出"、"导出"或"应用模型"节点。
- **字段列表**。单击右边列中的列表按钮可显示"选择字段"对话框，您可以在此对话框中选择节点中要用于提供迭代数据的字段。请参阅第 9 页的『选择用于迭代的字段』以获取更多信息。

要使用的值。仅当您选择**流参数 - 值**或**节点属性 - 值**时才可用。选择所选字段内要用作迭代值的值。您可以选择下列其中一个选项：

- **节点**。仅当您选择**流参数 - 值**时才可用。选择要对其设置循环且包含详细信息的节点。单击浏览按钮以打开"选择节点"对话框并选择所需节点；如果列出的节点过多，那么可以通过选择下列其中一个类别对显示结果进行过滤，以仅显示特定类型的节点："源"、"进程"、"图形"、"建模"、"输出"、"导出"或"应用模型"节点。
- **字段列表**。选择节点中用于提供迭代数据的字段。
- **值列表**。单击右边列中的列表按钮可显示"选择值"对话框，您可以在此对话框中选择节点中要用于提供迭代数据的字段。

创建用于流中的循环的迭代变量

您可以使用迭代变量在每次执行循环时更改流中的流参数值或选定节点的属性值。例如，如果流循环将对汽车销售数据进行分析并使用制造国家或地区作为迭代关键字，那么您可能会具有一个按型号显示销售额的图形输出，以及另一显示了废气排放信息的图形输出。在这些情况下，您可以创建迭代变量，这些变量将为生成的图形创建新标题，例如瑞典汽车排放和按型号排列的日本汽车销售额。使用"定义迭代变量"对话框可以设置任何您需要的变量。

要打开此对话框，请选择"循环"子选项卡左下角的**添加变量...** 按钮，或者右键单击流中的任何节点并选择**循环/条件执行** > **定义迭代变量**。

要设置迭代变量，请填写下列字段：

更改。选择要修改的属性的类型。可以从**流参数**或**节点属性**中进行选择。

- 如果选择**流参数**，请选择所需参数，然后通过循环的各个迭代，使用下列其中一个选项（如果在流中可用）定义应该将该参数设置为的值。
  - **全局变量**。选择应该将流参数设置为的全局变量。
  - **表输出单元**。要将流参数设置为表输出单元中的值，请从列表中选择表，然后输入要使用的**行**和**列**。
  - **手动输入**。如果要手动为此参数输入将在各个迭代中采用的值，请选择此选项。返回到"循环"子选项卡时，将创建一个可在其中输入所需文本的新列。
- 如果选择**节点属性**，请选择所需节点以及该节点的其中一个属性，然后选择要用于该属性的值。通过使用下列其中一个选项，可以设置新属性值：

– **单独**。属性值将使用迭代关键字值。请参阅第 7 页的『创建用于流中的循环的迭代关键字』以获取更多信息。
– **作为资源前缀**。使用迭代关键字值作为在**资源**字段中输入的内容的前缀。
– **作为资源后缀**。使用迭代关键字值作为在**资源**字段中输入的内容的后缀。

如果选择前缀或后缀选项，那么系统将提示您向**资源**字段添加附加文本。例如，如果迭代关键字值为制造国家或地区并且您选择**作为资源前缀**，那么可以在此字段中输入 - 按型号排列的销售额。

选择用于迭代的字段

创建迭代时，您可以使用"选择字段"对话框选择一个或多个字段。

排序依据 您可以通过选择下列其中一个选项对可供查看的字段进行排序：

- **自然** 字段的查看顺序即为这些字段向下传递到当前节点中的顺序。
- **名称** 使用字母顺序对可供查看的字段进行排序。
- **类型** 查看按其测量级别排序的字段。此选项在选择具有特定测量级别的字段时非常有用。

一次从列表中选择一个字段，或采用按住 Shift 并单击和按住 Ctrl 并单击的方法选择多个字段。此外，也可以使用列表下面的按钮根据测量级别选择多组字段，或选择或取消选择表中所有字段。

请注意，可供选择的字段将进行过滤，以仅显示适用于您使用的流参数或节点属性的字段。例如，如果您使用的是存储类型为字符串的流参数，那么将仅显示存储类型为字符串的字段。

流中的条件执行

通过条件执行，您可以根据与您所定义的条件相匹配的流内容来控制终端节点的运行方式；可能的示例如下：

- 根据给定值是 true 还是 false，控制是否将运行节点。
- 定义节点循环将以并行方式运行还是按顺序运行。

可以在流"执行"选项卡的**条件**子选项卡上设置要满足的条件。要显示该子选项卡，请选择**循环/条件执行**执行方式。

如果设置了**循环/条件执行**执行方式，那么在您运行流时，您定义的所有条件执行要求都将生效。（可选）您可以针对您的条件执行要求生成脚本代码，并通过单击"条件"子选项卡右下角的**粘贴...** 将此代码粘贴到脚本编辑器中；主要"执行"选项卡将显示此更改以显示**缺省（可选脚本）**执行方式，并将脚本显示在此选项卡的顶部。这意味着，您可以先使用多个循环对话框选项来定义条件，然后再生成可在脚本编辑器中进行进一步定制的脚本。请注意，当您单击**粘贴...** 时，您定义的所有循环要求也会显示在生成的脚本中。

要设置条件，请完成下列步骤：

1. 在"条件"子选项卡的右侧列中，单击"添加新条件"按钮  以打开"添加条件执行语句"对话框。在此对话框中，可以指定执行节点所必须满足的条件。
2. 在"添加条件执行语句"对话框中，指定以下内容：
   a. **节点**。选择要对其设置条件执行的节点。单击浏览按钮以打开"选择节点"对话框并选择所需节点；如果列出的节点过多，那么可以对显示结果进行过滤，以按下列其中一个类别显示节点："导出"、"图形"、"建模"或"输出"节点。
   b. **作为依据的条件**。指定执行节点所必须满足的条件。您可以从下列四个选项中选择其中一个：**流参数**、**全局变量**、**表输出单元**或**始终满足**。在对话框下半部分中输入的详细信息由您选择的条件控制。

- **流参数**从提供的列表中选择参数，然后选择该参数的**运算符**；例如，运算符可以是大于、等于、小于和介于之间等等。然后，输入**值**或最小值和最大值，具体取决于运算符。
- **全局变量**。从提供的列表中选择变量；例如，这可能包括平均值、总和、最小值、最大值或标准差。然后，选择**运算符**及所需值。
- **表输出单元**。从可用列表中选择表节点，然后选择表中的**行**和**列**。然后，选择**运算符**及所需值。
- **始终满足**。如果必须始终执行节点，请选择此选项。选择此选项后，将无需选择其他参数。

3. 重复步骤 1 和 2 所需次数，直到您设置了所有需要的条件。所选节点和执行该节点前所必须满足的条件将分别显示在该子选项卡主要部分中的**执行节点**和**如果满足此条件**列中。
4. 缺省情况下，将按显示顺序执行节点和条件；要在列表中上下移动节点和条件，请单击节点或条件以将其选中，然后使用该子选项卡右侧的向上或向下箭头更改顺序。

另外，您可以在"条件"子选项卡的底部设置下列选项：

- **按顺序对所有条件进行求值**。选择此选项可按条件在该子选项卡上的显示顺序对各个条件进行求值。对所有条件进行求值后，将立即执行那些条件求值为"true"的节点。
- **一次执行一个节点**。只有选中**按顺序对所有条件进行求值**时才可用。选中此选项表示，如果某个条件求值为"true"，那么将先执行与该条件关联的节点，然后再对下一个条件进行求值。
- **在首次命中之前进行求值**。选中此选项表示，将仅运行第一个根据您指定的条件返回"true"求值的节点。

执行和中断脚本

可以通过多种方法来执行脚本。例如，在流脚本或独立脚本对话框中，"运行此脚本"按钮将执行整个脚本：

图 *1.* ″运行此脚本″按钮

"运行选定的行"按钮用于执行您在脚本中选择的单一行或者相邻行所组成的块：

图 *2.* ″运行选定的行″按钮

可以使用以下方式执行脚本：

- 在流脚本或独立脚本对话框中，单击"运行此脚本"或"运行选定的行"按钮。
- 在**运行此脚本**设置为缺省执行方式的情况下运行流。
- 启动后以交互模式使用 `-execute` 标志。有关更多信息，请参阅第 55 页的『使用命令行自变量』主题。

注：如果在"超节点脚本"对话框中选择**运行此脚本**，那么将在执行超节点时执行超节点脚本。

中断脚本执行

"流脚本"对话框中的红色"停止"按钮将在脚本执行过程中被激活。使用此按钮可以放弃脚本和任何当前流的执行。

查找和替换

可在编辑脚本或表达式文本的位置（包括脚本编辑器和 CLEM 表达式构建器）或定义"报告"节点中的模板时使用"查找/替换"对话框。在上述任何区域中编辑文本时，按 Ctrl+F 键都可以访问此对话框，从而确保光标的焦点位于文本区域中。例如，使用"填充器"节点时，可以从"设置"选项卡上的任何文本区域或者表达式构建器中的文本字段中访问此对话框。

1. 在光标位于文本区域中时，按 Ctrl+F 键可以访问"查找/替换"对话框。
2. 输入要搜索的文本，或从最近搜索项下拉列表中选择。
3. 输入替换文本（如果有的话）。
4. 单击**查找下一个**开始搜索。
5. 单击**替换**替换当前选定的内容，或单击**全部替换**更新所有项或选定的实例。
6. 每次操作完成后，此对话框将关闭。从任一文本区域中按 F3 键，可重复上一次查找操作，或按 Ctrl+F，可再次访问该对话框。

搜索选项

匹配大小写。指定查找操作是否区分大小写；例如 *myvar* 是否与 *myVar* 匹配。无论怎样设置，替换文本始终完全按照输入插入。

仅限整个单词。指定查找操作是否匹配单词中嵌入的文本。如果选中，*spider* 的搜索结果将不会包括 *spiderman* 或 *spider-man*。

正则表达式。指定是否使用正则表达式语法（请参阅下一节）。如果选中，**仅限于整个单词**选项将禁用并且会忽略其值。

仅限所选文本。控制使用**全部替换**选项时的搜索范围。

正则表达式语法

使用正则表达式，您可以搜索特殊字符（如选项卡或换行字符)、字符的类或范围（如 *a* 到 *d*）、任何数字或非数字以及边界（如行首或行尾）。支持的表达式类型如下。

表 *1.* 字符匹配.

字符	匹配
x	字符 x
\\	反斜杠字符
\0n	含八进制值的字符 0n (0 <= n <= 7)
\0nn	含八进制值的字符 0nn (0 <= n <= 7)
\0mnn	含八进制值的字符 0mnn (0 <= m <= 3, 0 <= n <= 7)
\xhh	含十六进制值的字符 0xhh
\uhhhh	含十六进制值的字符 0xhhhh
\t	制表符 ('\u0009')
\n	换行符 ('\u000A')
\r	回车符 ('\u000D')
\f	换页符 ('\u000C')
\a	警报（蜂鸣）符 ('\u0007')
\e	转义符 ('\u001B')

表 *1.* 字符匹配 *(续).*

字符	匹配
\cx	x 对应的控制字符

表 *2.* 匹配字符类.

字符类	匹配
[abc]	a、b、或 c（简单类）
[^abc]	除 a、b、或 c 之外的所有字符（相减）
[a-zA-Z]	a 到 z 或 A 到 Z，包含（范围）
[a-d[m-p]]	a 到 d 或者 m 到 p（并集）。也可指定为 [a-dm-p]
[a-z&&[def]]	a 到 z 和 d、e、或 f（交集）
[a-z&&[^bc]]	a 到 z，b 和 c 除外（相减）。也可指定为 [ad-z]
[a-z&&[^m-p]]	a 到 z，而非 m 到 p（相减）。也可指定为 [a-lq-z]

表 *3.* 预定义字符类.

预定义字符类	匹配
.	任意字符（可能或不可能与行终止符匹配）
\d	任意数字：[0-9]
\D	非数字：[^0-9]
\s	空格字符：[\t\n\x0B\f\r]
\S	非空格字符：[^\s]
\w	单词字符：[a-zA-Z_0-9]
\W	非单词字符：[^\w]

表 *4.* 边界匹配.

边界匹配符	匹配
^	行首
$	行尾
\b	单词边界
\B	非单词边界
\A	输入的开头
\Z	除最后终止符外（如果有），输入的结尾
\z	输入的结尾

第 2 章 脚本语言

脚本编写语言概述

通过 IBM SPSS Modeler 的脚本编制工具，您可以创建一些脚本，这些脚本可以在 SPSS Modeler 用户界面上运行、处理输出对象并运行命令语法。您可以在 SPSS Modeler 中直接运行这些脚本。

IBM SPSS Modeler 中的脚本以脚本语言 Python 编写。IBM SPSS Modeler 所使用的基于 Java 的 Python 实现称为 Jython。脚本语言包含下列功能部件：

- 用于引用节点、流、工程、输出和其他 IBM SPSS Modeler 对象的格式。
- 可用于处理这些对象的一组脚本编制语句或命令。
- 用于设置变量、参数和其他对象的值的脚本编制表达式语言。
- 注释、连接符和文字文本块的支持。

以下各节描述了 Python 脚本语言、Python 的 Jython 实现以及在 IBM SPSS Modeler 内进行脚本编制的入门基本语法。具体属性和命令的有关信息则在随后的章节中提供。

Python 和 Jython

Jython 是 Python 脚本语言的实现，它以 Java 语言编写并与 Java 平台进行集成。Python 是一种面向对象的功能强大的脚本语言。Jython 具有成熟脚本语言的生产力特征，而且与 Python 不同的是，Jython 可以在任何支持 Java 虚拟机 (JVM) 的环境中运行。这意味着您在编写程序时可以使用 JVM 上的 Java 库。通过 Jython，您可以利用此差异并使用 Python 语言的语法和大部分功能

作为一种脚本语言，Python（及其 Jython 实现）易于学习并能够高效地进行编码，而且具备创建运行程序所需要的最小结构。可以在交互方式下输入代码，即一次输入一行。Python 是一种解释性脚本语言；它没有 Java 中的预编译步骤。Python 程序仅仅是文本文件，系统将在输入这些文件时对其进行解释（在解析语法错误后）。简单表达式（例如已定义的值）以及更加复杂的操作（例如函数定义）将立即执行并可供使用。可以快速测试任何对代码进行的更改。但是，脚本解释确实存在一些缺点。例如，由于使用未定义的变量不是编译器错误，因此只有在执行使用了该变量的语句的情况下，才会检测到此错误。在这种情况下，可以编辑并运行程序以调试错误。

Python 将所有内容（包括所有数据和代码）视为对象。因此，您可以使用多行代码来处理这些对象。某些选择类型（例如数字和字符串）将被更加方便地视为值而不是对象；Python 支持此行为。有一个受支持的 Null 值。此 Null 值具有保留名称 `None`。

有关 Python 和 Jython 脚本编制的更深入介绍以及一些示例脚本，请参阅 http://www.ibm.com/developerworks/java/tutorials/j-jython1/j-jython1.html 和 http://www.ibm.com/developerworks/java/tutorials/j-jython2/j-jython2.html 。

Python 脚本编制

本 Python 脚本语言指南介绍了在 IBM SPSS Modeler 中编制脚本时最可能使用的组件，其中包括概念和编程基础。这将为您提供足够的知识来开发自己的 Python 脚本，以便在 IBM SPSS Modeler 中使用。

运算

赋值通过使用等号 (=) 来完成。例如，要将值"3"赋值给名为"x"的变量，您可以使用以下语句：

```
x = 3
```

等号还可用于将字符串类型数据赋值给变量。例如，要将值"a string value"赋值给变量"y"，您可以使用以下语句：

```
y = "a string value"
```

下表列出了一些常用的比较运算和数值运算及其描述。

表 *5.* 常用的比较运算和数值运算

运算	描述
x < y	x 是否小于 y?
x > y	x 是否大于 y?
x <= y	x 是否小于或等于 y?
x >= y	x 是否大于或等于 y?
x == y	x 是否等于 y?
x != y	x 是否不等于 y?
x <> y	x 是否不等于 y?
x + y	将 y 与 x 相加
x - y	从 x 中减去 y
x * y	将 x 乘以 y
x / y	将 x 除以 y
x ** y	求 x 的 y 次幂

列表

列表是元素序列。列表可以包含任意数目的元素，而列表的元素可以是任何类型的对象。也可以将列表视为阵列。随着添加、除去或替换元素，列表中元素的数目可能会增加或减少。

示例

```
[]
```
任何空列表。

```
[1]
```
包含单个元素（整数）的列表。

```
["Mike", 10, "Don", 20]
```
包含 4 个元素（两个字符串元素和两个整数元素）的列表。

```
[[],[7],[8,9]]
```
列表的列表。每个子列表都是一个空列表或整数元素列表。

```
x = 7; y = 2; z = 3;
[1, x, y, x + y]
```
整数列表。此示例说明了变量和表达式的使用。

您可以向变量分配列表，例如：

```
mylist1 = ["one", "two", "three"]
```

然后，可以访问列表的特定元素，例如：

```
mylist[0]
```

这将生成以下输出：

```
one
```

方括号 ([]) 中的数字称为索引，它指向列表中的某个特定元素。将从 0 开始对列表中的元素编制索引。

您也可以选择列表中的一系列元素；这称为切割。例如，x[1:3] 将选择 x 的第 2 个和第 3 个元素。结尾索引是所选内容后面的一个索引。

字符串

字符串是一个被视为值的不可变字符序列。字符串支持所有生成新字符串的不可变序列函数和运算符。例如，"abcdef"[1:4] 将生成输出 "bcd"。

在 Python 中，字符由长度为 1 的字符串表示。

字符串字面值通过使用单重引用或三重引用来定义。使用单引号定义的字符串不能跨行，而使用三重引号定义的字符串可以跨行。可以将字符串括在单引号 (') 或双引号 (") 中。引用字符可以包含其他未转义的引用字符或已转义（即，前面带有反斜杠 (\) 字符）的引用字符。

示例

```
"This is a string"
'This is also a string'
"It's a string"
'This book is called "Python Scripting and Automation Guide".'
"This is an escape quote (\") in a quoted string"
```

Python 解析器将自动合并多个以空格分隔的字符串。这样您可以更轻松地输入长字符串，并且更容易混合单个字符串中的引号类型，例如：

```
"This string uses ' and " 'that string uses ".'
```

这将生成以下输出：

```
This string uses ' and that string uses ".
```

字符串支持一些有用的方法。下表列出了其中一些方法。

表 *6.* 字符串方法

方法	用法
s.capitalize()	对 s 执行首字母大写
s.count(ss {,start {,end}})	计算 ss 在 s[start:end] 中的出现次数
s.startswith(str {, start {, end}}) s.endswith(str {, start {, end}})	测试以查看 s 是否以 str 开头 测试以查看 s 是否以 str 结尾
s.expandtabs({size})	将制表符替换为空格，缺省 size 为 8
s.find(str {, start {, end}}) s.rfind(str {, start {, end}})	在 s 中查找 str 的第一个索引；如果找不到，那么结果为 -1。rfind 从右到左进行搜索。
s.index(str {, start {, end}}) s.rindex(str {, start {, end}})	在 s 中查找 str 的第一个索引；如果找不到，那么将引起 ValueError。rindex 从右到左进行搜索。
s.isalnum	测试以查看字符串是否为字母数字字符串
s.isalpha	测试以查看字符串是否为字母字符串
s.isnum	测试以查看字符串是否为数字字符串
s.isupper	测试以查看字符串是否为全部大写
s.islower	测试以查看字符串是否为全部小写

表 6. 字符串方法（续）

方法	用法
s.isspace	测试以查看字符串是否全是空格
s.istitle	测试以查看字符串是否为首字母大写的字母数字字符串序列
s.lower() s.upper() s.swapcase() s.title()	转换为全部小写 转换为全部大写 转换为大小写颠倒 转换为全标题形式
s.join(seq)	将 seq 中的字符串连接起来，以 s 作为分隔符
s.splitlines({keep})	将 s 分割为多行，如果 keep 为 true，那么将使用换行
s.split({sep {, max}})	使用 sep（缺省 sep 为空格）将 s 分割为"字"，最多分割 max 次数
s.ljust(width) s.rjust(width) s.center(width) s.zfill(width)	在宽度为 width 的字段中，将字符串左对齐 在宽度为 width 的字段中，将字符串右对齐 在宽度为 width 的字段中，将字符串居中对齐 用 0 进行填充。
s.lstrip() s.rstrip() s.strip()	除去前导空格 除去尾部空格 除去前导和尾部空格
s.translate(str {,delc})	除去 delc 中的所有字符后，使用表转换 s。str 应该是长度为 == 256 的字符串。
s.replace(old, new {, max})	使用字符串 new 全部替换或按照 max 出现次数替换字符串 old

备注

备注是由井号（或散列符号）(#) 引入的注释。同一行上位于井号后面的所有文本都将被视为备注的组成部分，并且将被忽略。备注可以开始于任何列。以下示例说明了备注的使用：

```
#The HelloWorld application is one of the most simple
print 'Hello World' # print the Hello World line
```

语句语法

Python 的语句语法非常简单。通常，每个源代码行都是单一语句。除 expression 和 assignment 语句外，每个语句都由一个关键字名称（例如 if 或 for）引入。可以在代码中任何语句之间的任意位置插入空白行或备注行。如果某一行中有多个语句，那么必须使用分号 (;) 来分隔每个语句。

超长语句可以分为多行。在这种情况下，要分到下一行的语句必须以反斜杠 (\) 结尾，例如：

```
x = "A looooooooooooooooooong string" + \
     "another looooooooooooooooooong string"
```

如果某个结构括在圆括号 (())、方括号 ([]) 或花括号 ({}) 内，那么语句可以在任何逗号后面分为新行，而不必插入反斜杠，例如：

```
x = (1, 2, 3, "hello",
      "goodbye", 4, 5, 6)
```

标识

标识用于对变量、函数、类和关键字进行命名。标识的长度任意，但必须以大写或小写的字母字符或下划线字符 (_) 开头。以下划线开头的名称将通常保留作为内部名称或专用名称。在第一个字符后面，标识可以包含任意数目的字母字符、0 到 9 的数字以及下划线字符，并且这些字符和数字可以任意组合。

Jython 中的一些保留字不可用于对变量、函数或类进行命名。这些保留字分为下列类别：

- **语句引导词：**assert、break、class、continue、def、del、elif、else、except、exec、finally、for、from、global、if、import、pass、print、raise、return、try 和 while
- **参数引导词：**as、import 和 in
- **运算符：**and、in、is、lambda、not 和 or

关键字使用不当通常会生成 SyntaxError。

代码块

代码块是在期望单个语句的位置使用的语句组。代码块可以跟随下列任何语句：if、elif、else、for、while、try、except、def 和 class。这些语句将引入带有冒号字符 (:) 的代码块，例如：

```
if x == 1:
    y = 2
    z = 3
elif:
    y = 4
    z = 5
```

使用缩进对代码块进行定界，而不是像 Java 一样使用花括号。代码块中的所有行都必须缩进到同一位置。这是因为对缩进的更改指示代码块结束。通常，每一级缩进四个空格。建议使用空格而不是制表符来缩进行。不得混用空格和制表符。模块的最外层块中的行必须从第一列开始，否则将发生 SyntaxError。

组成代码块的语句（以及冒号后面的语句）也可以包括在一行中，并以分号分隔，例如：

```
if x == 1: y = 2; z = 3;
```

将参数传递给脚本

将参数传递给脚本非常有用，因为这表示可以重复使用脚本而无需进行修改。在命令行中传递的参数将作为列表 sys.argv 中的值进行传递。使用命令 len(sys.argv) 可以获取所传递的值的数目。例如：

```
import sys
print "test1"
print sys.argv[0]
print sys.argv[1]
print len(sys.argv)
```

在此示例中，import 命令用于导入整个 sys 类，以便可以使用这个类中存在的方法，例如 argv。

可以使用以下行调用此示例中的脚本：

```
/u/mjloos/test1 mike don
```

结果为以下输出：

```
/u/mjloos/test1 mike don
test1
mike
don
3
```

示例

print 关键字将打印紧跟其后的参数。如果语句后跟逗号，那么不会在输出中新增一行。例如：

```
print "This demonstrates the use of a",
print " comma at the end of a print statement."
```

这将生成以下输出：

```
This demonstrates the use of a comma at the end of a print statement.
```

for 语句用于迭代代码块。例如：

```
mylist1 = ["one", "two", "three"]
for lv in mylist1:
    print lv
    continue
```

在此示例中，将为列表 mylist1 分配 3 个字符串。然后，将打印该列表的元素，每个元素占用一行。这将生成以下输出：

```
one
two
three
```

在此示例中，迭代器 lv 将依次采用列表 mylist1 中每个元素的值，因为 for 循环用于实现每个元素的代码块。迭代器可以是任意长度的任何有效标识。

if 语句是条件语句。该语句将对条件进行求值，并根据求值结果返回 true 或 false。例如：

```
mylist1 = ["one", "two", "three"]
for lv in mylist1:
    if lv == "two"
        print "The value of lv is ", lv
    else
        print "The value of lv is not two, but ", lv
    continue
```

在此示例中，对迭代器 lv 进行了求值。如果 lv 的值为 two，那么将返回一个字符串，该字符串不同于 lv 的值不是 two 时返回的字符串。这将生成以下输出：

```
The value of lv is not two, but one
The value of lv is two
The value of lv is not two, but three
```

数学方法

您可以从 math 模块访问有用的数学方法。下表列出了其中一些方法。除非另有说明，否则所有值将作为浮点数返回。

表 7. 数学方法

方法	用法
math.ceil(x)	将 x 的上限作为浮点数返回，即大于或等于 x 的最小整数
math.copysign(x, y)	返回带有 y 的符号的 x。copysign(1, -0.0) 将返回 -1
math.fabs(x)	返回 x 的绝对值
math.factorial(x)	返回 x 阶乘。如果 x 是负数或非整数，那么将发生 ValueError。
math.floor(x)	将 x 的下限作为浮点数返回，即小于或等于 x 的最大整数

表 7. 数学方法 (续)

方法	用法
math.frexp(x)	将 x 的尾数 (m) 和指数 (e) 作为 (m, e) 对返回。m 是浮点数，e 是整数，这样刚好满足 x == m * 2**e。如果 x 为 0，那么此方法将返回 (0.0, 0)，否则将返回 0.5 <= abs(m) < 1。
math.fsum(iterable)	返回 iterable 中值的精确浮点总和
math.isinf(x)	检查浮点数 x 是正不定式还是负不定式
math.isnan(x)	检查浮点数 x 是否为 NaN（非数字）
math.ldexp(x, i)	返回 x * (2**i)。此方法本质上是函数 frexp 的反函数。
math.modf(x)	返回 x 的小数和整数部分。这两个结果都带有 x 的符号，并且都是浮点数。
math.trunc(x)	返回已截断为 Integral 的 Real 值 x。
math.exp(x)	返回 e**x
math.log(x[, base])	返回以给定值 base 为底的 x 的对数。如果未指定 base，那么将返回 x 的自然对数。
math.log1p(x)	返回 1+x (base e) 的自然对数
math.log10(x)	返回以 10 为底的 x 的对数
math.pow(x, y)	返回 x 的 y 次幂。pow(1.0, x) 和 pow(x, 0.0) 将始终返回 1，即使 x 为 0 或非数字时也是如此。
math.sqrt(x)	返回 x 的平方根

除数学函数外，还提供了一些有用的三角函数法。下表列出了这些方法。

表 8. 三角函数法

方法	用法
math.acos(x)	返回以弧度表示的 x 的反余弦
math.asin(x)	返回以弧度表示的 x 的反正弦
math.atan(x)	返回以弧度表示的 x 的反正切
math.atan2(y, x)	返回以弧度表示的 atan(y / x)。
math.cos(x)	返回以弧度表示的 x 的余弦。
math.hypot(x, y)	返回欧几里得范数 sqrt(x*x + y*y)。这是从原点到点 (x, y) 的向量的长度。
math.sin(x)	返回以弧度表示的 x 的正弦
math.tan(x)	返回以弧度表示的 x 的正切
math.degrees(x)	将角度 x 从弧度转换为度
math.radians(x)	将角度 x 从度转换为弧度
math.acosh(x)	返回 x 的反双曲余弦值
math.asinh(x)	返回 x 的反双曲正弦值
math.atanh(x)	返回 x 的反双曲正切值
math.cosh(x)	返回 x 的双曲正弦值
math.sinh(x)	返回 x 的双曲正弦值
math.tanh(x)	返回 x 的双曲正切值

还有两个数学常量。`math.pi` 的值为数学常量 pi。`math.e` 的值为数学常量 e。

使用非 ASCII 字符

要使用非 ASCII 字符，Python 需要明确地将字符串编码和解码为 Unicode。在 IBM SPSS Modeler 中，假定 Python 脚本采用 UTF-8 进行编码，这是支持非 ASCII 字符的标准 Unicode 编码。以下脚本将执行编译，这是因为 SPSS Modeler 已将 Python 编译器设置为 UTF-8。

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", "テストノード", 96, 64)
```

但是，生成的节点将具有不正确的标签。

图 3. 错误显示的包含非 *ASCII* 字符的节点标签

标签不正确，因为 Python 已将字符串字面值自身转换为 ASCII 字符串。

Python 通过在字符串字面值前添加 `u` 字符前缀来支持指定 Unicode 字符串字面值：

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", u"テストノード", 96, 64)
```

这将创建 Unicode 字符串，并且将正确显示标签。

图 4. 正确显示的包含非 *ASCII* 字符的节点标签

使用 Python 和 Unicode 是一个非常大的主题，它超出了本文档的范围。提供了许多对此主题进行了更详细介绍的书籍和在线资源。

面向对象的程序设计

面向对象的程序设计基于在程序中创建目标问题模型的概念。面向对象的程序设计减少了编程错误并促进了代码的复用。Python 是一种面向对象的语言。以 Python 定义的对象具有下列特征：

- **身份。**每个对象都必须截然不同，并且必须可以对此特征进行测试。`is` 和 `is not` 测试可用于此目的。
- **状态。**每个对象都必须能够存储状态。属性（例如字段和实例变量）可用于此目的。
- **行为。**每个对象都必须能够处理其状态。方法可用于此目的。

Python 提供了支持面向对象的程序设计的下列特征：

- **基于类的对象创建。**类是用于创建对象的模板。对象是具有关联行为的数据结构。
- **多态性继承。**Python 支持单继承和多重继承。所有 Python 实例方法都具有多态性，并且可以由子类覆盖。
- **具有隐藏数据的封装。**Python 允许隐藏属性。隐藏后，将只能通过类的方法从类外部访问这些属性。类实现了用于修改数据的方法。

定义类

在 Python 类中，可以定义变量和方法。与 Java 不同，您可以在 Python 中对每个源文件（或模块）定义任意数目的公共类。因此，可以认为 Python 中的模块类似于 Java 中的软件包。

在 Python 中，类是使用 `class` 语句定义的。`class` 语句的格式如下：

```
class name (superclasses): statement
```

或

```
class name (superclasses):
    assignment
    .
    .
    function
    .
    .
```

定义类时，您可以选择提供零个或零个以上的赋值语句。这些赋值语句将创建该类的所有实例共享的类属性。您还可以提供零个或零个以上的函数定义。这些函数定义将创建方法。超类列表是可选的。

在同一作用域中（即模块、函数或类中），类名应该唯一。您可以将多个变量定义为引用同一类。

创建类实例

类用于保存类（或共享）属性，或者用于创建类实例。要创建某个类的实例，请将该类作为函数进行调用。例如，请考虑以下类：

```
class MyClass:
    pass
```

这里使用了 `pass` 语句，因为需要一个语句来完成类，但不需要以编程方式执行任何操作。

以下语句将创建类 `MyClass` 的实例：

```
x = MyClass()
```

向类实例添加属性

与 Java 不同，客户机可以在 Python 中向类实例添加属性。只有一个实例会发生更改。例如，要向实例 `x` 添加属性，请在该实例上设置新值：

```
x.attr1 = 1
x.attr2 = 2
    .
    .
x.attrN = n
```

定义类属性和方法

任何绑定在类中的变量都是类属性。任何在类中定义的函数都是方法。方法接收类的实例（通常称为 `self`）作为第一个自变量。例如，要定义一些类属性和方法，您可以输入以下代码：

```
class MyClass
    attr1 = 10        #class attributes
    attr2 = "hello"

    def method1(self):
        print MyClass.attr1   #reference the class attribute

    def method2(self):
        print MyClass.attr2   #reference the class attribute

    def method3(self, text):
        self.text = text        #instance attribute
        print text, self.text   #print my argument and my attribute

    method4 = method3   #make an alias for method3
```

在类中，您应该使用类名限定所有对类属性的引用，例如 MyClass.attr1。应该使用 self 变量限定所有对实例属性的引用；例如 self.text。在类外部，您应该使用类名（例如 MyClass.attr1）或使用类的实例（例如 x.attr1，其中 x 是类的实例）限定所有对类属性的引用。在类外部，应该使用类的实例限定所有对实例变量的引用；例如 x.text。

隐藏变量

可以通过创建专用变量来隐藏数据。专用变量只能由类自身进行访问。如果您声明格式为 __xxx 或 __xxx_yyy（即，带有两个前置下划线）的名称，那么 Python 解析器将自动向声明的名称添加类名以创建隐藏变量，例如：

```
class MyClass:
    __attr = 10   #private class attribute

    def method1(self):
        pass

    def method2(self, p1, p2):
        pass

    def __privateMethod(self, text):
        self.__text = text    #private attribute
```

与 Java 不同，在 Python 中必须使用 self 限定所有对实例变量的引用；未隐含地使用 this。

继承

从类进行继承的功能是面向对象的程序设计的基础。Python 同时支持单继承和多重继承。单继承表示只能存在一个超类。多重继承表示可以存在多个超类。

继承通过对其他类划分子类来实现。许多 Python 类都可以是超类。在 Python 的 Jython 实现中，只能从一个 Java 类进行直接或间接继承。无需提供超类。

超类中的任何属性或方法也包含在任何子类中，并且只要属性或方法未隐藏，类自身或任何客户机就可以使用这些属性或方法。可以在任何位置使用子类的任何实例，并且可以使用超类的实例；这是多态性示例。这些特征支持复用和轻松扩展。

示例

```
class Class1: pass    #no inheritance

class Class2: pass
```

```
class Class3(Class1): pass     #single inheritance

class Class4(Class3, Class2): pass     #multiple inheritance
```

第 3 章 在 IBM SPSS Modeler 中进行脚本编制

脚本类型

在 IBM SPSS Modeler 中，有 3 种类型的脚本：

- 流脚本，用于控制单个流的执行并且它存储在流中。
- 超节点脚本，用于控制超节点的行为。
- 独立或会话脚本，可用于跨多个不同的流协调执行。

提供了在 IBM SPSS Modeler 中的脚本内使用的多种方法，您可以使用这些方法来访问各种 SPSS Modeler 功能。另外，这些方法还在第 33 页的第 4 章, 『脚本编制 API』 中用于创建更高级的函数。

流、超节点流和图

在大多数情况下，术语流表示同一内容，而与流是从文件装入还是在超节点中使用无关。通常，它表示连接在一起的节点的集合，并且可以执行。但是，在脚本编制中，并非所有操作在任何位置都受支持，这意味着脚本作者应该了解其所使用的流变体。

流

流是主要的 IBM SPSS Modeler 文档类型。可以保存、查找、编辑和执行流。流还可以具有参数、全局值、脚本以及其他与之关联的信息。

超节点流

超节点流是一种在超节点中使用的流类型。与普通流一样，它包含链接在一起的节点。超节点流与普通流之间存在一些差异：

- 参数及所有脚本与超节点流所在的超节点进行关联，而不是与超节点流自身关联。
- 超节点流具有附加的输入和输出连接器节点（具体取决于超节点的类型）。这些连接器节点用于将信息流入和流出超节点流，并且在创建超节点时自动创建。

图

术语图涵盖普通流和超节点流支持的功能，例如添加和除去节点，以及修改节点之间的连接。

执行流

以下示例将运行流中的所有可执行节点，并且它是最简单的流脚本类型：

```
modeler.script.stream().runAll(None)
```

以下示例也将运行流中的所有可执行节点：

```
stream = modeler.script.stream()
stream.runAll(None)
```

在此示例中，流存储在名为 stream 的变量中。将流存储在变量中非常有用，因为脚本通常用于修改流或流中的节点。创建用于存储流的变量将生成一个更加简洁的脚本。

脚本编制上下文

modeler.script 模块提供了在其中执行脚本的上下文。此模块将在运行时自动导入 SPSS Modeler 脚本中。此模块定义了以下 4 个函数，这些函数提供可以访问其执行环境的脚本：

- session() 函数，用于返回脚本的会话。此会话定义语言环境等信息以及将用于运行任何流的 SPSS Modeler 后端（本地进程或联网 SPSS Modeler Server）。
- stream() 函数，此函数可以与流脚本及超节点脚本配合使用。此函数将返回拥有正在运行的流脚本或超节点脚本的流。
- diagram() 函数，此函数可以与超节点脚本配合使用。此函数将返回超节点内的图。对于其他脚本类型，此函数返回的内容与 stream() 函数相同。
- supernode() 函数，此函数可以与超节点脚本配合使用。此函数将返回拥有正在运行的脚本的超节点。

下表概述了这四个函数及其输出。

表 *9. modeler.script* 函数摘要

脚本类型	session()	stream()	diagram()	supernode()
独立	返回会话	在调用该脚本时返回当前受管流（例如，通过批处理方式 -stream 选项传递的流）或 None。	与 stream() 相同	不适用
流	返回会话	返回流	与 stream() 相同	不适用
超节点	返回会话	返回流	返回超节点流	返回超节点

modeler.script 模块还定义了一种使用退出码终止脚本的方式。exit(*exit-code*) 函数将停止执行脚本并返回所提供的整数退出码。

为流定义的其中一种方法是 runAll(List)。此方法将运行所有可执行节点。执行节点时生成的任何模型或输出都将添加到所提供的列表中。

流执行通常生成模型和图形等输出以及其他输出。要捕获此输出，脚本可以提供一个初始化为列表的变量，例如：

```
stream = modeler.script.stream()
results = []
stream.runAll(results)
```

执行完成后，可以从 results 列表访问执行所生成的任何对象。

引用现有节点

流通常是使用一些参数预先构建的，在执行流之前必须先修改这些参数。修改这些参数涉及下列任务：

1. 在相关流中找到节点。
2. 更改节点和/或流设置。

查找节点

流提供了多种查找现有节点的方式。下表概述了这些方法。

表 *10.* 查找现有节点的方法

方法	返回类型	描述
s.findAll(type, label)	集合	返回具有指定类型和标签的所有节点的列表。类型或标签可以为 None，在这种情况下将使用其他参数。
s.findAll(filter, recursive)	集合	返回指定过滤器所接受的所有节点的集合。如果递归标志为 True，那么还将搜索指定流内的任何超节点。
s.findByID(id)	节点(N)	返回具有所提供标识的节点或 None（如果不存在此类节点）。搜索范围限制为当前流。
s.findByType(type, label)	节点(N)	返回具有所提供类型和 / 或标签的节点。类型或名称可以为 None，在这种情况下将使用其他参数。如果有多个节点匹配，那么将选择并返回任意一个节点。如果没有匹配的节点，那么返回值为 None。
s.findDownstream(fromNodes)	集合	从所提供的节点列表中进行搜索，并返回所提供节点下游的节点集合。返回的列表包括最初提供的节点。
s.findUpstream(fromNodes)	集合	从所提供的节点列表中进行搜索，并返回所提供节点上游的节点集合。返回的列表包括最初提供的节点。

例如，如果流包含脚本需要访问的单个"过滤"节点，那么可以使用以下脚本来找到该"过滤"节点：

```
stream = modeler.script.stream()
node = stream.findByType("filter", None)
...
```

另外，如果已经知道节点的标识（如节点对话框的"注释"选项卡中所示），那么可以使用此标识来查找节点，例如：

```
stream = modeler.script.stream()
node = stream.findByID("id32FJT71G2")  # the filter node ID
...
```

设置属性

节点、流、模型和输出都具有可以访问并在大多数情况下可以设置的属性。这些属性通常用于修改对象的行为或外观。下表概述了可用于访问和设置对象属性的方法。

表 *11.* 用于访问和设置对象属性的方法

方法	返回类型	描述
p.getPropertyValue(propertyName)	对象	返回指定属性的值或 None（如果不存在此属性）。
p.setPropertyValue(propertyName, value)	不适用	设置指定属性的值。

表 *11.* 用于访问和设置对象属性的方法 *(续)*

方法	返回类型	描述
p.setPropertyValues(properties)	不适用	设置指定属性的值。属性图中的每个条目都包含一个表示属性名的键，以及应指定给该属性的值。
p.getKeyedPropertyValue(propertyName, keyName)	对象	返回指定属性的值及关联的键或 None（如果不存在此属性或键）。
p.setKeyedPropertyValue(propertyName, keyName, value)	不适用	设置指定属性的值和键。

例如，如果要设置位于流的开始位置的"变量文件"节点的值，那么可以使用以下脚本：

```
stream = modeler.script.stream()
node = stream.findByType("variablefile", None)
node.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
...
```

或者，您可能希望根据"过滤"节点来过滤字段。在这种情况下，还将根据字段名称键入值，例如：

```
stream = modeler.script.stream()
# Locate the filter node ...
node = stream.findByType("filter", None)
# ... and filter out the "Na" field
node.setKeyedPropertyValue("include", "Na", False)
```

创建节点以及修改流

在某些情况下，您可能希望向现有流添加新节点。向现有流添加节点通常涉及下列任务：

1. 创建节点。
2. 将节点链接到现有流。

创建节点

流提供了多种创建节点的方式。下表概述了这些方法。

表 *12.* 创建节点的方法

方法	返回类型	描述
s.create(nodeType, name)	节点(N)	创建具有指定类型的节点并将其添加到指定的流中。
s.createAt(nodeType, name, x, y)	节点(N)	创建具有指定类型的节点并将其添加到指定流中的指定位置。如果 x < 0 或 y < 0，那么未设置位置。
s.createModelApplier(modelOutput, name)	节点(N)	创建派生自所提供的模型输出对象的模型应用器节点。

例如，要在流中创建新的"类型"节点，您可以使用以下脚本：

```
stream = modeler.script.stream()
# Create a new type node
node = stream.create("type", "My Type")
```

链接和取消链接节点

在流中创建新节点时，这个新节点必须先连接到节点序列，然后才可使用。流提供了多种链接节点和取消链接节点的方法。下表概述了这些方法。

表 *13.* 用于链接和取消链接节点的方法

方法	返回类型	描述
s.link(source, target)	不适用	在源节点与目标节点之间创建新链接。
s.link(source, targets)	不适用	在源节点与所提供列表中的每个目标节点之间创建新链接。
s.linkBetween(inserted, source, target)	不适用	连接两个其他节点实例（源节点和目标节点）之间的节点，并将已插入节点的位置设置为位于这两个节点实例之间。将首先除去源节点与目标节点之间的任何直接链接。
s.linkPath(path)	不适用	在节点实例之间创建新路径。第一个节点将链接到第二个节点，而第二个节点将链接到第三个节点，依此类推。
s.unlink(source, target)	不适用	除去源节点与目标节点之间的任何直接链接。
s.unlink(source, targets)	不适用	除去源节点与目标列表中每个对象之间的任何直接链接。
s.unlinkPath(path)	不适用	除去节点实例之间存在的任何路径。
s.disconnect(node)	不适用	除去所提供节点与指定流中任何其他节点之间的任何链接。
s.isValidLink(source, target)	布尔值	如果在指定的源节点与目标节点之间创建链接是有效的，那么此方法将返回 True。此方法将检查这两个对象是否属于指定流，源节点是否可以提供链接以及目标节点是否可以接收链接，并确认创建此类链接不会在流中引起循环。

下面的示例脚本将执行 5 项任务：

1. 创建"变量文件"输入节点、"过滤"节点和"表"输出节点。
2. 将这些节点连接到一起。
3. 在"变量文件"输入节点上设置文件名。
4. 根据生成的输出过滤字段"药品"。
5. 执行"表"节点。

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", "My File Input ", 96, 64)
filternode = stream.createAt("filter", "Filter", 192, 64)
tablenode = stream.createAt("table", "Table", 288, 64)
stream.link(filenode, filternode)
stream.link(filternode, tablenode)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
filternode.setKeyedPropertyValue("include", "Drug", False)
results = []
tablenode.run(results)
```

导入、替换和删除节点

与创建和连接节点一样，通常需要替换和删除流中的节点。下表概述了可用于导入、替换和删除节点的方法。

表 *14.* 用于导入、替换和删除节点的方法

方法	返回类型	描述
s.replace(originalNode, replacementNode, discardOriginal)	不适用	替换指定流中的指定节点。原始节点和替换节点都必须属于指定流。
s.insert(source, nodes, newIDs)	列表	在所提供的列表中插入节点的副本。假定所提供列表中的所有节点都包含在指定流中。newIDs 标志指示应该为每个节点生成新标识，还是应该复制并使用现有标识。假定流中所有节点都具有唯一标识，这样在源流与指定流相同的情况下，必须将此标志设置为 True。此方法将返回新插入节点的列表，此列表中未定义节点的顺序（即，顺序不一定与输入列表中节点的顺序相同）。
s.delete(node)	不适用	从指定流中删除指定节点。该节点必须属于指定流。
s.deleteAll(nodes)	不适用	从指定流中删除所有指定节点。集合中的所有节点都必须属于指定流。
s.clear()	不适用	从指定流中删除所有节点。

遍历流中的节点

标识特定节点上游或下游的节点是一项常见需求。流提供了一些可用于标识这些节点的方法。下表概述了这些方法。

表 *15.* 用于标识上游节点和下游节点的方法

方法	返回类型	描述
s.iterator()	迭代器	返回针对指定流中包含的节点对象的迭代器。如果在 next() 函数的两次调用之间对流进行了修改，那么将取消定义迭代器的行为。
s.predecessorAt(node, index)	节点(N)	返回所提供节点的指定直接前趋或 None（如果索引超出范围）。
s.predecessorCount(node)	*int*	返回所提供节点的直接前趋数。
s.predecessors(node)	列表	返回所提供节点的直接前趋。
s.successorAt(node, index)	节点(N)	返回所提供节点的指定直接后继或 None（如果索引超出范围）。
s.successorCount(node)	*int*	返回所提供节点的直接后继数。
s.successors(node)	列表	返回所提供节点的直接后继。

清除或除去项

旧脚本编制支持 clear 命令的各种用法，例如：

- clear outputs 用于从管理器选用板中删除所有输出项。
- clear generated palette 用于从模型选用板中清除所有模型块。
- clear stream 用于除去流的内容。

Python 脚本编制支持一组相似的函数；removeAll() 命令用于清除流、输出和模型管理器。例如：

- 清除流管理器：

```
session = modeler.script.session()
session.getStreamManager.removeAll()
```

- 清除输出管理器：

```
session = modeler.script.session()
session.getDocumentOutputManager().removeAll()
```

- 清除模型管理器：

```
session = modeler.script.session()
session.getModelOutputManager().removeAll()
```

获取节点的相关信息

节点分为多种不同的类别，例如数据导入和导出节点、模型构建节点和其他类型的节点。每个节点都提供了一些可用于查找该节点的相关信息的方法。

下表概述了可用于获取节点的标识、名称和标签的方法。

表 *16.* 用于获取节点的标识、名称和标签的方法

方法	返回类型	描述
n.getLabel()	*string*	返回指定节点的显示标签。只有在属性 custom_name 是非空字符串并且属性 use_custom_name 未进行设置的情况下，标签才是前一属性的值；否则，标签是 getName() 的值。
n.setLabel(label)	不适用	设置指定节点的显示标签。如果新标签为非空字符串，那么会将其指定给属性 custom_name，并且会将 False 指定给属性 use_custom_name，以使指定的标签优先；否则，会将空字符串指定给属性 custom_name，并将 True 指定给属性 use_custom_name。
n.getName()	*string*	返回指定节点的名称。
n.getID()	*string*	返回指定节点的标识。每次创建新节点时都会创建一个新标识。将节点作为流的组成部分进行保存时，标识将随节点一起持久存储，以便在打开流时保留节点标识。但是，如果将已保存的节点插入流中，那么会将已插入的节点视为新对象并为其分配一个新标识。

下表概述了可用于获取节点的其他相关信息的方法。

表 *17.* 用于获取节点的相关信息的方法

方法	返回类型	描述
n.getTypeName()	*string*	返回此节点的脚本编制名称。此名称即为可用于创建该节点的新实例的名称。
n.isInitial()	布尔值	如果此节点是初始节点（即，位于流的开始位置的节点），那么此方法将返回 True。
n.isInline()	布尔值	如果此节点是内嵌节点（即，位于流中部的节点），那么此方法将返回 True。
n.isTerminal()	布尔值	如果此节点是终端节点（即，位于流的结束位置的节点），那么此方法将返回 True。
n.getXPosition()	*int*	返回节点在流中的 x 位置偏移量。
n.getYPosition()	*int*	返回节点在流中的 y 位置偏移量。
n.setXYPosition(x, y)	不适用	设置节点在流中的位置。
n.setPositionBetween(source, target)	不适用	设置节点在流中的位置，以使其位于所提供的节点之间。
n.isCacheEnabled()	布尔值	如果已启用高速缓存，那么此方法将返回 True，否则将返回 False。
n.setCacheEnabled(val)	不适用	对此对象启用或禁用高速缓存。如果高速缓存已满并且高速缓存功能进入禁用状态，那么高速缓存将进行清仓。
n.isCacheFull()	布尔值	如果已禁用高速缓存，那么此方法将返回 True，否则将返回 False。
n.flushCache()	不适用	清空此节点的高速缓存。如果高速缓存未启用或者未满，那么此方法不起任何作用。

第 4 章 脚本编制 API

脚本编制 API 简介

脚本编制 API 提供对各种不同的 SPSS Modeler 功能的访问。目前描述的所有方法是 API 的组成部分，可以在脚本内隐式地访问这些方法而不执行进一步的导入。但是，如果要引用 API 类，那么必须使用以下语句显式导入 API：

```
import modeler.api
```

此导入语句是许多脚本编制 API 示例所必需的。

可以在文档 *IBM SPSS Modeler Python Scripting API Reference Guide* 中找到关于通过脚本编制 API 提供的类、方法和参数的完整指南。

示例 1：使用定制过滤器搜索节点

第 27 页的『查找节点』一节提供了关于使用节点的类型名称作为搜索条件在流中搜索节点的示例。在某些情况下，需要执行更加通用的搜索，并且此搜素可使用 NodeFilter 类和流 findAll() 方法来实现。这种搜索包括以下两个步骤：

1. 创建用于扩展 NodeFilter 并实现 accept() 方法的定制版本的新类。
2. 使用此新类的实例来调用流 findAll() 方法。这将返回所有满足 accept() 方法中定义的条件的节点。

以下示例显示如何在流中搜索已启用节点高速缓存的节点。所返回的节点列表可用于清空或禁用这些节点的高速缓存。

```
import modeler.api

class CacheFilter(modeler.api.NodeFilter):
 """A node filter for nodes with caching enabled"""
 def accept(this, node):
  return node.isCacheEnabled()

cachingnodes = modeler.script.stream().findAll(CacheFilter(), False)
```

示例 2：允许用户基于其权限获取目录或文件信息

为避免对用户公开 PSAPI，可使用一种称为 session.getServerFileSystem() 的方法，通过调用 PSAPI 函数来创建文件系统对象。

以下示例显示了如何允许用户基于连接到 IBM SPSS Modeler Server 的用户的权限来获取目录或文件信息。

```
import modeler.api
stream = modeler.script.stream()
sourceNode = stream.findByID('')
session = modeler.script.session()
fileSystem = session.getServerFileSystem()
parameter = stream.getParameterValue('VPATH')
serverDirectory = fileSystem.getServerFile(parameter)
files = fileSystem.getFiles(serverDirectory)
for f in files:
    if f.isDirectory():
        print 'Directory:'
    else:
```

```
        print 'File:'
        sourceNode.setPropertyValue('full_filename',f.getPath())
        break
    print f.getName(),f.getPath()
stream.execute()
```

元数据：关于数据的信息

由于节点在流中连接在一起，因此提供了关于在各个节点提供的列或字段的信息。例如，在 Modeler UI 中，这将使您能够选择作为排序或汇总依据的字段。此信息称为数据模型。

另外，脚本还可以通过查看传入或传出节点的字段来访问数据模型。对于某些节点，输入数据模型与输出数据模型相同，例如，排序节点仅用于对记录进行重新排序，而不更改数据模型。某些节点（例如"派生"节点）可以添加新字段。其他节点（例如"过滤"节点）可以重命名或除去字段。

在下面的示例中，脚本采用标准 IBM SPSS Modeler druglearn.str 流，并为每个字段都构建一个删除了其中一个输入字段的模型。它通过以下过程来完成此操作：

1. 从"类型"节点访问输出数据模型。
2. 对输出数据模型中的各个字段进行遍历。
3. 针对各个输入字段修改"过滤"节点。
4. 更改所构建的模型的名称。
5. 运行模型构建节点。

注：请记住，在 druglean.str 流中运行脚本之前，请先将脚本编制语言设置为 Python（该流在先前版本的 IBM SPSS Modeler 中创建，因此流脚本编制语言设置为旧脚本编制语言）。

```
import modeler.api

stream = modeler.script.stream()
filternode = stream.findByType("filter", None)
typenode = stream.findByType("type", None)
c50node = stream.findByType("c50", None)
# Always use a custom model name
c50node.setPropertyValue("use_model_name", True)

lastRemoved = None
fields = typenode.getOutputDataModel()
for field in fields:
    # If this is the target field then ignore it
    if field.getModelingRole() == modeler.api.ModelingRole.OUT:
        continue

    # Re-enable the field that was most recently removed
    if lastRemoved != None:
        filternode.setKeyedPropertyValue("include", lastRemoved, True)

    # Remove the field
    lastRemoved = field.getColumnName()
    filternode.setKeyedPropertyValue("include", lastRemoved, False)

    # Set the name of the new model then run the build
    c50node.setPropertyValue("model_name", "Exclude " + lastRemoved)
    c50node.run([])
```

DataModel 对象提供了多种用于访问关于数据模型内的字段或列的信息的方法。下表概述了这些方法。

表 *18.* 用于访问关于字段或列的信息的 *DataModel* 对象方法

方法	返回类型	描述
d.getColumnCount()	*int*	返回数据模型中的列数。
d.columnIterator()	迭代器	返回一个迭代器，用于以"自然"插入顺序返回各列。此迭代器将返回列实例。
d.nameIterator()	迭代器	返回一个迭代器，用于以"自然"插入顺序返回各列的名称。
d.contains(name)	布尔值	如果此 DataModel 中存在具有指定名称的列，那么此方法返回 True，否则，返回 False。
d.getColumn(name)	Column	返回具有指定名称的列。
d.getColumnGroup(name)	ColumnGroup	返回指定的列组或 None（如果不存在这样的列组）。
d.getColumnGroupCount()	*int*	返回此数据模型中的列组树。
d.columnGroupIterator()	迭代器	返回一个迭代器，用于依次返回各个列组。
d.toArray()	Column[]	将数据模型作为列阵列返回。列按其"自然"插入顺序进行排序。

每个字段（列对象）都提供了多种用于访问关于列的信息的方法。下表列出了其中一些方法。

表 *19.* 用于访问关于列的信息的列对象方法

方法	返回类型	描述
c.getColumnName()	*string*	返回列的名称。
c.getColumnLabel()	*string*	返回列的标签或空字符串（如果不存在与该列关联的标签）。
c.getMeasureType()	MeasureType	返回列的度量类型。
c.getStorageType()	StorageType	返回列的存储类型。
c.isMeasureDiscrete()	布尔值	返回 True（如果列是独立的）。作为集合或标志的列将被视作独立的列。
c.isModelOutputColumn()	布尔值	返回 True（如果列是模型输出列）。
c.isStorageDatetime()	布尔值	返回 True（如果列的存储为时间、日期或时间戳记值）。
c.isStorageNumeric()	布尔值	返回 True（如果列的存储为整数或实数）。
c.isValidValue(value)	布尔值	返回 True（如果指定的值对于此存储有效）和 valid（如果有效的列值已知）。
c.getModelingRole()	ModelingRole	返回列的建模角色。
c.getSetValues()	Object[]	返回列的有效值阵列或 None（如果值未知或者列不是集合）。
c.getValueLabel(value)	*string*	返回列中的值的标签或空字符串（如果不存在与该值关联的标签）。
c.getFalseFlag()	对象	返回列的"false"指标值或 None（如果值未知或者列不是标志）。

表 *19.* 用于访问关于列的信息的列对象方法 *(续)*

方法	返回类型	描述
c.getTrueFlag()	对象	返回列的"true"指标值或 None（如果值未知或者列不是标志）。
c.getLowerBound()	对象	返回列中的值的下限值或 None（如果值未知或者列不连续）。
c.getUpperBound()	对象	返回列中的值的上限值或 None（如果值未知或者列不连续）。

请注意，大多数用于访问关于列的信息的方法都在 DataModel 对象本身中定义了等效方法。例如，下列两个语句是等效语句：

```
dataModel.getColumn("someName").getModelingRole()
dataModel.getModelingRole("someName")
```

访问已生成的对象

执行某个流通常涉及生成附加输出对象。这些附加对象可能是新模型，也可能是提供要在后续执行中使用的信息的输出。

在以下示例中，druglearn.str 流再次用作流的起始点。在此示例中，将执行流中的所有节点，并且结果将存储在列表中。然后，脚本将遍历这些结果，并且执行所产生的任何模型输出都将保存为 IBM SPSS Modeler 模型 (.gm) 文件，而模型将以 PMML 格式导出。

```
import modeler.api

stream = modeler.script.stream()

# Set this to an existing folder on your system.
# Include a trailing directory separator
modelFolder = "C:/temp/models/"

# Execute the stream
models = []
stream.runAll(models)

# Save any models that were created
taskrunner = modeler.script.session().getTaskRunner()
for model in models:
    # If the stream execution built other outputs then ignore them
    if not(isinstance(model, modeler.api.ModelOutput)):
        continue

    label = model.getLabel()
    algorithm = model.getModelDetail().getAlgorithmName()

    # save each model...
    modelFile = modelFolder + label + algorithm + ".gm"
    taskrunner.saveModelToFile(model, modelFile)

    # ...and export each model PMML...
    modelFile = modelFolder + label + algorithm + ".xml"
    taskrunner.exportModelToFile(model, modelFile, modeler.api.FileFormat.XML)
```

任务运行器类提供了一种运行各项常见任务的便捷方式。下表概述了此类中提供的方法。

表 *20.* 用于执行常见任务的任务运行器类方法

方法	返回类型	描述
t.createStream(name, autoConnect, autoManage)	流	创建并返回新的流。请注意，必须以不公开方式创建流而不向用户显示这些流的代码应该将 autoManage 标志设置为 False。
t.exportDocumentToFile(documentOutput, filename, fileFormat)	不适用	使用指定的文件格式将流描述导出至文件。
t.exportModelToFile(modelOutput, filename, fileFormat)	不适用	使用指定的文件格式将模型导出至文件。
t.exportStreamToFile(stream, filename, fileFormat)	不适用	使用指定的文件格式将流导出至文件。
t.insertNodeFromFile(filename, diagram)	节点(N)	从指定文件中读取并返回节点，然后将其插入所提供的图中。请注意，此方法可用于同时读取节点对象和超节点对象。
t.openDocumentFromFile(filename, autoManage)	文档输出	从指定文件读取并返回文档。
t.openModelFromFile(filename, autoManage)	模型输出	从指定文件读取并返回模型。
t.openStreamFromFile(filename, autoManage)	流	从指定文件读取并返回流。
t.saveDocumentToFile(documentOutput, filename)	不适用	将文档保存到指定的文件位置。
t.saveModelToFile(modelOutput, filename)	不适用	将模型保存到指定的文件位置。
t.saveStreamToFile(stream, filename)	不适用	将流保存到指定的文件位置。

处理错误

Python 语言提供了通过 try...except 代码块执行的错误处理方法。可以在脚本内使用此方法来捕获异常，并处理将导致脚本终止的问题。

在以下示例脚本中，已尝试从 IBM SPSS Collaboration and Deployment Services Repository 中检索模型。此操作可能会导致抛出异常，例如可能未正确设置存储库登录凭证或者存储库路径错误。在此脚本中，这可能会导致抛出 ModelerException（IBM SPSS Modeler 生成的所有异常都派生自 modeler.api.ModelerException）。

```
import modeler.api

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # print goes to the Modeler UI script panel Debug tab
    print "Everything OK"
except modeler.api.ModelerException, e:
    print "An error occurred:", e.getMessage()
```

注：某些脚本编制操作可能会导致抛出标准 Java 异常；这些异常并非派生自 ModelerException。要捕获这些异常，可以使用附加的 except 块来捕获所有 Java 异常，例如：

```
import modeler.api

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # print goes to the Modeler UI script panel Debug tab
    print "Everything OK"
except modeler.api.ModelerException, e:
    print "An error occurred:", e.getMessage()
except java.lang.Exception, e:
    print "A Java exception occurred:", e.getMessage()
```

流、会话和超节点参数

参数提供了一种在运行时传递值而不是在脚本中直接对这些值进行硬编码的有用方式。参数及其值的定义方式与流相同，即定义为流或超节点的参数表中的条目或命令行中的参数。流类和超节点类实现了一组由 ParameterProvider 对象定义的函数，如下表所示。会话提供了 getParameters() 调用，此调用将返回定义这些函数的对象。

表 *21.* 由 *ParameterProvider* 对象定义的函数

方法	返回类型	描述
p.parameterIterator()	迭代器	返回此对象的参数名的迭代器。
p.getParameterDefinition(parameterName)	参数定义	返回具有指定名称的参数的参数定义或 None（如果此提供程序中不存在此类参数）。结果可以是调用此方法时的定义快照，并且不需要反映通过此提供程序对该参数进行的任何后续修改。
p.getParameterLabel(parameterName)	*string*	返回指定参数的标签或 None（如果不存在此类参数）。
p.setParameterLabel(parameterName, label)	不适用	设置指定参数的标签。
p.getParameterStorage(parameterName)	参数存储	返回指定参数的存储或 None（如果不存在此类参数）。
p.setParameterStorage(parameterName, storage)	不适用	设置指定参数的存储。
p.getParameterType(parameterName)	参数类型	返回指定参数的类型或 None（如果不存在此类参数）。
p.setParameterType(parameterName, type)	不适用	设置指定参数的类型。
p.getParameterValue(parameterName)	对象	返回指定参数的值或 None（如果不存在此类参数）。
p.setParameterValue(parameterName, value)	不适用	设置指定参数的值。

在以下示例中，脚本汇总了一些 Telco 数据以查找具有最低平均收入数据的区域。然后，将使用此区域设置一个流参数。接下来，将在"选择"节点中使用该流参数从数据中排除此区域，然后根据剩余的数据构建流失模型。

此示例并不真实，这是因为脚本本身生成了"选择"节点，并因此已经在"选择"节点表达式中直接生成正确的值。但是，流通常是预先构建的，因此通过这种方式设置参数提供了有用的示例。

示例脚本的第一部分用于创建流参数，该流参数将包含平均收入最低的区域。另外，此脚本还将在汇总分支和模型构建分支中创建节点，并将这些节点连接在一起。

```
import modeler.api

stream = modeler.script.stream()

# Initialize a stream parameter
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)

# First create the aggregation branch to compute the average income per region
statisticsimportnode = stream.createAt("statisticsimport", "SPSS File", 114, 142)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/telco.sav")
statisticsimportnode.setPropertyValue("use_field_format_for_storage", True)

aggregatenode = modeler.script.stream().createAt("aggregate", "Aggregate", 294, 142)
aggregatenode.setPropertyValue("keys", ["region"])
aggregatenode.setKeyedPropertyValue("aggregates", "income", ["Mean"])

tablenode = modeler.script.stream().createAt("table", "Table", 462, 142)

stream.link(statisticsimportnode, aggregatenode)
stream.link(aggregatenode, tablenode)

selectnode = stream.createAt("select", "Select", 210, 232)
selectnode.setPropertyValue("mode", "Discard")
# Reference the stream parameter in the selection
selectnode.setPropertyValue("condition", "'region' = '$P-LowestRegion'")

typenode = stream.createAt("type", "Type", 366, 232)
typenode.setKeyedPropertyValue("direction", "churn", "Target")

c50node = stream.createAt("c50", "C5.0", 534, 232)

stream.link(statisticsimportnode, selectnode)
stream.link(selectnode, typenode)
stream.link(typenode, c50node)
```

此示例脚本将创建以下流。

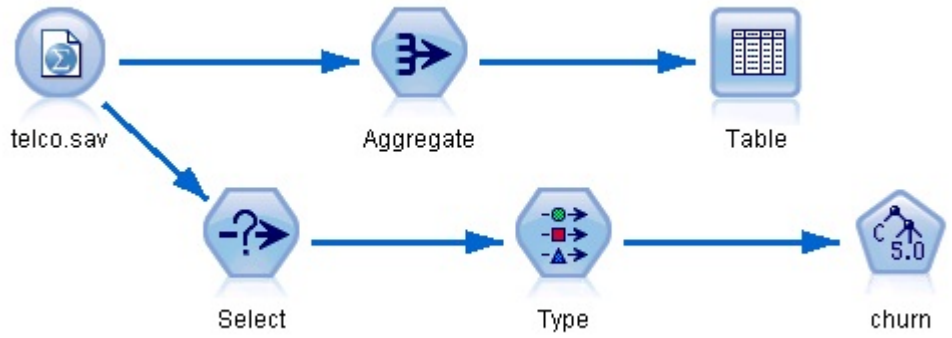

图 *5.* 示例脚本生成的流

示例脚本的以下部分用于执行位于汇总分支末尾的"表"节点。

```
# First execute the table node
results = []
tablenode.run(results)
```

示例脚本的以下部分用于访问执行"表"节点所生成的表输出。随后，此脚本将对表中的各行执行迭代，以查找平均收入最低的区域。

```
# Running the table node should produce a single table as output
table = results[0]

# table output contains a RowSet so we can access values as rows and columns
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None

# From the way the aggregate node is defined, the first column
# contains the region and the second contains the average income
row = 0
rowcount = rowset.getRowCount()
while row < rowcount:
    if rowset.getValueAt(row, 1) < min_income:
        min_income = rowset.getValueAt(row, 1)
        min_region = rowset.getValueAt(row, 0)
    row += 1
```

脚本的以下部分使用平均收入最低的区域来设置先前创建的"LowestRegion"流参数。然后，在从训练数据中排除了指定区域的情况下，此脚本将运行模型构建器。

```
# Check that a value was assigned
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# Finally run the model builder with the selection criteria
c50node.run([])
```

完整的示例脚本如下所示。

```
import modeler.api

stream = modeler.script.stream()

# Create a stream parameter
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)

# First create the aggregation branch to compute the average income per region
statisticsimportnode = stream.createAt("statisticsimport", "SPSS File", 114, 142)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/telco.sav")
statisticsimportnode.setPropertyValue("use_field_format_for_storage", True)

aggregatenode = modeler.script.stream().createAt("aggregate", "Aggregate", 294, 142)
aggregatenode.setPropertyValue("keys", ["region"])
aggregatenode.setKeyedPropertyValue("aggregates", "income", ["Mean"])

tablenode = modeler.script.stream().createAt("table", "Table", 462, 142)

stream.link(statisticsimportnode, aggregatenode)
stream.link(aggregatenode, tablenode)

selectnode = stream.createAt("select", "Select", 210, 232)
selectnode.setPropertyValue("mode", "Discard")
# Reference the stream parameter in the selection
selectnode.setPropertyValue("condition", "'region' = '$P-LowestRegion'")

typenode = stream.createAt("type", "Type", 366, 232)
```

```
typenode.setKeyedPropertyValue("direction", "churn", "Target")

c50node = stream.createAt("c50", "C5.0", 534, 232)

stream.link(statisticsimportnode, selectnode)
stream.link(selectnode, typenode)
stream.link(typenode, c50node)


# First execute the table node
results = []
tablenode.run(results)

# Running the table node should produce a single table as output
table = results[0]

# table output contains a RowSet so we can access values as rows and columns
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None

# From the way the aggregate node is defined, the first column
# contains the region and the second contains the average income
row = 0
rowcount = rowset.getRowCount()
while row < rowcount:
    if rowset.getValueAt(row, 1) < min_income:
        min_income = rowset.getValueAt(row, 1)
        min_region = rowset.getValueAt(row, 0)
    row += 1

# Check that a value was assigned
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# Finally run the model builder with the selection criteria
c50node.run([])
```

全局值

全局值用于计算指定字段的各项汇总统计。可以在流内部的任何位置访问这些汇总值。在流中，可以按名称访问全局值，这一点与流参数相似。全局值与流参数的差异在于，关联值将在"设置全局值"节点运行时自动进行更新，而不是通过脚本编制或命令行来指定。可以通过调用流的 `getGlobalValues()` 方法来访问该流的全局值。

GlobalValues 对象定义下表中显示的函数。

表 *22. GlobalValues* 对象所定义的函数

方法	返回类型	描述
`g.fieldNameIterator()`	迭代器	返回至少具有一个全局值的每个字段名称的迭代器。
`g.getValue(type, fieldName)`	对象	返回指定类型和字段名称的全局值或 None（如果找不到值）。虽然未来的功能可能会返回各种值类型，但通常期望返回值为数字。
`g.getValues(fieldName)`	图	返回包含指定字段名称的已知条目的图或 None（如果该字段没有现有条目）。

GlobalValues.Type 定义可用的汇总统计的类型。可用的汇总统计如下：

- MAX：字段的最大值。
- MEAN：字段的均值。
- MIN：字段的最小值。
- STDDEV：字段的标准差。
- SUM：字段中值的总和。

例如，以下脚本将访问"收入"字段的均值，此均值由"设置全局值"节点计算：

```
import modeler.api

globals = modeler.script.stream().getGlobalValues()
mean_income = globals.getValue(modeler.api.GlobalValues.Type.MEAN, "income")
```

使用多个流 - 独立脚本

要使用多个流，必须使用独立脚本。可以在 IBM SPSS Modeler UI 内编辑和运行独立脚本，也可以在批处理方式下将独立脚本作为命令行参数进行传递。

以下独立脚本将打开两个流。其中一个流用于构建模型，而第二个流用于绘制预测值的分布。

```
# Change to the appropriate location for your system
demosDir = "C:/Program Files/IBM/SPSS/Modeler/18.1.1/DEMOS/streams/"

session = modeler.script.session()
tasks = session.getTaskRunner()

# Open the model build stream, locate the C5.0 node and run it
buildstream = tasks.openStreamFromFile(demosDir + "druglearn.str", True)
c50node = buildstream.findByType("c50", None)
results = []
c50node.run(results)

# Now open the plot stream, find the Na_to_K derive and the histogram
plotstream = tasks.openStreamFromFile(demosDir + "drugplot.str", True)
derivenode = plotstream.findByType("derive", None)
histogramnode = plotstream.findByType("histogram", None)

# Create a model applier node, insert it between the derive and histogram nodes
# then run the histgram
applyc50 = plotstream.createModelApplier(results[0], results[0].getName())
applyc50.setPositionBetween(derivenode, histogramnode)
plotstream.linkBetween(applyc50, derivenode, histogramnode)
histogramnode.setPropertyValue("color_field", "$C-Drug")
histogramnode.run([])

# Finally, tidy up the streams
buildstream.close()
plotstream.close()
```

以下示例显示还可迭代打开的流（"流"选项卡中处于打开状态的所有流）的方式。请注意，仅在独立脚本中才支持此操作。

```
for stream in modeler.script.streams():
    print stream.getName()
```

第 5 章 脚本编制提示

本章简要介绍使用脚本的技巧和方法，包括修改流执行、在脚本中采用加密密码以及访问 IBM SPSS Collaboration and Deployment Services Repository 中的对象等。

修改流执行

运行流时，将按缺省情形下的优化顺序来执行其终端节点。某些情况下，您可能更喜欢以其他顺序来执行。要修改流的执行顺序，请在流属性对话框的"执行"选项卡上完成以下步骤：

1. 打开一个空脚本。
2. 单击工具栏上的**追加缺省脚本**按钮来添加缺省流脚本。
3. 将缺省流脚本中语句的顺序更改为您希望的执行顺序。

对节点执行循环

您可以使用 for 循环对流中的所有节点进行循环。例如，以下两个脚本示例用于对所有节点进行循环并将"过滤"节点中的字段名更改为大写。

可以在具有"过滤"节点的任何流中使用此脚本，即使实际上不过滤任何字段也是如此。只需添加传递所有字段的"过滤"节点即可将整个面板中的字段名称更改为大写。

```
# Alternative 1: using the data model nameIterator() function
stream = modeler.script.stream()
for node in stream.iterator():
    if (node.getTypeName() == "filter"):
        # nameIterator() returns the field names
        for field in node.getInputDataModel().nameIterator():
            newname = field.upper()
            node.setKeyedPropertyValue("new_name", field, newname)
```

```
# Alternative 2: using the data model iterator() function
stream = modeler.script.stream()
for node in stream.iterator():
    if (node.getTypeName() == "filter"):
        # iterator() returns the field objects so we need
        # to call getColumnName() to get the name
        for field in node.getInputDataModel().iterator():
            newname = field.getColumnName().upper()
            node.setKeyedPropertyValue("new_name", field.getColumnName(), newname)
```

此脚本在当前流的所有节点中进行循环，并检查每个节点是否为过滤节点。如果是，那么脚本将循环该节点中的每个字段，并使用 field.upper() 或 field.getColumnName().upper() 函数将名称更改为大写。

访问 IBM SPSS Collaboration and Deployment Services Repository 中的对象

如果您具有 IBM SPSS Collaboration and Deployment Services Repository 许可证，那么可以使用脚本命令在存储库中存储以及从中检索对象。使用存储库在企业应用程序、工具和解决方案环境中对数据挖掘模型和相关预测对象的生命周期进行管理。

连接到 IBM SPSS Collaboration and Deployment Services Repository

要访问存储库，必须首先通过 SPSS Modeler 用户界面的**工具**菜单或使用命令行建立到该存储库的有效连接。有关更多信息，请参阅第 58 页的『IBM SPSS Collaboration and Deployment Services Repository 连接自变量』。

访问存储库

可从会话访问存储库，例如：

```
repo = modeler.script.session().getRepository()
```

从存储库检索对象

在脚本中，使用 `retrieve*` 函数来访问各种对象，包括流、模型、输出和节点。下表中显示检索函数的摘要。

表 *23.* 检索脚本编制函数

对象类型	存储库函数
流	repo.retrieveStream(String path, String version, String label, Boolean autoManage)
模型	repo.retrieveModel(String path, String version, String label, Boolean autoManage)
输出	repo.retrieveDocument(String path, String version, String label, Boolean autoManage)
节点(N)	repo.retrieveProcessor(String path, String version, String label, ProcessorDiagram diagram)

例如，您可以使用以下函数从存储库检索流：

```
stream = repo.retrieveStream("/projects/retention/risk_score.str", None, "production", True)
```

此示例从指定的文件夹检索 `risk_score.str` 流。标签 `production` 标识要检索的流版本，最后一个参数指定 SPSS Modeler 将管理流（例如，如果 SPSS Modeler 用户界面可见，流在**流**选项卡中显示）。作为替代方法，要使用特定未删除的版本：

```
stream = repo.retrieveStream("/projects/retention/risk_score.str", "0:2015-10-12 14:15:41.281", None, True)
```

注：如果版本和标签参数都为 `None`，那么将返回最新版本。

在存储库中存储对象

要使用脚本编制在存储库中存储对象，请使用 `store*` 函数。下表中显示存储函数的摘要。

表 *24.* 存储脚本编制函数

对象类型	存储库函数
流	repo.storeStream(ProcessorStream stream, String path, String label)
模型	repo.storeModel(ModelOutput modelOutput, String path, String label)
输出	repo.storeDocument(DocumentOutput documentOutput, String path, String label)
节点(N)	repo.storeProcessor(Processor node, String path, String label)

例如，您可以使用以下函数存储新版本的 `risk_score.str` 流：

```
versionId = repo.storeStream(stream, "/projects/retention/risk_score.str", "test")
```

此示例存储新版本的流、将 `"test"` 标签与其相关联，并返回新创建的版本的版本标记。

注：如果不想将标签与新版本相关联，请针对标签传递 None。

管理存储库文件夹

通过使用存储库中的文件夹，您可以将对象组织到逻辑组并使其更易于查看相关的对象。使用 createFolder() 函数创建文件夹，如以下示例中所示：

```
newpath = repo.createFolder("/projects", "cross-sell")
```

此示例在 "/projects" 文件夹中创建名为 "cross-sell" 的新文件夹。函数将返回新文件夹的完整路径。

要重命名文件夹，请使用 renameFolder() 函数：

```
repo.renameFolder("/projects/cross-sell", "cross-sell-Q1")
```

第一个参数是要重命名的文件夹的完整路径，而第二个是要为此文件夹指定的新名称。

要删除空文件夹，请使用 deleteFolder() 函数：

```
repo.deleteFolder("/projects/cross-sell")
```

锁定和解锁对象

对于脚本，您可以锁定一个对象，以防止其他用户更新任一现有版本或新建版本。还可以解锁已锁定的对象。

锁定和解锁对象的语法为：

```
repo.lockFile(REPOSITORY_PATH)
repo.lockFile(URI)

repo.unlockFile(REPOSITORY_PATH)
repo.unlockFile(URI)
```

对于存储和检索对象，REPOSITORY_PATH 指出对象在存储库中的位置。路径必须用英文引号引起并以正斜杠作为分隔符。路径不区分大小写。

```
repo.lockFile("/myfolder/Stream1.str")
repo.unlockFile("/myfolder/Stream1.str")
```

除此之外，还可以使用统一资源标识 (URI) 而非存储库路径来给出对象的位置。URI 必须包含前缀 spsscr:，同时必须完全括在引号中。只有正斜杠可以作为路径分隔符，空格必须以编码形式出现。即在路径中以 %20 代替空格。URI 不区分大小写。示例如下：

```
repo.lockFile("spsscr:///myfolder/Stream1.str")
repo.unlockFile("spsscr:///myfolder/Stream1.str")
```

注意，对象锁定适用于对象的所有版本 - 您无法锁定或解锁单个版本。

生成加密密码

某些情况下，可能需要在脚本中包含密码，例如，您可能需要访问受密码保护的数据源。加密密码可用在：

- 数据库源和输出节点的节点属性
- 登录到服务器的命令行参数
- 存储在 *.par* 文件（由导出节点的"发布"选项卡生成的参数文件）中的数据库连接属性

通过此用户界面，可以使用一个工具根据 Blowfish 算法来生成加密密码（有关详细信息，请参阅 http://www.schneier.com/blowfish.html 以获取更多信息）。进行编码后，可以复制密码并将其存储到脚本文件和命令行自变量中。用于 databasenode 和 databaseexportnode 的节点属性 epassword 存储加密密码。

1. 要生成加密密码，请从"工具"菜单中选择：

   对密码进行编码...
2. 在"密码"文本框中指定一个密码。
3. 单击**编码**，以便为您的密码生成随机编码。
4. 单击"复制"按钮将加密密码复制到剪贴板。
5. 将此密码粘贴到所需的脚本或参数中。

脚本检查

通过单击"独立脚本"对话框工具栏上的红色检查按钮，可以快速检查所有类型脚本的语法。

图 *6.* 流脚本工具栏图标

脚本检查将就您编码中的错误发出警报并给出改进建议。要查看错误行，请单击该对话框下半部分的反馈。此时将以红色突出显示错误。

从命令行编写脚本

通过编写脚本可以运行通常在用户界面中执行的操作。启动 IBM SPSS Modeler 时，只需在命令行中指定和运行一个独立流。例如：

```
client -script scores.txt -execute
```

`-script` 标记表示装入指定脚本，而 `-execute` 标记表示执行该脚本文件中的所有命令。

与早期版本的兼容性

在以前版本的 IBM SPSS Modeler 中创建的脚本通常应该无需更改就可以在当前版本中运行。不过，模型块现在可以自动插入到流中（此为默认设置），并可替代或补充流中此类型的现有模型块。实际发生的行为取决于**将模型添加到流中**和**替换原有模型**选项（**工具 > 选项 > 用户选项 > 通知**）的设置。例如，您可能需要修改以前版本中的脚本，在该版本中模型块替换是通过删除现有模型块并插入新的模型块来完成。

在当前版本中创建的脚本在以前的版本中可能无法正常运行。

如果在旧版本中创建的脚本使用了已被替换（或不被支持）的命令，那么使用旧形式命令的脚本仍然会得到支持，但将显示一条警告消息。例如，旧的 `generated` 关键字已被 `model` 替换，且 `clear generated` 已被 `clear generated palette` 替换。沿用旧形式的脚本仍然可以运行，但将显示一条警告消息。

访问流执行结果

许多 IBM SPSS Modeler 节点生成模型、图表和表数据等输出对象。在这些输出中，许多输出都包含可由脚本用于指导后续执行的有用值。这些值分组到内容容器（简称为容器）中，您可以通过用于识别各个容器的标记或标识来访问这些容器。访问这些值的方式取决于该容器使用的格式或"内容模型"。

例如，许多预测模型输出使用称为 PMML 的 XML 变体来表示关于模型的信息，例如决策树在每个拆分点使用哪些字段，或者神经网络中的神经元如何连接以及以何种强度进行连接。使用 PMML 的模型输出提供可用于访问此信息的 XML 内容模型。例如：

```
stream = modeler.script.stream()
# Assume the stream contains a single C5.0 model builder node
# and that the datasource, predictors and targets have already been
# set up
modelbuilder = stream.findByType("c50", None)
results = []
modelbuilder.run(results)
modeloutput = results[0]

# Now that we have the C5.0 model output object, access the
# relevant content model
cm = modeloutput.getContentModel("PMML")

# The PMML content model is a generic XML-based content model that
# uses XPath syntax. Use that to find the names of the data fields.
# The call returns a list of strings match the XPath values
dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField", "name")
```

IBM SPSS Modeler 在脚本编制中支持以下内容模型：

- **表内容模型**提供对表示为行和列的简单表数据的访问
- **XML 内容模型**提供对以 XML 格式存储的内容的访问
- **JSON 内容模型**提供对以 JSON 格式存储的内容的访问
- **列统计内容模型**提供对有关特定字段的汇总统计的访问
- **成对列统计内容模型**用于访问两个字段的摘要统计或者两个单独字段的值

表内容模型

表内容模型提供用于访问简单行和列数据的简单模型。特定列中的值必须全部具有同一存储类型（例如，字符串或整数）。

API

表 25. *API*

返回	方法	描述
int	getRowCount()	返回此表中的行数。
int	getColumnCount()	返回此表中的列数。
字符串	getColumnName(int columnIndex)	返回位于指定列索引的列的名称。列索引从 0 开始。
StorageType	getStorageType(int columnIndex)	返回位于指定索引的列的存储类型。列索引从 0 开始。
对象	getValueAt(int rowIndex, int columnIndex)	返回位于指定行和列索引的值。行和列索引从 0 开始。
void	reset()	清空与词内容模型关联的任何内部存储器。

节点和输出

下表列出了用于构建包含此类内容模型的输出的节点。

表 *26.* 节点和输出

节点名	输出名称	容器标识
表	表	"table"

示例脚本

```
stream = modeler.script.stream()
from modeler.api import StorageType

# Set up the variable file import node
varfilenode = stream.createAt("variablefile", "DRUG Data", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")

# Next create the aggregate node and connect it to the variable file node
aggregatenode = stream.createAt("aggregate", "Aggregate", 192, 96)
stream.link(varfilenode, aggregatenode)

# Configure the aggregate node
aggregatenode.setPropertyValue("keys", ["Drug"])
aggregatenode.setKeyedPropertyValue("aggregates", "Age", ["Min", "Max"])
aggregatenode.setKeyedPropertyValue("aggregates", "Na", ["Mean", "SDev"])

# Then create the table output node and connect it to the aggregate node
tablenode = stream.createAt("table", "Table", 288, 96)
stream.link(aggregatenode, tablenode)

# Execute the table node and capture the resulting table output object
results = []
tablenode.run(results)
tableoutput = results[0]

# Access the table output's content model
tablecontent = tableoutput.getContentModel("table")

# For each column, print column name, type and the first row
# of values from the table content
col = 0
while col < tablecontent.getColumnCount():
     print tablecontent.getColumnName(col), \
     tablecontent.getStorageType(col), \
     tablecontent.getValueAt(0, col)
     col = col + 1
```

"脚本调试"选项卡中的输出类似于以下内容：

```
Age_Min Integer 15
Age_Max Integer 74
Na_Mean Real 0.730851098901
Na_SDev Real 0.116669731242
Drug String drugY
Record_Count Integer 91
```

XML 内容模型

XML 内容模型提供对基于 XML 的内容的访问。

XML 内容模型使用户能够访问基于 XPath 表达式的组件。XPath 表达式是用于定义调用者需要哪些元素或属性的字符串。XML 内容模型隐藏了构造各种对象以及编译表达式的细节，而 XPath 支持人员通常需要这些细节。这使得通过 Python 脚本编制进行调用更为简单。

XML 内容模型提供用于将 XML 文档以字符串形式返回的函数。这允许 Python 脚本用户使用其首选 Python 库来解析 XML。

API

表 *27. API*

返回	方法	描述
字符串	`getXMLAsString()`	以字符串形式返回 XML。
数字	`getNumericValue(String xpath)`	返回使用数字返回类型对路径进行求值的结果（例如，对与路径表达式相匹配的元素的数量进行计数）。
布尔值	`getBooleanValue(String xpath)`	返回对所指定路径表达式求值所得的布尔结果。
字符串	`getStringValue(String xpath, String attribute)`	返回与指定路径相匹配的属性值或 XML 节点值。
字符串列表	`getStringValues(String xpath, String attribute)`	返回一个列表，其中包含与指定路径相匹配的所有属性值或 XML 节点值。
字符串列表的列表	`getValuesList(String xpath, <List of strings> attributes, boolean includeValue)`	返回一个列表，其中包含与指定路径相匹配的所有属性值以及 XML 节点值（如果需要）。
散列表（key:string，value:list of string）	`getValuesMap(String xpath, String keyAttribute, <List of strings> attributes, boolean includeValue)`	返回一个散列表，该表将关键字属性或 XML 节点值用作关键字，并将指定属性值的列表用作表值。
布尔值	`isNamespaceAware()`	返回 XML 解析器是否应知道名称空间。缺省值为 `False`。
`void`	`setNamespaceAware(boolean value)`	设置 XML 解析器是否应知道名称空间。这还将调用 `reset()` 以确保后续调用应用更改。
`void`	`reset()`	清空与此内容模型（例如，高速缓存的 DOM 对象）关联的任何内部存储器。

节点和输出

下表列出了用于构建包含此类内容模型的输出的节点。

表 *28.* 节点和输出

节点名	输出名称	容器标识
大多数模型构建器	生成次数最多的模型	`"PMML"`
`"autodataprep"`	`n/a`	`"PMML"`

示例脚本

用于访问内容的 Python 脚本编制代码可能类似于以下内容：

```
results = []
modelbuilder.run(results)
modeloutput = results[0]
cm = modeloutput.getContentModel("PMML")

dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField", "name")
predictedNames = cm.getStringValues("//MiningSchema/MiningField[@usageType='predicted']", "name")
```

JSON 内容模型

JSON 内容模型用于提供对 JSON 格式内容的支持。这提供了基本 API，以允许调用者在假定它们知道要访问哪些值的情况下抽取值。

API

表 *29. API*

返回	方法	描述
字符串	`getJSONAsString()`	以字符串形式返回 JSON 内容。
对象	`getObjectAt(<List of cbjecta> path, JSONArtifact artifact)` 抛出异常	返回指定路径处的对象。所提供的 root 工件可能为空，在这种情况下将使用内容的根。返回的值可能是一个文字字符串、整数、实数或布尔值或者是 JSON 工件（JSON 对象或 JSON 数组）。
散列表（key:object，value:object）	`getChildValuesAt(<List of object> path, JSONArtifact artifact)` 抛出异常	如果指定路径指向 JSON 对象，那么将返回该路径的子值，否则返回 null。表中的关键字是字符串，而关联值可能是一个文字字符串、整数、实数或布尔值或者是 JSON 工件（JSON 对象或 JSON 数组）。
对象列表	`getChildrenAt(<List of object> path path, JSONArtifact artifact)` 抛出异常	如果指定路径指向 JSON 数组，那么将返回位于该路径的对象的列表，否则返回 null。返回的值可能是一个文字字符串、整数、实数或布尔值或者是 JSON 工件（JSON 对象或 JSON 数组）。
`void`	`reset()`	清空与此内容模型（例如，高速缓存的 DOM 对象）关联的任何内部存储器。

示例脚本

如果存在根据 JSON 格式创建的输出构建器节点，那么将使用以下脚本来访问一组书籍的相关信息：

```
results = []
outputbuilder.run(results)
output = results[0]
cm = output.getContentModel("jsonContent")

bookTitle = cm.getObjectAt(["books", "ISIN123456", "title"], None)

# Alternatively, get the book object and use it as the root
# for subsequent entries
book = cm.getObjectAt(["books", "ISIN123456"], None)
bookTitle = cm.getObjectAt(["title"], book)

# Get all child values for aspecific book
```

```
bookInfo = cm.getChildValuesAt(["books", "ISIN123456"], None)

# Get the third book entry. Assumes the top-level "books" value
# contains a JSON array which can be indexed
bookInfo = cm.getObjectAt(["books", 2], None)

# Get a list of all child entries
allBooks = cm.getChildrenAt(["books"], None)
```

列统计信息内容模型和成对比较统计信息内容模型

列统计内容模型提供对可以为每个字段（单变量统计）计算的统计的访问。成对统计内容模型提供对可以在成对的字段或某个字段的多个值之间计算的统计的访问。

可能的统计度量包括：

- 计数
- 唯一计数
- 有效计数
- 均数
- 合计
- 最小值
- 最大值
- 范围
- Variance
- 标准差
- 平均值的标准误差
- 偏度
- 偏度标准误差
- 峰度
- 峰度标准误差
- 中位数
- 众数
- Pearson
- 协方差
- T 检验
- F 检验

某些值仅适用于单列统计，而其他值仅适用于成对统计。

生成这些统计的节点为：

- **"统计"节点**生成列统计并且可以在指定相关字段的情况下生成成对统计
- **"数据审核"节点**生成列统计并且可以在指定交叠字段的情况下生成成对统计。
- **"均值"节点**在比较字段对或将字段的值与其他字段摘要进行比较时生成成对统计。

哪些内容模型和统计可用将取决于特定节点的功能和该节点中的设置。

ColumnStatsContentModel API

表 30. *ColumnStatsContentModel API*.

返回	方法	描述
List<StatisticType>	getAvailableStatistics()	返回此模型中的可用统计。并非所有字段都必须具有所有统计的值。
List<String>	getAvailableColumns()	返回已为其计算统计的列名。
Number	getStatistic(String column, StatisticType statistic)	返回与列关联的统计值。
void	reset()	清空与词内容模型关联的任何内部存储器。

PairwiseStatsContentModel API

表 31. *PairwiseStatsContentModel API*.

返回	方法	描述
List<StatisticType>	getAvailableStatistics()	返回此模型中的可用统计。并非所有字段都必须具有所有统计的值。
List<String>	getAvailablePrimaryColumns()	返回已为其计算统计的主要列名。
List<Object>	getAvailablePrimaryValues()	返回已为其计算统计的主要列的值。
List<String>	getAvailableSecondaryColumns()	返回已为其计算统计的次要列名。
Number	getStatistic(String primaryColumn, String secondaryColumn, StatisticType statistic)	返回与列关联的统计值。
Number	getStatistic(String primaryColumn, Object primaryValue, String secondaryColumn, StatisticType statistic)	返回与主要列值及次要列关联的统计值。
void	reset()	清空与词内容模型关联的任何内部存储器。

节点和输出

下表列出了用于构建包含此类内容模型的输出的节点。

表 32. 节点和输出.

节点名	输出名称	容器标识	附注
"means" ("均值"节点)	"means"	"columnStatistics"	
"means" ("均值"节点)	"means"	"pairwiseStatistics"	
"dataaudit" ("数据审核"节点)	"means"	"columnStatistics"	
"statistics" ("统计"节点)	"statistics"	"columnStatistics"	仅在检查特定字段时生成。
"statistics" ("统计"节点)	"statistics"	"pairwiseStatistics"	仅在字段相关时生成。

示例脚本

```
from modeler.api import StatisticType
stream = modeler.script.stream()

# Set up the input data
varfile = stream.createAt("variablefile", "File", 96, 96)
varfile.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")

# Now create the statistics node. This can produce both
# column statistics and pairwise statistics
statisticsnode = stream.createAt("statistics", "Stats", 192, 96)
statisticsnode.setPropertyValue("examine", ["Age", "Na", "K"])
statisticsnode.setPropertyValue("correlate", ["Age", "Na", "K"])
stream.link(varfile, statisticsnode)

results = []
statisticsnode.run(results)
statsoutput = results[0]
statscm = statsoutput.getContentModel("columnStatistics")
if (statscm != None):
 cols = statscm.getAvailableColumns()
 stats = statscm.getAvailableStatistics()
 print "Column stats:", cols[0], str(stats[0]), " = ", statscm.getStatistic(cols[0], stats[0])
statscm = statsoutput.getContentModel("pairwiseStatistics")
if (statscm != None):
 pcols = statscm.getAvailablePrimaryColumns()
 scols = statscm.getAvailableSecondaryColumns()
 stats = statscm.getAvailableStatistics()
 corr = statscm.getStatistic(pcols[0], scols[0], StatisticType.Pearson)
 print "Pairwise stats:", pcols[0], scols[0], " Pearson = ", corr
```

第 6 章 命令行自变量

调用软件

您可以使用操作系统的命令行来如下启动 IBM SPSS Modeler：

1. 在安装了 IBM SPSS Modeler 的计算机上，打开 DOS 或命令提示符窗口。
2. 要以交互方式启动 IBM SPSS Modeler 界面，请输入 `modelerclient` 命令，然后输入所需的参数；例如：

```
modelerclient -stream report.str -execute
```

可用参数（标记）允许您连接到服务器、装入流、运行脚本或根据需要指定其他参数。

使用命令行自变量

您可以将命令行自变量（也称为标记）附加到最初的 `modelerclient` 命令以更改对 IBM SPSS Modeler 的调用。

存在多种可用的命令行自变量类型，本节的随后内容将对其进行描述。

表 *33.* 命令行自变量类型.

自变量类型	描述位置
系统自变量	有关更多信息，请参阅第 56 页的『系统自变量』主题。
参数自变量	有关更多信息，请参阅第 57 页的『参数自变量』主题。
服务器连接自变量	有关更多信息，请参阅第 57 页的『服务器连接自变量』主题。
IBM SPSS Collaboration and Deployment Services Repository 连接自变量	有关更多信息，请参阅第 58 页的『IBM SPSS Collaboration and Deployment Services Repository 连接自变量』主题。
IBM SPSS Analytic Server 连接自变量	请参阅主题第 59 页的『IBM SPSS Analytic Server 连接自变量』以获取更多信息。

例如，可以使用 `-server`、`-stream` 和 `-execute` 标记来连接到服务器，然后装入并运行流，如下所示：

```
modelerclient -server -hostname myserver -port 80 -username dminer
-password 1234 -stream mystream.str -execute
```

请注意，针对本地客户机安装运行时，不需要指定服务器连接自变量。

可以用双引号括起包含空格的参数值，例如：

```
modelerclient -stream mystream.str -Pusername="Joe User" -execute
```

还可以用此种方式执行 IBM SPSS Modeler 状态和脚本，但要分别使用 `-state` 和 `-script` 标记。

注：如果您在命令中使用结构化参数，那么必须在引号之前加上反斜杠。这可以避免在解释字符串的过程中除去引号。

调试命令行自变量

要调试命令行，请使用 modelerclient 命令在指定所需自变量的情况下启动 IBM SPSS Modeler。这样可以验证命令是否将按期望方式执行。另外，您还可以在"会话参数"对话框（"工具"菜单 ->"设置会话参数"）中对通过命令行传递的任何参数的值进行确认。

系统自变量

下表描述可用于用户界面命令行调用的系统自变量。

表 *34.* 系统自变量

自变量	行为/描述
@ <commandFile>	@ 符号后跟文件名，此文件用于指定命令列表。当 modelerclient 遇到以 @ 开头的参数时，它将在该文件中对命令进行操作，就如同在命令行中一样。请参阅主题第 59 页的『组合多个参数』，了解更多信息。
-directory <dir>	设置缺省工作目录。在本地模式下，该目录将同时用于数据操作和输出。示例：-directory c:/ or -directory c:\\
-server_directory <dir>	为数据设置缺省服务器目录。通过 -directory 标记指定的工作目录将用于输出。
-execute	在启动后执行启动时所加载的流、状态或脚本。如果在流或状态之外还加载了脚本，则脚本将单独执行。
-stream <stream>	启动时加载指定的流。可以指定多个流，但是最后一个指定的流将被设置为当前流。
-script <script>	启动时加载指定的独立脚本。如下所述，除流或状态之外，此标记还可用于指定脚本，但在启动时仅可加载一个脚本。
-model <model>	在启动时加载指定的已生成模型（.gm 格式的文件）。
-state <state>	在启动时，加载指定的已保存状态。
-project <project>	加载指定工程。在启动时仅可加载一个工程。
-output <output>	在启动时加载已保存的输出项目（.cou 格式的文件）。
-help	显示命令行自变量列表。指定此选项后，将忽略所有其它参数并显示帮助屏幕。
-P <name>=<value>	用于设置启动参数。还可用于设置节点属性（通道参数）。

注：也可以在用户界面中设置缺省目录。要访问上述选项，请在"文件"菜单中选择**设置工作目录**或**设置服务器目录**。

加载多个文件

命令行模式下，您可以通过在启动时重复输入每个加载对象的相关参数来加载多个流、状态和输出。例如，要加载和运行两个称为 report.str 和 train.str 的流，您可以使用如下命令：

```
modelerclient -stream report.str -stream train.str -execute
```

从 IBM SPSS Collaboration and Deployment Services Repository 加载对象

因为可以从某个文件或 IBM SPSS Collaboration and Deployment Services Repository（如果已获许可）加载特定对象，可以使用文件名前缀 spsscr: 以及选择性地使用 file:（对于磁盘上的对象）来指示 IBM SPSS Modeler 在什么位置查找对象。前缀可与以下标记配合使用：

- -stream
- -script
- -output
- -model

- -project

您可以使用前缀创建 URI 以指定对象的位置，例如 -stream "spsscr:///folder_1/scoring_stream.str"。如果指定了 spsscr: 前缀，那么要求已在同一命令中指定了有效的 IBM SPSS Collaboration and Deployment Services Repository 连接。因此，完整的命令应形如以下的示例：

```
modelerclient -spsscr_hostname myhost -spsscr_port 8080
-spsscr_username myusername -spsscr_password mypassword
-stream "spsscr:///folder_1/scoring_stream.str" -execute
```

注意，在命令行中 必须使用 URI。不支持像 REPOSITORY_PATH 这样的简单路径。（此种路径仅适用于脚本。）有关 IBM SPSS Collaboration and Deployment Services Repository 中对象的 URI 的详细信息，请参阅第 43 页的『访问 IBM SPSS Collaboration and Deployment Services Repository 中的对象』主题。

参数自变量

参数可用作在 IBM SPSS Modeler 的命令行执行期间的标记。在命令行自变量中，-P 标记用于表示形如 -P *<name>=<value>* 的参数。

形式参数可以是：

- **简单参数**（即，直接在 CLEM 表达式中使用的参数）。
- **槽参数**，也称为**节点属性**。此类参数用于修改流中各个节点的设置。请参阅主题第 63 页的『节点属性概述』以获取更多信息。
- **命令行自变量**，用于更改对 IBM SPSS Modeler 的调用。

例如，您可以提供数据源用户名和密码作为命令行标志，如下所示：

```
modelerclient -stream response.str -P:databasenode.datasource="{\"ORA 10gR2\",user1,mypsw,false}"
```

其格式与 databasenode 节点属性的 datasource 参数相同。有关更多信息，请参阅：第 75 页的『databasenode 属性』。

如果传递经过编码的密码，那么最后一个参数应该设置为 true。另请注意，数据库用户名和密码前不应该使用任何前导空格（除非，您的用户名或密码确实包含前导空格）。

注：如果已命名节点，那么您必须将节点名括在双引号中并使用反斜杠对引号进行转义。例如，如果以上示例中数据源节点的名称为 Source_ABC，那么此条目将如下所示：

```
modelerclient -stream response.str -P:databasenode.\"Source_ABC\".datasource="{\"ORA 10gR2\",
 user1,mypsw,true}"
```

用于标识结构化参数的引号前还需要有反斜杠，如以下 TM1 数据源示例中所示：

```
clemb -server -hostname 9.115.21.169 -port 28053 -username administrator
    -execute -stream C:\Share\TM1_Script.str -P:tm1import.pm_host="http://9.115.21.163:9510/pmhub/pm"
    -P:tm1import.tm1_connection={\"SData\",\"\",\"admin\",\"apple\"}
    -P:tm1import.selected_view={\"SalesPriorCube\",\"salesmargin%\"}
```

服务器连接自变量

-server 标记指示 IBM SPSS Modeler 应连接到公共服务器，标记 -hostname、-use_ssl、-port、-username、-password 和 -domain 用于指示 IBM SPSS Modeler 如何连接到公共服务器。如果未指定 -server 参数，那么使用缺省 或本地 服务器。

示例

连接到公共服务器：

```
modelerclient -server -hostname myserver -port 80 -username dminer
-password 1234 -stream mystream.str -execute
```

连接到服务器集群：

```
modelerclient -server -cluster "QA Machines" \
-spsscr_hostname pes_host -spsscr_port 8080 \
-spsscr_username asmith -spsscr_epassword xyz
```

请注意，连接到服务器集群需要通过在整个 IBM SPSS Collaboration and Deployment Services 中使用过程协调器，因此 -cluster 参数必须与存储库连接选项 (spsscr_*) 结合使用。有关更多信息，请参阅『IBM SPSS Collaboration and Deployment Services Repository 连接自变量』主题。

表 *35.* 服务器连接自变量.

自变量	行为/描述
-server	以服务器模式运行 IBM SPSS Modeler，同时使用标志 -hostname、-port、-username、-password 和 -domain 连接到公共服务器。
-hostname<name>	服务器的主机名。仅在服务器方式下可用。
-use_ssl	指定连接应使用 SSL（安全套接字层）。此标记为可选项，缺省设置为不使用 SSL。
-port<number>	所指定服务器的端口号。仅在服务器方式下可用。
-cluster<name>	指定指向服务器集群（而不是已命名的服务器）的连接；此参数可用来替代 hostname、port 和 use_ssl 参数。名称为聚类名，或标识 IBM SPSS Collaboration and Deployment Services Repository 中聚类的唯一 URI。服务器集群由 IBM SPSS Collaboration and Deployment Services 中的过程协调器管理。有关更多信息，请参阅『IBM SPSS Collaboration and Deployment Services Repository 连接自变量』主题。
-username<name>	这是用于登录服务器的用户名。仅在服务器方式下可用。
-password<password>	这是用于登录服务器的密码。仅在服务器方式下可用。 **注：**如果未使用 -password 自变量，那么系统将提示您输入密码。
-epassword<encodedpasswordstring>	这是用于登录服务器的经过编码的密码。仅在服务器方式下可用。 **注：**可以通过 IBM SPSS Modeler 应用程序的"工具"菜单生成经过编码的密码。
-domain<name>	这是用于登录服务器的域。仅在服务器方式下可用。
-P <name>=<value>	用于设置启动参数。另外，还可用于设置节点属性（槽参数）。

IBM SPSS Collaboration and Deployment Services Repository 连接自变量

如果想通过命令行来存储或检索 IBM SPSS Collaboration and Deployment Services 中的对象，那么必须指定一个指向该 IBM SPSS Collaboration and Deployment Services Repository 的有效连接。例如：

```
modelerclient -spsscr_hostname myhost -spsscr_port 8080
-spsscr_username myusername -spsscr_password mypassword
-stream "spsscr:///folder_1/scoring_stream.str" -execute
```

下表列出了可用于建立连接的自变量。

表 *36. IBM SPSS Collaboration and Deployment Services Repository* 连接自变量

自变量	行为/描述
-spsscr_hostname <hostname or IP address>	安装 IBM SPSS Collaboration and Deployment Services Repository 的服务器的主机名或 IP 地址。

表 *36. IBM SPSS Collaboration and Deployment Services Repository* 连接自变量 *(续)*

自变量	行为/描述
-spsscr_port <number>	IBM SPSS Collaboration and Deployment Services Repository 接受连接的端口号（通常，缺省值为 8080）。
-spsscr_use_ssl	指定连接应使用 SSL（安全套接字层）。此标记为可选项，缺省设置为不使用 SSL。
-spsscr_username<name>	登录到 IBM SPSS Collaboration and Deployment Services Repository 的用户名。
-spsscr_password<password>	登录到 IBM SPSS Collaboration and Deployment Services Repository 的密码。
-spsscr_epassword<encoded password>	登录到 IBM SPSS Collaboration and Deployment Services Repository 的加密密码。
-spsscr_providername <name>	用于登录到 IBM SPSS Collaboration and Deployment Services Repository（Active Directory 或 LDAP）的认证服务提供程序。如果使用本机（本地存储库）提供程序，那么这不是必需的。

IBM SPSS Analytic Server 连接自变量

如果要通过命令行存储或检索 IBM SPSS Analytic Server 中的对象，那么必须指定与 IBM SPSS Analytic Server 的有效连接。

注：Analytic Server 的缺省位置是从 SPSS Modeler Server 获取的。用户还可以连接通过**工具** > **Analytic Server 连接**定义自己的 Analytic Server 。

下表列出了可用于建立连接的自变量。

表 *37. IBM SPSS Analytic Server* 连接自变量

自变量	行为/描述
-analytic_server_username	用于登录 IBM SPSS Analytic Server 的用户名。
-analytic_server_password	用于登录 IBM SPSS Analytic Server 的密码。
-analytic_server_epassword	用于登录 IBM SPSS Analytic Server 的经过编码的密码。
-analytic_server_credential	用于登录 IBM SPSS Analytic Server 的凭证。

组合多个参数

通过使用后跟文件名的 @ 符号，可以在调用时指定的单个命令文件中组合使用多个自变量。这使您能够缩短命令行调用，并且可以克服操作系统在命令长度方面的限制。例如，以下启动命令使用了 <commandFileName > 的引用文件中的指定参数。

```
modelerclient @<commandFileName>
```

如果需用空格，那么请用引号将命令文件的文件名和路径括起来，如下所示：

```
modelerclient @ "C:\Program Files\IBM\SPSS\Modeler\nn\scripts\my_command_file.txt"
```

命令文件可以包含先前在启动时逐个指定的所有自变量，并且每个自变量各占一行。例如：

```
-stream report.str
-Porder.full_filename=APR_orders.dat
-Preport.filename=APR_report.txt
-execute
```

当写入和引用命令文件时，必须遵循以下限制：

- 每条命令各占一行。
- 不要在命令文件中嵌入 `@CommandFile` 参数。

第 7 章 属性参考信息

属性参考信息概述

可以为节点、流、工程和超节点指定多个不同的属性。某些属性在所有节点中通用，例如"名称"、"注释"和"工具提示"，有些属性则只针对某些特定的节点类型。其他属性汲及高级流操作，例如高速缓存或"超节点"行为。可通过标准用户界面（例如当打开对话框编辑节点选项时）访问属性，还可以多种其他方式使用属性。

- 可通过脚本修改属性（如本章所述）。有关更多信息，请参阅『属性语法』。
- 可在"超节点"参数中应用节点属性。
- 启动 IBM SPSS Modeler 时，节点属性还可用作命令行选项（使用 -P 标记）的一部分。

在 IBM SPSS Modeler 的脚本编制上下文中，节点属性和流属性通常称为**槽参数**。在本指南中，它们指的是节点或流的属性。

有关脚本编写语言的详细信息，请参阅脚本编写语言。

属性语法

可以使用下列语法设置的属性

```
OBJECT.setPropertyValue(PROPERTY, VALUE)
```

或：

```
OBJECT.setKeyedPropertyValue(PROPERTY, KEY, VALUE)
```

可以使用下列语法检索的属性的值：

```
VARIABLE = OBJECT.getPropertyValue(PROPERTY)
```

或：

```
VARIABLE = OBJECT.getKeyedPropertyValue(PROPERTY, KEY)
```

其中 OBJECT 是节点或输出，PROPERTY 是表达式引用的节点属性的名称，而 KEY 是键控属性的键值。例如，以下语法用于查找过滤节点，然后设置缺省值以包含所有字段并根据下游数据过滤 Age 字段：

```
filternode = modeler.script.stream().findByType("filter", None)
filternode.setPropertyValue("default_include", True)
filternode.setKeyedPropertyValue("include", "Age", False)
```

可以使用流 findByType(TYPE, LABEL) 函数查找 IBM SPSS Modeler 中使用的所有节点。必须至少指定一个 TYPE 或 LABEL。

结构化属性

脚本编写通过结构化属性，增强语法解析清晰度的方式有二：

- 指定复杂节点属性名称的结构，例如"类型"节点、"过滤"节点或"均衡"节点。
- 提供一种可一次指定多种属性的格式。

复杂接口的结构化

带有表和其他复杂接口的节点（"类型"、"过滤"和"均衡"节点）的脚本必须遵循特定的结构以便正确解析。这些属性需要比单个标识的名称更为复杂的名称，此名称称为键。例如，在"过滤"节点内，每个可用字段（在其上游）均处于开或关的状态。要引用此信息，"过滤"节点将为每个字段（无论字段为 true 还是 false）存储一个信息项。此属性的值可能（或指定为）True 或 False。假如"过滤"节点 mynode（在其上游）有一名为 年龄 的字段。要关闭此字段，请将带有键 Age 的属性 include 的值设置为 False，如下所示：

```
mynode.setKeyedPropertyValue("include", "Age", False)
```

为设置多重属性而结构化

对于许多节点而言，您可以一次指定节点或流的多个属性。这称为**多重集合命令**或**设置块**。

在某些情况下，结构化属性相当复杂。示例如下：

```
sortnode.setPropertyValue("keys", [["K", "Descending"], ["Age", "Ascending"], ["Na", "Descending"]])
```

结构化属性的另一个优势在于，在某个节点稳定之前可以在该节点上设置若干个属性。缺省情况下，多重集合将在基于单个属性设置的操作运行之前，在块中设置所有属性。例如，如果在定义"固定文件"节点时分两步设置字段属性，由于节点在两个设置均生效之前是不一致的，将会导致发生错误。以多重集合方式定义属性，可使得在更新数据模型前设置上述两个属性，从而避免上述问题发生。

缩写

在节点属性语法中使用标准缩写。了解缩写有助于构建脚本。

表 *38.* 语法中使用的标准缩写

缩写	含义
abs	绝对值
len	长度
最小	最小值
最大	最大值
correl	相关
covar	协方差
num	数字或数值
pct	百分比
transp	透明度
xval	交叉验证
变量	方差或变量（源节点中）

节点和流属性示例

在 IBM SPSS Modeler 中可以各种方式使用节点和流属性。此类属性经常用作脚本的一部分，作为 **独立脚本** 的一部分用以实现多个流或操作的自动化；或用作 **流脚本** 的一部分用以实现单个流内部的过程自动化。还可通过在"超节点"内使用节点参数来指定节点参数。就最基础的水平而言，属性还可用作命令行选项来启动 IBM SPSS Modeler。将 -p 参数用作命令行调用的一部分时，可以使用流属性来更改流设置。

表 *39.* 节点和流属性示例

属性	含义
s.max_size	涉及到节点 s 的属性 max_size 。

表 *39.* 节点和流属性示例 *(续)*

属性	含义
s:samplenode.max_size	涉及到节点 s 的属性 max_size，其必须为样本节点。
:samplenode.max_size	涉及到当前流中样本节点的属性 max_size （只能有一个样本流）。
s:sample.max_size	涉及到节点 s 的属性 max_size，其必须为样本节点。
t.direction.Age	涉及到"类型"节点 t 中年龄字段的角色。
:.max_size	*** 非法操作 *** 必须指定节点名或节点类型。

示例 s:sample.max_size 说明不一定要写出节点类型的全称。

示例 t.direction.Age 说明，当某个节点属性比带有单个值的单个通道复杂时，某些通道名称将自行结构化。此类通道称为 **结构化** 或 **复杂** 属性。

节点属性概述

每种类型的节点都具有自己的一组合法属性，并且每个属性都具有类型。此类型可以是一般类型（数字、标志或字符串），在这种情况下，属性设置将强制转换为正确类型。如果无法进行强制转换，那么将发生错误。另外，通过属性引用可以指定合法值的范围，例如 Discard、PairAndDiscard 和 IncludeAsText，此时如果采用其他值，那么将出现错误。应通过采用值 true 或 false 来读取或设置标志属性。（设置值时也可识别如下变异值： Off、OFF、off、No、NO、no、n、N、f、F、false、False、FALSE 或 0，但在某些情况下读取属性值时会出错。所有其他值都将被视为 true。使用 true 和 false 时保持一致将可以避免混淆。）在本指南的参考表中，**属性描述**列对结构化属性进行了说明，并给出了属性的使用格式。

公共节点属性

IBM SPSS Modeler 中的很多属性通用于所有节点（包括超节点）。

表 *40.* 公共节点属性.

属性名称	数据类型	属性描述
use_custom_name	标志	
name	字符串	读取画布中某个节点名称的只读属性（自动或定制）。
custom_name	字符串	指定节点的定制名称。
tooltip	字符串	
annotation	字符串	
keywords	字符串	指定与对象关联的关键字列表的结构化通道（例如 ["Keyword1" "Keyword2"] ）。
cache_enabled	标志	
node_type	source_supernode process_supernode terminal_supernode 所有为进行脚本编制而指定的节点名	按类型引用节点的只读属性。例如，除按名称引用节点（例如 real_income ）之外，还可以指定类型（例如 userinputnode 或 filternode ）。

超节点属性以及所有其他节点的属性均将单独讨论。有关更多信息，请参阅第 331 页的第 21 章, 『超节点属性』主题。

第 8 章 流属性

通过脚本编写可以控制多种流属性。要引用流属性，您必须设置执行方法以使用脚本：

```
stream = modeler.script.stream()
stream.setPropertyValue("execute_method", "Script")
```

示例

节点属性用于引用当前流中的节点。如下所示的流脚本可作为一个示例：

```
stream = modeler.script.stream()
annotation = stream.getPropertyValue("annotation")

annotation = annotation + "\n\nThis stream is called \"" + stream.getLabel() + "\" and
 contains the following nodes:\n"

for node in stream.iterator():
    annotation = annotation + "\n" + node.getTypeName() + " node called \"" + node.getLabel()
 + "\""

stream.setPropertyValue("annotation", annotation)
```

上述示例使用节点属性创建了包含流中所有节点的列表，并将该列表写入流的注释中。生成的注解具有如下形式：

```
This stream is called "druglearn" and contains the following nodes:

type node called "Define Types"
derive node called "Na_to_K"
variablefile node called "DRUG1n"
neuralnetwork node called "Drug"
c50 node called "Drug"
filter node called "Discard Fields"
```

流属性的具体说明见于下表。

表 *41.* 流属性.

属性名称	数据类型	属性描述
execute_method	Normal Script	

表 *41.* 流属性 *(*续*).*

属性名称	数据类型	属性描述
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
date_baseline	*number*	
date_2digit_baseline	*number*	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	
time_rollover	标志	
import_datetime_as_string	标志	
decimal_places	*number*	
decimal_symbol	Default Period Comma	
angles_in_radians	标志	
use_max_set_size	标志	
max_set_size	*number*	

表 *41.* 流属性 *(续).*

属性名称	数据类型	属性描述
ruleset_evaluation	Voting FirstHit	
refresh_source_nodes	标志	用于在流执行中自动刷新源节点。
script	字符串	
annotation	字符串	
name	字符串	**注：**这是一个只读属性。如果想要更改流的名称，您应该使用其它名称加以保存。
parameters		使用此属性可以更新来自独立脚本中的流参数。
nodes		详细信息参见下方。
encoding	SystemDefault "UTF-8"	
stream_rewriting	*boolean*	
stream_rewriting_maximise_sql	*boolean*	
stream_rewriting_optimise_clem_执行	*boolean*	
stream_rewriting_optimise_syntax_执行	*boolean*	
enable_parallelism	*boolean*	
sql_generation	*boolean*	
database_caching	*boolean*	
sql_logging	*boolean*	
sql_generation_logging	*boolean*	
sql_log_native	*boolean*	
sql_log_prettyprint	*boolean*	
record_count_suppress_input	*boolean*	
record_count_feedback_interval	*integer*	
use_stream_auto_create_node_settings	布尔值	如果值为 true，那么将使用特定于流的设置，否则将使用用户首选项。
create_model_applier_for_new_models	布尔值	如果值为 true，那么在模型构建器创建新模型并且没有处于活动状态的更新链接时，将添加一个新的模型应用器。 **注：**如果您使用的是 IBM SPSS Modeler Batch V15，那么必须显式地在脚本中添加模型应用器。
create_model_applier_update_links	createEnabled createDisabled doNotCreate	定义自动添加模型应用器节点时创建的链接类型。
create_source_node_from_builders	布尔值	如果值为 true，那么在源构建器创建新的源输出并且没有处于活动状态的更新链接时，将添加一个新的源节点。

表 *41.* 流属性 *(*续*).*

属性名称	数据类型	属性描述
create_source_node_update_links	createEnabled createDisabled doNotCreate	定义自动添加源节点时创建的链接类型。
has_coordinate_system	布尔值	如果为 true，请对整个流应用坐标系。
coordinate_system	*string*	所选投影坐标系的名称。
deployment_area	ModelRefresh Scoring None	选择要如何部署流。如果该值设置为 None，那么不使用其他部署条目。
scoring_terminal_node_id	*string*	在流中选择评分分支。它可以是流中的任何终端节点。
scoring_node_id	*string*	选择评分分支中的模型块。
model_build_node_id	*string*	选择流中的建模节点。

第 9 章 源节点属性

源节点公共属性

所有源节点的公共属性如下所示，后面的主题是具体节点的相关信息。

示例 1

```
varfilenode = modeler.script.stream().create("variablefile", "Var. File")
varfilenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
varfilenode.setKeyedPropertyValue("check", "Age", "None")
varfilenode.setKeyedPropertyValue("values", "Age", [1, 100])
varfilenode.setKeyedPropertyValue("type", "Age", "Range")
varfilenode.setKeyedPropertyValue("direction", "Age", "Input")
```

示例 2

此脚本假定指定的数据文件包含表示多行字符串的字段 Region。

```
from modeler.api import StorageType
from modeler.api import MeasureType

# Create a Variable File node that reads the data set containing
# the "Region" field
varfilenode = modeler.script.stream().create("variablefile", "My Geo Data")
varfilenode.setPropertyValue("full_filename", "C:/mydata/mygeodata.csv")
varfilenode.setPropertyValue("treat_square_brackets_as_lists", True)

# Override the storage type to be a list...
varfilenode.setKeyedPropertyValue("custom_storage_type", "Region", StorageType.LIST)
# ...and specify the type if values in the list and the list depth
varfilenode.setKeyedPropertyValue("custom_list_storage_type", "Region", StorageType.INTEGER)
varfilenode.setKeyedPropertyValue("custom_list_depth", "Region", 2)

# Now change the measurement to indentify the field as a geospatial value...
varfilenode.setKeyedPropertyValue("measure_type", "Region", MeasureType.GEOSPATIAL)
# ...and finally specify the necessary information about the specific
# type of geospatial object
varfilenode.setKeyedPropertyValue("geo_type", "Region", "MultiLineString")
varfilenode.setKeyedPropertyValue("geo_coordinates", "Region", "2D")
varfilenode.setKeyedPropertyValue("has_coordinate_system", "Region", True)
varfilenode.setKeyedPropertyValue("coordinate_system", "Region",
 "ETRS_1989_EPSG_Arctic_zone_5-47")
```

表 *42.* 源节点公共属性.

属性名称	数据类型	属性描述
direction	Input Target Both None Partition Split Frequency RecordID	字段角色的键控属性。 用法格式： NODE.direction.FIELDNAME **注：** 现在，不推荐使用值 In 和 Out。在将来的版本中可能取消对这些值的支持。

表 *42.* 源节点公共属性 *(*续*).*

属性名称	数据类型	属性描述
type	范围 (Range) Flag Set Typeless Discrete Ordered Set Default	字段类型。如果将该属性设置为 *Default*，那么将清除所有 values 属性设置，如果将 value_mode 设置为 *Specify*，那么它将重新设置为 *Read*。如果 value_mode 已设置为 *Pass* 或 *Read*，那么它将不受 type 设置的影响。 用法格式： NODE.type.FIELDNAME
storage	未知 字符串 整数 实数 Time 日期 Timestamp	字段存储类型的只读键控属性。 用法格式： NODE.storage.FIELDNAME
check	None Nullify Coerce Discard Warn Abort	字段类型和范围检查的键控属性。 用法格式： NODE.check.FIELDNAME
values	[*value value*]	对于连续型（范围）字段而言，第一个是最小值，后一个是最大值。对于名义（集合）字段，请指定所有值。对标志字段而言，第一个值代表 *false*，后一个值代表 *true*。设置该属性将自动把 value_mode 属性设置为 *Specify*。存储器是根据列表中的第一个值确定的，例如，如果第一个值为 *string*，那么存储器将设置为 String。 用法格式： NODE.values.FIELDNAME
value_mode	Read Pass Read+ Current Specify	确定下一次数据传递中设置某个字段值的方式。 用法格式： NODE.value_mode.FIELDNAME 注意，不能将此属性直接设置为 *Specify*；要使用特定值，需设置 values 属性。
default_value_mode	Read Pass	指定用缺省方式设置所有字段值。 用法格式： NODE.default_value_mode 该设置可以通过使用 value_mode 属性，用特定字段进行覆盖。
extend_values	标志	当 value_mode 设置为 *Read* 时将应用。设为 *T* 则将新读取的值添加到任意现有字段值。设置为 *F* 则丢弃现有值并添加新读取值。 用法格式： NODE.extend_values.FIELDNAME
value_labels	字符串	用于指定值标签。请注意，必须先指定值。
enable_missing	标志	当设置为 *T* 时，那么激活对字段缺失值的跟踪。 用法格式： NODE.enable_missing.FIELDNAME

表 *42.* 源节点公共属性 *(续).*

属性名称	数据类型	属性描述
missing_values	[*value value ...*]	指定表示缺失数据的数据值。 用法格式： NODE.missing_values.FIELDNAME
range_missing	标志	此属性设置为 *T* 时，指定是否为字段定义缺失值（空白）范围。 用法格式： NODE.range_missing.FIELDNAME
missing_lower	字符串	range_missing 为 true 时，此属性指定缺失值范围的下限。 用法格式： NODE.missing_lower.FIELDNAME
missing_upper	字符串	range_missing 为 true 时，此属性指定缺失值范围的上限。 用法格式： NODE.missing_upper.FIELDNAME
null_missing	标志	当此属性设置为 *T* 时，将用空（在本软件中显示为 $null$ 的未定义值）表示缺失值。 用法格式： NODE.null_missing.FIELDNAME
whitespace_missing	标志	当该属性设置为 *T* 时，仅包含空白（空格、制表符和换行符）的值将被当成缺失值。 用法格式： NODE.whitespace_missing.FIELDNAME
description	字符串	用于指定字段标签或描述。
default_include	标志	用于指定默认行为是传递还是过滤字段的键控属性： NODE.default_include 示例： set mynode:filternode.default_include = false
include	标志	用于指出是包含还是过滤单个字段的键控属性： NODE.include.FIELDNAME.
new_name	*string*	
measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	此键控属性类似于 type，因为它可用于定义与字段关联的测量。区别在于，还可以向 Python 脚本编制中的 setter 函数传递其中一个 MeasureType 值，而 getter 始终返回 MeasureType 值。

表 42. 源节点公共属性 (续).

属性名称	数据类型	属性描述
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	对于收集字段（深度为 0 的列表），此键控属性定义与基础值关联的测量类型。
geo_type	Point MultiPoint LineString MultiLineString 多边形 MultiPolygon	对于地理空间字段，此键控属性定义该字段表示的地理空间对象的类型。这应该与值的列表深度保持一致。
has_coordinate_system	布尔值	对于地理空间字段，此属性定义该字段是否具有坐标系
coordinate_system	*string*	对于地理空间字段，此键控属性定义该字段的坐标系。
custom_storage_type	Unknown / MeasureType.UNKNOWN String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP List / MeasureType.LIST	此键控属性类似于 custom_storage，因为它可用于定义字段的覆盖存储。区别在于，还可以向 Python 脚本编制中的 setter 函数传递其中一个 StorageType 值，而 getter 始终返回 StorageType 值。
custom_list_storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	对于列表字段，此键控属性指定基础值的存储类型。
custom_list_depth	*integer*	对于列表字段，此键控属性指定字段的深度
max_list_length	*integer*	仅可用于测量级别为地理空间或集合的数据。指定列表可包含的元素数量以设置列表的最大长度。
max_string_length	*integer*	仅可用于无类型数据，生成 SQL 以创建表时使用。输入数据中最大字符串的值；这将在表中生成一个大小足够包含该字符串的列。

asimport 属性

Analytic Server 源使您可以在 Hadoop 分布式文件系统 (HDFS) 上运行流。

示例

```
node.setPropertyValue("use_default_as", False)
node.setPropertyValue("connection",
["false","9.119.141.141","9080","analyticserver","ibm","admin","admin","false","","","",""])
```

表 *43. asimport* 属性.

asimport 属性	**数据类型**	**属性说明**
data_source	*string*	数据源名称。
use_default_as	*boolean*	如果设置为 True，请使用服务器 options.cfg 文件中配置的缺省 Analytic Server 连接。如果设置为 False，请使用此节点的连接。
connection	["string","string","string", "string","string","string","string", "string","string","string", "string" ,"string"]	这是包含 Analytic Server 连接详细信息的列表属性。格式为：["is_secure_connect", "server_url", "server_port", "context_root", "consumer", "user_name", "password", "use-kerberos-auth", "kerberos-krb5-config-file-path", "kerberos-jaas-config-file-path", "kerberos-krb5-service-principal-name", "enable-kerberos-debug"]，其中 is_secure_connect: 指示是否使用安全连接，其值为 true 或 false。use-kerberos-auth: 指示是否使用 Kerberos 认证，其值为 true 或 false。enable-kerberos-debug: 指示是否使用 Kerberos 认证的调试模式，其值为 true 或 false。

cognosimport 节点属性

IBM Cognos 源节点从 Cognos Analytics 数据库导入数据。

示例

```
node = stream.create("cognosimport", "My node")
node.setPropertyValue("cognos_connection", ["http://mycogsrv1:9300/p2pd/servlet/dispatch",
 True, "", "", ""])
node.setPropertyValue("cognos_package_name", "/Public Folders/GOSALES")
node.setPropertyValue("cognos_items", ["[GreatOutdoors].[BRANCH].[BRANCH_CODE]",
"[GreatOutdoors].[BRANCH].[COUNTRY_CODE]"])
```

表 *44. cognosimport* 节点属性.

cognosimport 节点属性	**数据类型**	**属性描述**
mode	Data 报告	指定是导入 Cognos 数据（缺省）还是报告。
cognos_connection	*["string",flag,"string", "string" ,"string"]*	这是包含 Cognos 服务器连接详细信息的列表属性。格式为：["Cognos_server_URL", login_mode, "namespace", "username", "password"] 其中： Cognos_server_URL 是包含源的 Cognos 服务器的 URL。 login_mode 指示是否使用匿名登录，其值为 true 或 false；如果设置为 true，那么应将下列字段设置为 ""。 namespace 指定用于登录服务器的安全认证提供程序。 username 和 password 为用于登录 Cognos 服务器的用户名和密码。 作为替代 login_mode 的选项，还可以使用以下方式： • anonymousMode。例如：['Cognos_server_url', 'anonymousMode', "namespace", "username", "password"] • credentialMode。例如：['Cognos_server_url', 'credentialMode', "namespace", "username", "password"] • storedCredentialMode。例如：['Cognos_server_url', 'storedCredentialMode', "stored_credential_name"] 其中 stored_credential_name 是存储库中 Cognos 凭证的名称。
cognos_package_name	*string*	您要将数据对象导入其中的 Cognos 数据包的路径和名称，例如： /Public Folders/GOSALES **注：**只有正斜杠有效。
cognos_items	*["field","field", ... ,"field"]*	要导入的一个或多个数据对象的名称。字段格式为 [*namespace*].[*query_subject*].[*query_item*]
cognos_filters	字段	导入数据前要应用的一个或多个过滤器的名称。
cognos_data_parameters	列表	数据的提示参数的值。"名称/值"对括在花括号内，并且多个对以逗号分隔，而整个字符串括在方括号内。 格式： [["*param1*", "*value*"],...,["*paramN*", "*value*"]]
cognos_report_directory	字段	要从中导入报告的文件夹或包的 Cognos 路径，例如： /Public Folders/GOSALES **注：**只有正斜杠有效。

表 44. *cognosimport* 节点属性 *(续)*.

cognosimport 节点属性	**数据类型**	**属性描述**
cognos_report_name	字段	要导入的报告的报告位置中的路径和名称。
cognos_report_parameters	列表	报告参数的值。"名称/值"对括在花括号内，并且多个对以逗号分隔，而整个字符串括在方括号内。 格式： [["*param1*", "*value*"],...,["*paramN*", "*value*"]]

databasenode 属性

"数据库"节点可用于通过 ODBC（开放式数据库连接）从各种其他数据包（包括 Microsoft SQL Server、Db2 和 Oracle 等等）中导入数据。

示例

```
import modeler.api
stream = modeler.script.stream()
nnode = stream.create("database", "My node")
node.setPropertyValue("mode", "Table")
node.setPropertyValue("query", "SELECT * FROM drug1n")
node.setPropertyValue("datasource", "Drug1n_db")
node.setPropertyValue("username", "spss")
node.setPropertyValue("password", "spss")
node.setPropertyValue("tablename", ".Drug1n")
```

表 45. *databasenode* 属性.

databasenode 属性	**数据类型**	**属性描述**
mode	Table Query	借助对话框控件，指定将 *Table* 连接到数据库表，或借助 SQL 来指定用 *Query* 查询选定数据库。
datasource	字符串	数据库名称（另请参阅下面的注释）。
username	字符串	数据库连接详细信息（另请参阅下面的注释）。
password	字符串	
credential	*string*	IBM SPSS Collaboration and Deployment Services 中存储的凭证的名称。可以使用此属性来代替 username 和 password 属性。凭证的用户名和密码必须与访问数据库所需的用户名和密码相匹配
use_credential		设置为 True 或 False。
epassword	字符串	指定经过编码的密码，以代替在脚本中硬编码密码。 有关更多信息，请参阅第 45 页的『生成加密密码』主题。在执行期间，此属性为只读。
tablename	字符串	要访问的表名称。

表 45. *databasenode* 属性 *(续)*.

databasenode 属性	数据类型	属性描述
strip_spaces	None Left Right Both	丢弃字符串中前端和尾部空格的选项。
use_quotes	AsNeeded Always Never	指定向数据库发送查询时，是否将表名和列名括在引号内（例如，如果这些名称包含空格或标点）。
query	字符串	指定要提交查询所对应的 SQL 编码。

注：如果 datasource 属性中的数据库名称包含一个或多个空格、句点（也称为"句号"）或下划线，那么您可以使用"反斜杠双引号"格式将其作为字符串进行处理。例如：`"{\"db2v9.7.6_linux\"}"` 或 `"{\"TDATA 131\"}"`。 此外，始终将 datasource 字符串值用双引号和花括号围起来，如下例所示：`"{\"SQL Server\",spssuser,abcd1234,false}"`。

注：如果 datasource 属性中的数据库名称包含空格，那么您还可以具有下列格式的单个数据源属性来代替 datasource、username 和 password 等各个属性：

表 46. *databasenode* 属性 - 特定于数据源.

databasenode 属性	数据类型	属性描述
datasource	字符串	格式： `[database_name,username,password[,true | false]]` 最后一个参数与经过加密的密码配合使用。如果将其设为 true，将会在使用之前对密码进行解密。

如果您要更改数据源，也可使用此格式；不过，如果您只想更改用户名或密码，那么可使用 username 或 password 属性。

datacollectionimportnode 属性

Data Collection 数据导入节点根据市场调查产品使用的 Data Collection 数据模型导入调查数据。必须安装 Data Collection 数据库才可使用此节点。

示例

```
node = stream.create("datacollectionimport", "My node")
node.setPropertyValue("metadata_name", "mrQvDsc")
node.setPropertyValue("metadata_file", "C:/Program Files/IBM/SPSS/DataCollection/DDL/Data/
Quanvert/Museum/museum.pkd")
node.setPropertyValue("casedata_name", "mrQvDsc")
node.setPropertyValue("casedata_source_type", "File")
node.setPropertyValue("casedata_file", "C:/Program Files/IBM/SPSS/DataCollection/DDL/Data/
```

```
Quanvert/Museum/museum.pkd")
node.setPropertyValue("import_system_variables", "Common")
node.setPropertyValue("import_multi_response", "MultipleFlags")
```

表 47. *datacollectionimportnode* 属性.

datacollectionimportnode 属性	**数据类型**	**属性描述**
metadata_name	字符串	MDSC 的名称。特殊值 DimensionsMDD 表示应使用标准 Data Collection 元数据文档。其他可能的值包括： mrADODsc mrI2dDsc mrLogDsc mrQdiDrsDsc mrQvDsc mrSampleReportingMDSC mrSavDsc mrSCDsc mrScriptMDSC 特殊值 none 指示不存在 MDSC。
metadata_file	字符串	存储元数据的文件的名称。
casedata_name	字符串	CDSC 的名称。可能的值包括： mrADODsc mrI2dDsc mrLogDsc mrPunchDSC mrQdiDrsDsc mrQvDsc mrRdbDsc2 mrSavDsc mrScDSC mrXmlDsc 特殊值 none 指示不存在 CDSC。
casedata_source_type	未知 File 文件夹 UDL DSN	指出 CDSC 的源类型。
casedata_file	字符串	当 casedata_source_type 为 *File* 时，那么指定包含案例数据的文件。
casedata_folder	字符串	当 casedata_source_type 为 *Foder* 时，那么指定包含案例数据的文件夹。
casedata_udl_string	字符串	当 casedata_source_type 为 *UDL* 时，那么为包含案例数据的数据源指定 OLD-DB 连接字符串。
casedata_dsn_string	字符串	当 casedata_source_type 为 *DSN*，那么为数据源指定 ODBC 连接字符串。
casedata_project	字符串	从 Data Collection 数据库中读取观测值数据时，可以输入工程的名称。对于所有其他的观测值数据类型，应将此设置留空。

表 47. *datacollectionimportnode* 属性 (续).

datacollectionimportnode 属性	数据类型	属性描述
version_import_mode	All Latest Specify	定义版本处理方式。
specific_version	字符串	当 version_import_mode 为 *Specify* 时，那么定义要导入案例数据的版本。
use_language	字符串	定义是否应使用指定语言的标签。
language	字符串	如果 use_language 的值为 True，那么定义导入时要使用的语言代码。语言代码应为案例数据中的某一可用代码。
use_context	字符串	定义是否应导入特定的上下文。上下文用于区分与响应相关的描述。
context	字符串	如果 use_context 的值为 true，那么定义导入环境。环境应是案例数据中的某一可用环境。
use_label_type	字符串	定义是否应导入指定标签类型。
label_type	字符串	如果 use_label_type 的值为 true，那么定义要导入的标签类型。标签类型应是案例数据中的某一可用标签类型。
user_id	字符串	对于要求显式登录的数据库，您可以提供用于访问数据源的用户标识和密码。
password	字符串	
import_system_variables	常用 None All	指定要导入的系统变量。
import_codes_variables	标志	
import_sourcefile_variables	标志	
import_multi_response	MultipleFlags Single	

excelimportnode 属性

Excel 导入节点可从 Microsoft Excel 以 .xlsx 文件格式导入数据。 不需要指定 ODBC 数据源。

示例

```
#To use a named range:node = stream.create("excelimport", "My node")
node.setPropertyValue("excel_file_type", "Excel2007")
node.setPropertyValue("full_filename", "C:/drug.xlsx")
node.setPropertyValue("use_named_range", True)
node.setPropertyValue("named_range", "DRUG")
node.setPropertyValue("read_field_names", True)

#To use an explicit range:node = stream.create("excelimport", "My node")
```

```
node.setPropertyValue("excel_file_type", "Excel2007")
node.setPropertyValue("full_filename", "C:/drug.xlsx")
node.setPropertyValue("worksheet_mode", "Name")
node.setPropertyValue("worksheet_name", "Drug")
node.setPropertyValue("explicit_range_start", "A1")
node.setPropertyValue("explicit_range_end", "F300")
```

表 *48. excelimportnode* 属性.

excelimportnode 属性	**数据类型**	**属性描述**
excel_file_type	Excel2007	
full_filename	字符串	完整文件名（包括路径）。
use_named_range	布尔值	是否使用指定的范围。如果为 true，那么将用 named_range 属性来指定读取范围，但忽略其他工作表和数据范围设置。
named_range	字符串	
worksheet_mode	Index Name	指定是否通过索引或名称来定义工作表。
worksheet_index	*integer*	要读取工作表的索引，开始时第一个工作表的索引为 0，第二个工作表的索引为 1，依此类推。
worksheet_name	字符串	要读取工作表的名称。
data_range_mode	FirstNonBlank ExplicitRange	指定确定范围的方式。
blank_rows	StopReading ReturnBlankRows	当 data_range_mode 为 *FirstNonBlank* 时，指定空行处理方式。
explicit_range_start	字符串	当 data_range_mode 为 *ExplicitRange* 时，指定要读取范围的起点。
explicit_range_end	字符串	
read_field_names	布尔值	指定是否应将指定范围的第一行用作字段（列）名称。

extensionimportnode 属性

通过扩展导入节点，您可以运行 R 或 Python for Spark 脚本来导入数据。

Python for Spark 示例

```
##### Script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_importer", "extension_importer")
node.setPropertyValue("syntax_type", "Python")

python_script = """
import spss.pyspark
from pyspark.sql.types import *
```

```
cxt = spss.pyspark.runtime.getContext()

_schema = StructType([StructField('id', LongType(), nullable=False), \
StructField('age', LongType(), nullable=True), \
StructField('Sex', StringType(), nullable=True), \
StructField('BP', StringType(), nullable=True), \
StructField('Cholesterol', StringType(), nullable=True), \
StructField('K', DoubleType(), nullable=True), \
StructField('Na', DoubleType(), nullable=True), \
StructField('Drug', StringType(), nullable=True)])

if cxt.isComputeDataModelOnly():
 cxt.setSparkOutputSchema(_schema)
else:
 df = cxt.getSparkInputData()
 if df is None:
  drugList=[(1,23,'F','HIGH','HIGH',0.792535,0.031258,'drugY'), \
(2,47,'M','LOW','HIGH',0.739309,0.056468,'drugC'),\
               (3,47,'M','LOW','HIGH',0.697269,0.068944,'drugC'),\
               (4,28,'F','NORMAL','HIGH',0.563682,0.072289,'drugX'),\
               (5,61,'F','LOW','HIGH',0.559294,0.030998,'drugY'),\
               (6,22,'F','NORMAL','HIGH',0.676901,0.078647,'drugX'),\
               (7,49,'F','NORMAL','HIGH',0.789637,0.048518,'drugY'),\
               (8,41,'M','LOW','HIGH',0.766635,0.069461,'drugC'),\
               (9,60,'M','NORMAL','HIGH',0.777205,0.05123,'drugY'),\
               (10,43,'M','LOW','NORMAL',0.526102,0.027164,'drugY')]
  sqlcxt = cxt.getSparkSQLContext()
  rdd = cxt.getSparkContext().parallelize(drugList)
  print 'pyspark read data count = '+str(rdd.count())
  df = sqlcxt.createDataFrame(rdd, _schema)

 cxt.setSparkOutputData(df)
"""

node.setPropertyValue("python_syntax", python_script)
```

R 示例

```
#### Script example for R
node.setPropertyValue("syntax_type", "R")

R_script = """# 'JSON Import' Node v1.0 for IBM SPSS Modeler
# 'RJSONIO' package created by Duncan Temple Lang - http://cran.r-project.org/web/packages/RJSONIO
# 'plyr' package created by Hadley Wickham http://cran.r-project.org/web/packages/plyr
# Node developer: Danil Savine - IBM Extreme Blue 2014
# Description: This node allows you to import into SPSS a table data from  a JSON.
# Install function for packages
packages <- function(x){
  x <- as.character(match.call()[[2]])
  if (!require(x,character.only=TRUE)){
    install.packages(pkgs=x,repos="http://cran.r-project.org")
    require(x,character.only=TRUE)
  }
}
#  packages
packages(RJSONIO)
packages(plyr)
### This function is used to generate automatically the dataModel
getMetaData <- function (data) {
  if (dim(data)[1]<=0) {

    print("Warning : modelerData has no line, all fieldStorage fields set to strings")
    getStorage <- function(x){return("string")}

  } else {
```

```
    getStorage <- function(x) {
      res <- NULL
      #if x is a factor, typeof will return an integer so we treat the case on the side
      if(is.factor(x)) {
        res <- "string"
      } else {
        res <- switch(typeof(unlist(x)),
                      integer = "integer",
                      double = "real",
                      character = "string",
                      "string")
      }
      return (res)
    }
  }

  col = vector("list", dim(data)[2])
  for (i in 1:dim(data)[2]) {
    col[[i]] <- c(fieldName=names(data[i]),
                  fieldLabel="",
                  fieldStorage=getStorage(data[i]),
                  fieldMeasure="",
                  fieldFormat="",
                  fieldRole="")
  }
  mdm<-do.call(cbind,col)
  mdm<-data.frame(mdm)
  return(mdm)
}
# From JSON to a list
txt <- readLines('C:/test.json')
formatedtxt <- paste(txt, collapse = '')
json.list <- fromJSON(formatedtxt)
  # Apply path to json.list
  if(strsplit(x='true', split='
  ' ,fixed=TRUE)[[1]][1]) {
    path.list <- unlist(strsplit(x='id_array', split=','))
    i = 1
    while(i<length(path.list)+1){
      if(is.null(getElement(json.list, path.list[i]))){
         json.list <- json.list[[1]]
      }else{
      json.list <- getElement(json.list, path.list[i])
      i <- i+1
      }
    }
  }
# From list to dataframe via unlisted json
i <-1
filled <- data.frame()
while(i < length(json.list)+ 1){
  unlisted.json <- unlist(json.list[[i]])
  to.fill <- data.frame(t(as.data.frame(unlisted.json, row.names = names(unlisted.json))), stringsAsFactors=FALSE)
  filled <- rbind.fill(filled,to.fill)
  i <-  1 + i
}
# Export to SPSS Modeler Data
modelerData <- filled
print(modelerData)
modelerDataModel <- getMetaData(modelerData)
print(modelerDataModel)

"""

node.setPropertyValue("r_syntax", R_script)
```

表 49. *extensionimportnode* 属性

extensionimportnode 属性	**数据类型**	**属性描述**
syntax_type	*R* *Python*	指定要运行的脚本 - R 或 Python（R 为缺省值）
r_syntax	*string*	要运行的 R 脚本编制语法。
python_syntax	*string*	要运行的 Python 脚本编制语法。

fixedfilenode 属性

"固定文件"节点从固定字段文本文件（即，文件中的字段不进行定界，而是从同一位置开始且长度固定）中导入数据。机器生成的数据或遗存数据经常以固定字段格式存储。

示例

```
node = stream.create("fixedfile", "My node")
node.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node.setPropertyValue("record_len", 32)
node.setPropertyValue("skip_header", 1)
node.setPropertyValue("fields", [["Age", 1, 3], ["Sex", 5, 7], ["BP", 9, 10], ["Cholesterol",
 12, 22], ["Na", 24, 25], ["K", 27, 27], ["Drug", 29, 32]])
node.setPropertyValue("decimal_symbol", "Period")
node.setPropertyValue("lines_to_scan", 30)
```

表 50. *fixedfilenode* 属性.

fixedfilenode 属性	**数据类型**	**属性描述**
record_len	*number*	指定每条记录中的字符数。
line_oriented	标志	跳过每条记录尾部的换行符。
decimal_symbol	Default Comma Period	用于数据源中的十进制分隔符的类型。
skip_header	*number*	指定要在每条记录开头忽略的行数。用于忽略列标题。
auto_recognize_datetime	标志	指定在源数据中是否自动标识日期或时间。
lines_to_scan	*number*	
fields	列表	结构化属性。
full_filename	字符串	要读取文件的全称（包括目录）。
strip_spaces	None Left Right Both	在导入时丢弃字符串中前端和尾部的空格。
invalid_char_mode	Discard Replace	从数据输入中除去无效字符（空值、0 或当前编码中所没有的字符），或用指定的单字符符号替换无效字符。
invalid_char_replacement	字符串	
use_custom_values	标志	

表 *50. fixedfilenode* 属性 *(续).*

fixedfilenode 属性	**数据类型**	**属性描述**
custom_storage	未知 字符串 整数 实数 Time 日期 Timestamp	
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	只有在指定了定制存储的情况下，此属性才适用。

表 50. *fixedfilenode* 属性 (续).

fixedfilenode 属性	数据类型	属性描述
custom_time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	只有在指定了定制存储的情况下，此属性才适用。
custom_decimal_symbol	字段	只有在指定了定制存储的情况下适用。
encoding	StreamDefault SystemDefault "UTF-8"	指定文本编码方法。

gsdata_import 节点属性

使用"地理空间"源节点可以将图或空间数据引入数据挖掘会话中。

表 51. *gsdata_import* 节点属性

gsdata_import 节点属性	数据类型	属性描述
full_filename	*string*	输入您要装入的 .shp 文件的文件路径。
map_service_URL	*string*	输入要连接到的映射服务 URL。
map_name	*string*	仅当使用 map_service_URL 时；此类型包含映射服务的顶级文件夹结构。

sasimportnode 属性

SAS 导入节点可将 SAS 数据导入到 IBM SPSS Modeler 中。

示例

```
node = stream.create("sasimport", "My node")
node.setPropertyValue("format", "Windows")
node.setPropertyValue("full_filename", "C:/data/retail.sas7bdat")
node.setPropertyValue("member_name", "Test")
node.setPropertyValue("read_formats", False)
node.setPropertyValue("full_format_filename", "Test")
node.setPropertyValue("import_names", True)
```

表 *52. sasimportnode* 属性.

sasimportnode 属性	**数据类型**	**属性描述**
format	Windows UNIX Transport SAS7 SAS8 SAS9	要导入文件的格式。
full_filename	字符串	输入的完整文件名（包括路径）。
member_name	字符串	指定要从特定 SAS 传输文件中导入的成员。
read_formats	标志	从指定格式文件中读取数据格式（例如变量标签）。
full_format_filename	字符串	
import_names	NamesAndLabels LabelsasNames	指定在导入时映射变量名称和标签的方法。

simgennode 属性

"模拟生成"节点提供了一种生成模拟数据的简单方法 - 使用用户指定的统计分布从头开始生成数据，或者使用对现有历史数据运行"模拟拟合"节点而获取的分布自动生成数据。对于模型输入中存在不确定性的情况，此节点在对预测模型的结果进行评估时非常有用。

表 *53. simgennode* 属性.

simgennode 属性	**数据类型**	**属性描述**
fields	结构化属性	请参阅示例
correlations	结构化属性	请参阅示例
keep_min_max_setting	布尔值	
refit_correlations	布尔值	
max_cases	*integer*	最小值为 1000，最大值为 2,147,483,647
create_iteration_field	布尔值	
iteration_field_name	*string*	
replicate_results	布尔值	
random_seed	*integer*	
parameter_xml	*string*	以字符串形式返回参数 XML

fields 示例

以下是使用下列语法的结构化槽参数：

```
simgennode.setPropertyValue("fields", [
    [field1, storage, locked, [distribution1], min, max],
    [field2, storage, locked, [distribution2], min, max],
    [field3, storage, locked, [distribution3], min, max]
])
```

distribution 是分布名称的声明，此名称后跟包含属性名称/值对的列表。每个分布以下列方式定义：

```
[distributionname, [[par1], [par2], [par3]]]
```

```
simgennode = modeler.script.stream().createAt("simgen", u"Sim Gen", 726, 322)
simgennode.setPropertyValue("fields", [["Age", "integer", False, ["Uniform",[["min","1"],["max","2"]]], "", ""]])
```

例如，要创建用于生成具有二项分布的单个字段的节点，您可以使用以下脚本：

```
simgen_node1 = modeler.script.stream().createAt("simgen", u"Sim Gen", 200, 200)
simgen_node1.setPropertyValue("fields", [["Education", "Real", False, ["Binomial", [["n", 32],
 ["prob", 0.7]]], "", ""]])
```

二项分布使用两个参数：n 和 prob。由于二项分布不支持最小值和最大值，因此这两个参数将作为空字符串提供。

注：您不能直接设置 distribution；可以将其与 fields 属性一起使用。

以下示例显示所有可能的分发类型。 请注意，阈值在 NegativeBinomialFailures 和 NegativeBinomialTrial 中均输入为 thresh。

```
stream = modeler.script.stream()

simgennode = stream.createAt("simgen", u"Sim Gen", 200, 200)

beta_dist = ["Field1", "Real", False, ["Beta",[["shape1","1"],["shape2","2"]]], "", ""]
binomial_dist = ["Field2", "Real", False, ["Binomial",[["n" ,"1"],["prob","1"]]], "", ""]
categorical_dist = ["Field3", "String", False, ["Categorical", [["A",0.3],["B",0.5],["C",0.2]]], "", ""]
dice_dist = ["Field4", "Real", False, ["Dice", [["1" ,"0.5"],["2","0.5"]]], "", ""]
exponential_dist = ["Field5", "Real", False, ["Exponential", [["scale","1"]]], "", ""]
fixed_dist = ["Field6", "Real", False, ["Fixed", [["value","1" ]]], "", ""]
gamma_dist = ["Field7", "Real", False, ["Gamma", [["scale","1"],["shape"," 1"]]], "", ""]
lognormal_dist = ["Field8", "Real", False, ["Lognormal", [["a","1"],["b","1" ]]], "", ""]
negbinomialfailures_dist = ["Field9", "Real", False, ["NegativeBinomialFailures",[["prob","0.5"],["thresh","1"]]], "", ""]
negbinomialtrial_dist = ["Field10", "Real", False, ["NegativeBinomialTrials",[["prob","0.2"],["thresh","1"]]], "", ""]
normal_dist = ["Field11", "Real", False, ["Normal", [["mean","1"] ,["stddev","2"]]], "", ""]
poisson_dist = ["Field12", "Real", False, ["Poisson", [["mean","1"]]], "", ""]
range_dist = ["Field13", "Real", False, ["Range", [["BEGIN","[1,3]"] ,["END","[2,4]"],["PROB","[[0.5],[0.5]]"]]], "", ""]
triangular_dist = ["Field14", "Real", False, ["Triangular", [["min","0"],["max","1"],["mode","1"]]], "", ""]
uniform_dist = ["Field15", "Real", False, ["Uniform", [["min","1"],["max","2"]]], "", ""]
weibull_dist = ["Field16", "Real", False, ["Weibull", [["a","0"],["b","1 "],["c","1"]]], "", ""]

simgennode.setPropertyValue("fields", [\
beta_dist, \
binomial_dist, \
categorical_dist, \
dice_dist, \
exponential_dist, \
fixed_dist, \
gamma_dist, \
lognormal_dist, \
negbinomialfailures_dist, \
negbinomialtrial_dist, \
normal_dist, \
poisson_dist, \
range_dist, \
triangular_dist, \
uniform_dist, \
weibull_dist
])
```

相关示例

以下是使用下列语法的结构化槽参数：

```
simgennode.setPropertyValue("correlations", [
    [field1, field2, correlation],
    [field1, field3, correlation],
    [field2, field3, correlation]
])
```

相关性可以是介于 +1 与 -1 之间的任何数字。您可以根据需要指定相关性。任何未指定的相关性都将设置为 0。如果存在任何未知字段，那么应该在相关性矩阵（或表）上设置相关性值，并以红色文本显示该值。如果存在未知字段，那么无法执行节点。

statisticsimportnode 属性

IBM SPSS Statistics 文件节点从 IBM SPSS Statistics 使用的 *.sav* 文件格式以及保存在 IBM SPSS Modeler 中的高速缓存文件（其也使用相同格式）读取数据。

有关此节点属性的信息，请参阅第 315 页的『statisticsimportnode 属性』。

tm1odataimport 节点属性

IBM Cognos TM1 源节点从 Cognos TM1 数据库导入数据。

表 *54. tm1odataimport* 节点属性

tm1odataimport 节点属性	**数据类型**	**属性描述**
admin_host	*string*	REST API 的主机名的 URL。
server_name	*string*	从 admin_host 中选择的 TM1 服务器的名称。
credential_type	*inputCredential* 或 *storedCredential*	用于指示凭证类型。
input_credential	列表	当 credential_type 为 inputCredential 时；指定域、用户名和密码。
stored_credential_name	*string*	当 credential_type 为 storedCredential 时；指定 C&DS 服务器上的凭证名称。
selected_view	*["field" "field"]*	列表属性，其中包含所选 TM1 多维数据集的详细信息以及要从中将数据导入到 SPSS 的多维数据集视图的名称。例如：TM1_import.setPropertyValue("selected_view", ['plan_BudgetPlan', 'Goal Input'])
is_private_view	*flag*	指定 selected_view 是否是专用视图。缺省值为 false。

表 54. *tm1odataimport* 节点属性（续）

tm1odataimport 节点属性	**数据类型**	**属性描述**
selected_columns	*["field"]*	指定所选的列；只能指定一个项。 例如：setPropertyValue("selected_columns", ["Measures"])
selected_rows	*["field" "field"]*	指定所选的行。 例如：setPropertyValue("selected_rows", ["Dimension_1_1", "Dimension_2_1", "Dimension_3_1", "Periods"])

tm1import 节点属性（不推荐）

IBM Cognos TM1 源节点从 Cognos TM1 数据库导入数据。

注：此节点在 Modeler 18.0 中不推荐。替换节点脚本名称为 *tm1odataimport*。

表 55. *tm1import* 节点属性.

tm1import 节点属性	**数据类型**	**属性描述**
pm_host	*string*	**注：**仅限 V16.0 和 17.0 主机名。例如：TM1_import.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')
tm1_connection	*["field","field", ... ,"field"]*	**注：**仅限 V16.0 和 17.0 包含 TM1 服务器的连接详细信息的列表属性。格式为：["TM1_Server_Name","tm1_ username","tm1_password"] 例如：TM1_import.setPropertyValue ("tm1_connection", ['Planning Sample', "admin", "apple"])
selected_view	*["field" "field"]*	列表属性，其中包含所选 TM1 多维数据集的详细信息以及要从中将数据导入到 SPSS 的多维数据集视图的名称。例如：TM1_import.setPropertyValue ("selected_view", ['plan_BudgetPlan', 'Goal Input'])
selected_column	*["field"]*	指定所选的列；只能指定一个项。 例如：setPropertyValue("selected_columns", ["Measures"])
selected_rows	*["field" "field"]*	指定所选的行。 例如：setPropertyValue("selected_rows", ["Dimension_1_1", "Dimension_2_1", "Dimension_3_1", "Periods"])

twcimport 节点属性

TWC 源节点会从 The Weather Company（一家 IBM 企业）导入天气数据。您可将其用于获取某个位置的历史天气数据或预测天气数据。这有助于您制定天气驱动的业务解决方案，用于使用最准确且最精确的可用天气数据来制定更好的决策。

表 *56. twcimport* 节点属性

twcimport 节点属性	**数据类型**	**属性描述**
TWCDataImport.latitude	实数	指定 [-90.0 90.0] 格式的纬度值
TWCDataImport.longitude	实数	指定 [-180.0 180.0] 格式的经度值。
TWCDataImport.licenseKey	*string*	指定从 The Weather Company 获取的许可证密钥。
TWCDataImport.measurmentUnit	English Metric Hybrid	指定度量单位。可能的值包括 English、Metric 或 Hybrid。Metric 为缺省值。
TWCDataImport.dataType	Historical Forecast	指定要输入的天气数据类型。可能的值包括 Historical 或 Forecast。 Historical 为缺省值。
TWCDataImport.startDate	整数	如果针对 TWCDataImport.dataType 指定了 Historical，请按 yyyyMMdd 格式指定开始日期。
TWCDataImport.endDate	整数	如果针对 TWCDataImport.dataType 指定了 Historical，请按 yyyyMMdd 格式指定结束日期。
TWCDataImport.forecastHour	6 12 24 48	如果针对 TWCDataImport.dataType 指定了 Forecast，请针对小时指定 6、12、24 或 48。

userinputnode 属性

用户输入节点提供了一种用于创建综合数据的简单方式 - 可以从头开始创建也可以通过更改现有数据进行创建。此节点非常有用，例如，在希望为建模创建测试数据集时，即可使用此节点。

示例

```
node = stream.create("userinput", "My node")
node.setPropertyValue("names", ["test1", "test2"])
node.setKeyedPropertyValue("data", "test1", "2, 4, 8")
node.setKeyedPropertyValue("custom_storage", "test1", "Integer")
node.setPropertyValue("data_mode", "Ordered")
```

表 *57. userinputnode* 属性.

userinputnode 属性	**数据类型**	**属性描述**
data		

表 57. *userinputnode* 属性 *(续)*.

userinputnode 属性	**数据类型**	**属性描述**
names		设置或返回节点所生成的字段名称列表的结构化通道。
custom_storage	未知 字符串 整数 实数 Time 日期 Timestamp	可用于设置或返回某个字段存储的通道。
data_mode	Combined Ordered	如果指定了 Combined，那么设定值以及最小/最大值的每个组合都将生成一个记录。生成的记录数等于每个字段中值的数量的乘积。如果指定了有序，那么从每条记录的每一列中提取一个值来生成数据行。生成的记录数等于一个与字段相关的最大数值。将为所有数据值较少的字段添加空值。
values		**注：**此属性已由 userinputnode.data 取代，因此不推荐继续使用。

variablefilenode 属性

"变量文件"节点读取自由格式字段文本文件中的数据，即，其记录包含固定数量的字段，但包含不定数量字符的文件。此节点对于具有固定长度标题文本和某些特定类型注解的文件也非常有用。

示例

```
node = stream.create("variablefile", "My node")
node.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node.setPropertyValue("read_field_names", True)
node.setPropertyValue("delimit_other", True)
node.setPropertyValue("other", ",")
node.setPropertyValue("quotes_1", "Discard")
node.setPropertyValue("decimal_symbol", "Comma")
node.setPropertyValue("invalid_char_mode", "Replace")
node.setPropertyValue("invalid_char_replacement", "|")
node.setKeyedPropertyValue("use_custom_values", "Age", True)
node.setKeyedPropertyValue("direction", "Age", "Input")
node.setKeyedPropertyValue("type", "Age", "Range")
node.setKeyedPropertyValue("values", "Age", [1, 100])
```

表 58. *variablefilenode* 属性.

variablefilenode 属性	**数据类型**	**属性描述**
skip_header	*number*	指定每条记录开头要忽略的字符数。
num_fields_auto	标志	自动确定每条记录中的字段数。记录必须以换行符终止。

表 *58. variablefilenode* 属性 *(续).*

variablefilenode 属性	**数据类型**	**属性描述**
num_fields	*number*	手动指定每条记录中的字段数。
delimit_space	标志	指定文件中用于划定字段边界的字符。
delimit_tab	标志	
delimit_new_line	标志	
delimit_non_printing	标志	
delimit_comma	标志	当逗点在流中同时用作十进制分隔符和字段定界符时，将 delimit_other 设置为 *true*，然后使用其他属性将逗点指定为定界符。
delimit_other	标志	允许使用其他属性来指定定制定界符。
other	字符串	在 delimit_other 为 *true* 时，指定要使用的定界符。
decimal_symbol	Default Comma Period	指定用于数据源中的十进制分隔符。
multi_blank	标志	将多个相邻空格定界符视为一个单一定界符处理。
read_field_names	标志	将数据文件的第一行作为列的标签。
strip_spaces	None Left Right Both	在导入时丢弃字符串中前端和尾部的空格。
invalid_char_mode	Discard Replace	从数据输入中除去无效字符（空值、0 或当前编码中所没有的字符），或用指定的单字符符号替换无效字符。
invalid_char_replacement	字符串	
break_case_by_newline	*flag*	指定行定界符为换行符。
lines_to_scan	*number*	指定具体数据类型的扫描行数。
auto_recognize_datetime	标志	指定在源数据中是否自动标识日期或时间。
quotes_1	Discard PairAndDiscard IncludeAsText	指定导入后单引号的处理方式。
quotes_2	Discard PairAndDiscard IncludeAsText	指定导入后双引号的处理方式。
full_filename	字符串	要读取的文件全称（包括目录）。
use_custom_values	标志	
custom_storage	未知 字符串 整数 实数 Time 日期 Timestamp	

表 *58. variablefilenode* 属性 *(续).*

variablefilenode 属性	**数据类型**	**属性描述**
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	只有在指定了定制存储的情况下适用。
custom_time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	只有在指定了定制存储的情况下适用。
custom_decimal_symbol	字段	只有在指定了定制存储的情况下适用。
encoding	StreamDefault SystemDefault "UTF-8"	指定文本编码方法。

xmlimportnode 属性

"XML 源"节点将 XML 格式的数据导入到流中。可以导入单个文件，也可以导入某个目录中的所有文件。您可以选择性地指定模式文件，以便从中读取 XML 结构。

示例

```
node = stream.create("xmlimport", "My node")
node.setPropertyValue("full_filename", "c:/import/ebooks.xml")
node.setPropertyValue("records", "/author/name")
```

表 59. *xmlimportnode* 属性.

xmlimportnode 属性	数据类型	属性描述
read	single directory	读取单个数据文件（缺省），或目录中的所有 XML 文件。
recurse	标志	指定是否另外读取指定目录的所有子目录中的 XML 文件。
full_filename	字符串	（必需）要导入的 XML 文件的完整路径和文件名（如果 read = single）。
directory_name	字符串	（必需）要从中导入 XML 文件的目录的完整路径和名称（如果 read = directory）。
full_schema_filename	字符串	要从中读取 XML 结构的 XSD 或 DTD 文件的完整路径和文件名。如果您省略了此参数，将从 XML 源文件中读取结构。
records	字符串	XPath 表达式（例如，/author/name），用以定义记录边界。每次在源文件中遇到此元素时，都将创建新的记录。
mode	read specify	读取所有数据（缺省），或指定要读取的项目。
fields		要导入的项目（元素和属性）列表。列表中的每项为 XPath 表达式。

dataviewimport 属性

"数据视图"节点将"数据视图"数据导入 IBM SPSS Modeler 中。

示例

```
stream = modeler.script.stream()

dvnode = stream.createAt("dataviewimport", "Data View", 96, 96)
dvnode.setPropertyValue("analytic_data_source",
["","/folder/adv", "LATEST"])
```

```
dvnode.setPropertyValue("table_name", ["","com.ibm.spss.Table"])
dvnode.setPropertyValue("data_access_plan",
["","DataAccessPlan"])
dvnode.setPropertyValue("optional_attributes",
[["","NewDerivedAttribute"]])
dvnode.setPropertyValue("include_xml", True)
dvnode.setPropertyValue("include_xml_field", "xml_data")
```

表 *60. dataviewimport* 属性

dataviewimport 属性	**数据类型**	**属性描述**
analytic_data_source	*string*	存储在 IBM SPSS Collaboration and Deployment Services 中的分析数据视图对象。要使用的版本的路径名和版本标签。 ["Object ID","Full path", "Version"]
table_name	*string*	分析数据视图中使用的数据视图表。表名必须是包限定表名。您可以通过从 IBM SPSS Collaboration and Deployment Services Deployment Manager 客户机中导出 BOM 并在已导出的 zip 归档的 default.bom 文件中进行查找来获取包。除非 BOM 已从 IBM Operational Decision Management (iLOG) 中导入，否则包名应始终相同。 ["Object ID", "Name"]
data_access_plan	*string*	用于为分析数据视图提供数据的数据访问方案。 ["Object ID", "Name"]
optional_attributes	*string*	要包含的派生属性的列表。 [["ID1","Name1"], ["ID2", "Name2"]]
include_xml	布尔值	如果要包含某个具有 XOM 实例数据的字段，那么为 True。除非使用 IBM Analytical Decision Management (iLOG) 节点，否则建议的设置为 false。启用此项可能会增加许多额外处理。
include_xml_field	*string*	在 include_xml 设置为 true时要添加的字段的名称。

第 10 章 记录操作节点属性

appendnode 属性

"追加"节点用于连接多组记录。另外，也可以用于将结构类似但内容不同的数据集组合到一起。

示例

```
node = stream.create("append", "My node")
node.setPropertyValue("match_by", "Name")
node.setPropertyValue("match_case", True)
node.setPropertyValue("include_fields_from", "All")
node.setPropertyValue("create_tag_field", True)
node.setPropertyValue("tag_field_name", "Append_Flag")
```

表 *61. appendnode* 属性.

appendnode 属性	数据类型	属性描述
match_by	Position Name	可以根据字段在主数据源中的位置或输入数据集中字段的名称来附加数据集。
match_case	标志	匹配字段名称时启用区分大小写。
include_fields_from	Main All	
create_tag_field	标志	
tag_field_name	字符串	

aggregatenode 属性

"汇总"节点将一系列输入记录替换为经过摘要和汇总的输出记录。

示例

```
node = stream.create("aggregate", "My node")
# dbnode is a configured database import node
stream.link(dbnode, node)
node.setPropertyValue("contiguous", True)
node.setPropertyValue("keys", ["Drug"])
node.setKeyedPropertyValue("aggregates", "Age", ["Sum", "Mean"])
node.setPropertyValue("inc_record_count", True)
node.setPropertyValue("count_field", "index")
node.setPropertyValue("extension", "Aggregated_")
node.setPropertyValue("add_as", "Prefix")
```

表 62. *aggregatenode* 属性.

aggregatenode 属性	**数据类型**	**属性描述**
keys	列表	列出可用作汇总键的字段。例如，如果 Sex 和 Region 是键字段，M 和 F 与区域 N 和 S 的每个唯一性组合（四个唯一性组合）都将具有一个经过汇总的记录。
contiguous	标志	如果您知道输入中所有具有相同键值的记录都分组到一起（例如，如果按键字段对输入进行了排序），请选择此选项。这样做有助于提高性能。
aggregates		一种结构化属性，它列出其值将被汇总的数字字段以及选定的汇总模式。
aggregate_exprs		键控属性，输入派生字段名称（带有用于计算此名称的汇总表达式）。例如： `aggregatenode.setKeyedPropertyValue ("aggregate_exprs", "Na_MAX", "MAX('Na')")`
extension	字符串	对重复的汇总字段指定前缀或后缀（样本如下）。
add_as	Suffix Prefix	
inc_record_count	标志	创建一个额外字段，该字段指定为形成每条汇总记录汇总了多少条输入记录。
count_field	字符串	指定记录计数字段的名称。
allow_approximation	布尔值	在 Analytic Server 中执行汇总时允许估算订单统计
bin_count	*integer*	指定要在估算中使用的分级数

balancenode 属性

"均衡"节点用于纠正数据集中的不平衡，以使其遵循指定的条件。"均衡"伪指令根据指定系数调整条件成立的记录所占的比例。

示例

```
node = stream.create("balance", "My node")
node.setPropertyValue("training_data_only", True)
node.setPropertyValue("directives", [[1.3, "Age > 60"], [1.5, "Na > 0.5"]])
```

表 63. *balancenode* 属性.

balancenode 属性	**数据类型**	**属性描述**
directives		根据指定数字平衡字段值比例的结构化属性（参阅下面的示例）。
training_data_only	标志	指定只应该对训练数据进行平衡。如果流中不存在分区字段，那么将忽略此选项。

此节点属性使用以下格式：

[[*number, string*] \ [*number, string*] \ ... [*number, string*]].

注：如果在表达式中嵌入了使用双引号的字符串，那么必须在这些字符串之前加上转义字符"\"。"\"字符同时也是行连字符，可用于清楚地将参数排列成一行。

cplexoptnode 属性

借助 CPLEX Optimization 节点，可以通过优化编程语言 (OPL) 模型文件来使用基于优化的复杂数学 (CPLEX)。IBM Analytical Decision Management 产品中提供了此功能，但您现在还可以在 SPSS Modeler 中使用 CPLEX 节点而不需要 IBM Analytical Decision Management。

有关 CPLEX Optimization 和 OPL 的更多信息，请参阅 IBM Analytical Decision Management 文档。

表 *64. cplexoptnode* 属性

cplexoptnode 属性	**数据类型**	**属性说明**
opl_model_text	字符串	将由 CPLEX Optimization 节点运行并生成优化结构的 OPL（优化编程语言）脚本程序。
opl_tuple_set_name	字符串	这是 OPL 模型中对应于传入数据的元组集合名称。这不是必需的，并且通常通过脚本进行设置。仅应用于编辑所选数据源的字段映射。
data_input_map	结构化属性的列表	数据源的输入字段映射。这不是必需的，并且通常通过脚本进行设置。仅应用于编辑所选数据源的字段映射。

表 *64. cplexoptnode* 属性 *(续)*

cplexoptnode 属性	**数据类型**	**属性说明**
md_data_input_map	结构化属性的列表	OPL 中定义的每个元组与每个对应字段数据源（传入数据）之间的字段映射。用户可以根据每个数据源进行逐个编辑。通过此脚本，可以直接设置属性来一次性设置所有映射。该设置不会显示在用户界面中。 列表中的每个实体都是结构化数据： **数据源标记。**数据源的标记，位于数据源下拉列表中。例如，对于 0_Products_Type，该标记是 0。 **数据源索引。**数据源的自然顺序（索引）。这由连接顺序确定。 **源节点。**数据源的源节点（注释）。该属性位于数据源下拉列表中。例如，对于 0_Products_Type，源节点是 Products。 **连接节点。**连接当前 CPLEX 优化节点的先验节点（注释）。该属性位于数据源下拉列表中。例如，对于 0_Products_Type，连接节点是 Type。 **元组集合名称。**数据源的元组集合名称。它必须与 OPL 中定义的内容匹配。 **元组字段名称。**数据源的元组集合字段名称。它必须与 OPL 元组集合定义中定义的内容匹配。 **存储类型。**字段存储类型。可能的值是 int、float 或 string。 **数据字段名称。**数据源的字段名称。**示例：** [[0,0,'Product','Type','Products','prod_id_tup','int','prod_id'], [0,0,'Product','Type','Products','prod_name_tup','string', 'prod_name'],[1,1,'Components','Type','Components', 'comp_id_tup','int','comp_id'],[1,1,'Components','Type', 'Components','comp_name_tup','string','comp_name']]
opl_data_text	字符串	这是用于 OPL 的一些变量或数据的定义。
output_value_mode	字符串	可能的值为 raw 或 dvar。如果指定了 dvar，那么用户必须在"输出"选项卡上指定 OPL 中用于输出的对象函数变量名称。如果指定了 raw，那么将直接输出目标函数，而与名称无关。
decision_variable_name	字符串	OPL 中定义的目标函数变量名称。仅当 output_value_mode 属性设置为 dvar 时才会启用该变量。
objective_function_value_fieldname	字符串	目标函数值要在输出中使用的字段名称。缺省值为 _OBJECTIVE。

表 64. *cplexoptnode* 属性 (续)

cplexoptnode 属性	数据类型	属性说明
output_tuple_set_names	字符串	来自传入数据的预定义元组的名称。该名称充当决策变量的索引，预期随着变量输出一起输出。输出元组必须与 OPL 中的决策变量定义一致。如果有多个索引，必须用逗号 (,) 连接元组名称。 单个元组的示例是 Products，对应的 OPL 定义是 dvar float+ Production[Products]; 多个元组的示例是 Products,Components，对应的 OPL 定义是 dvar float+ Production[Products][Components];
decision_output_map	结构化属性的列表	OPL 中定义的将成为输出的变量与输出字段之间的字段映射。列表中的每个实体都是结构化数据: **变量名称**。OPL 中要输出的变量名称。 **存储类型**。可能的值是 int、float 或 string。 **输出字段名称**。结果中的预期字段名称（输出或导出）。**示例:** [['Production','int','res'],['Remark','string','res_1']['Cost','float','res_2']]

derive_stbnode 属性

"空间时间限制"节点根据纬度、经度和时间戳记字段派生了空间时间限制。您还可以将频繁的空间时间限制标识为逗留。

示例

```
node = modeler.script.stream().createAt("derive_stb", "My node", 96, 96)

# Individual Records mode
node.setPropertyValue("mode", "IndividualRecords")
node.setPropertyValue("latitude_field", "Latitude")
node.setPropertyValue("longitude_field", "Longitude")
node.setPropertyValue("timestamp_field", "OccurredAt")
node.setPropertyValue("densities", ["STB_GH7_1HOUR", "STB_GH7_30MINS"])
node.setPropertyValue("add_extension_as", "Prefix")
node.setPropertyValue("name_extension", "stb_")

# Hangouts mode
node.setPropertyValue("mode", "Hangouts")
node.setPropertyValue("hangout_density", "STB_GH7_30MINS")
node.setPropertyValue("id_field", "Event")
node.setPropertyValue("qualifying_duration", "30MINUTES")
node.setPropertyValue("min_events", 4)
node.setPropertyValue("qualifying_pct", 65)
```

表 *65.* "空间时间限制"节点属性

derive_stbnode 属性	**数据类型**	**属性描述**
mode	IndividualRecords Hangouts	
latitude_field	字段	
longitude_field	字段	
timestamp_field	字段	
hangout_density	密度	单一密度。请参阅 densities 以了解有效的密度值。
densities	[*density*,*density*,..., *density*]	每个密度都是一个字符串，例如 STB_GH8_1DAY。 **注：**对于哪些密度有效，存在限制。对于 geohash，可以使用 GH1 到 GH15 中的值。对于 temporal 部分，可以使用下列值： EVER 1YEAR 1MONTH 1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2HOURS 1HOUR 30MINS 15MINS 10MINS 5MINS 2MINS 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SEC
id_field	字段	
qualifying_duration	1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2Hours 1HOUR 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	必须是字符串。
min_events	*integer*	最小有效整数值为 2。

表 65. "空间时间限制"节点属性 (续)

derive_stbnode 属性	数据类型	属性描述
qualifying_pct	*integer*	必须介于 1 与 100 之间。
add_extension_as	Prefix Suffix	
name_extension	*string*	

distinctnode 属性

"区分"节点通过将第一个区分记录传递到数据流，或者通过丢弃第一个记录并将任何重复记录传递到数据流，除去重复的记录。

示例

```
node = stream.create("distinct", "My node")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("fields", ["Age" "Sex"])
node.setPropertyValue("keys_pre_sorted", True)
```

表 66. *distinctnode* 属性.

distinctnode 属性	数据类型	属性描述
mode	Include Discard	既可以将第一条区分记录包括在数据流中，也可以丢弃第一条区分记录并将任何重复记录传送到数据流。
grouping_fields	列表	列出用于确定记录是否相同的字段。 **注：**从 IBM SPSS Modeler 16 起，不推荐使用此属性。
composite_value	结构化槽	请参阅以下示例。
composite_values	结构化槽	请参阅以下示例。
inc_record_count	*flag*	创建一个额外字段，该字段指定为形成每条汇总记录汇总了多少条输入记录。
count_field	*string*	指定记录计数字段的名称。
sort_keys	结构化槽。	**注：**从 IBM SPSS Modeler 16 起，不推荐使用此属性。
default_ascending	*flag*	
low_distinct_key_count	标志	指定只存在少量记录以及/或者键字段的少量唯一值。
keys_pre_sorted	标志	指定具有相同键值的所有记录在输入中分组在一起。
disable_sql_generation	标志	

composite_value 属性的示例

composite_value 属性具有以下通用格式：

```
node.setKeyedPropertyValue("composite_value", FIELD, FILLOPTION)
```

FILLOPTION 的格式为 [FillType, Option1, Option2, ...].

示例：

```
node.setKeyedPropertyValue("composite_value", "Age", ["First"])
node.setKeyedPropertyValue("composite_value", "Age", ["last"])
node.setKeyedPropertyValue("composite_value", "Age", ["Total"])
node.setKeyedPropertyValue("composite_value", "Age", ["Average"])
node.setKeyedPropertyValue("composite_value", "Age", ["Min"])
node.setKeyedPropertyValue("composite_value", "Age", ["Max"])
node.setKeyedPropertyValue("composite_value", "Date", ["Earliest"])
node.setKeyedPropertyValue("composite_value", "Date", ["Latest"])
node.setKeyedPropertyValue("composite_value", "Code", ["FirstAlpha"])
node.setKeyedPropertyValue("composite_value", "Code", ["LastAlpha"])
```

定制选项需要多个参数，这些参数将以列表形式添加，例如：

```
node.setKeyedPropertyValue("composite_value", "Name", ["MostFrequent", "FirstRecord"])
node.setKeyedPropertyValue("composite_value", "Date", ["LeastFrequent", "LastRecord"])
node.setKeyedPropertyValue("composite_value", "Pending", ["IncludesValue", "T", "F"])
node.setKeyedPropertyValue("composite_value", "Marital", ["FirstMatch", "Married", "Divorced",
"Separated"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "Space"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "Comma"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "UnderScore"])
```

composite_values 属性的示例

composite_values 属性具有以下通用格式：

```
node.setPropertyValue("composite_values", [
                        [FIELD1, [FILLOPTION1]],
                        [FIELD2, [FILLOPTION2]],
.
.
])
```

示例：

```
node.setPropertyValue("composite_values", [
    ["Age", ["First"]],
    ["Name", ["MostFrequent", "First"]],
    ["Pending", ["IncludesValue", "T"]],
    ["Marital", ["FirstMatch", "Married", "Divorced", "Separated"]],
    ["Code", ["Concatenate", "Comma"]]
])
```

extensionprocessnode 属性

通过扩展变换节点，可以将提取来自流的数据，并使用 R 脚本编制或 Python for Spark 脚本编制将变换应用于此数据。

Python for Spark 示例

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_process", "extension_process")
node.setPropertyValue("syntax_type", "Python")

process_script = """
import spss.pyspark.runtime
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()

if cxt.isComputeDataModelOnly():
    _schema = StructType([StructField("Age", LongType(), nullable=True), \
                        StructField("Sex", StringType(), nullable=True), \
                        StructField("BP", StringType(), nullable=True), \
                        StructField("Na", DoubleType(), nullable=True), \
                        StructField("K", DoubleType(), nullable=True), \
                        StructField("Drug", StringType(), nullable=True)])
    cxt.setSparkOutputSchema(_schema)
else:
    df = cxt.getSparkInputData()
    print df.dtypes[:]
    _newDF = df.select("Age","Sex","BP","Na","K","Drug")
    print _newDF.dtypes[:]
    cxt.setSparkOutputData(_newDF)
"""

node.setPropertyValue("python_syntax", process_script)
```

R 示例

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", """day<-as.Date(modelerData$dob, format="%Y-%m-%d")
next_day<-day + 1
modelerData<-cbind(modelerData,next_day)
var1<-c(fieldName="Next day",fieldLabel="",fieldStorage="date",fieldMeasure="",fieldFormat="",
fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

表 *67. extensionprocessnode* 属性

extensionprocessnode 属性	数据类型	属性描述
syntax_type	*R* *Python*	指定要运行的脚本 - R 或 Python（R 为缺省值）
r_syntax	*string*	要运行的 R 脚本编制语法。
python_syntax	*string*	要运行的 Python 脚本编制语法。
use_batch_size	*flag*	启用使用批处理。

表 67. *extensionprocessnode* 属性 (续)

extensionprocessnode 属性	数据类型	属性描述
batch_size	*integer*	指定要包含在每个批处理中的数据记录数。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_missing	*flag*	此选项用于将缺失值转换为 R NA 值。
convert_datetime	*flag*	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。

mergenode 属性

"合并"节点使用多个输入记录，并创建包含某些或全部输入字段的单个输出记录。这对于合并来源不同的数据 非常有用，例如内部客户数据和已购买人群统计数据。

示例

```
node = stream.create("merge", "My node")
# assume customerdata and salesdata are configured database import nodes
stream.link(customerdata, node)
stream.link(salesdata, node)
node.setPropertyValue("method", "Keys")
node.setPropertyValue("key_fields", ["id"])
node.setPropertyValue("common_keys", True)
node.setPropertyValue("join", "PartialOuter")
node.setKeyedPropertyValue("outer_join_tag", "2", True)
node.setKeyedPropertyValue("outer_join_tag", "4", True)
node.setPropertyValue("single_large_input", True)
node.setPropertyValue("single_large_input_tag", "2")
node.setPropertyValue("use_existing_sort_keys", True)
node.setPropertyValue("existing_sort_keys", [["id", "Ascending"]])
```

表 68. *mergenode* 属性.

mergenode 属性	数据类型	属性描述
method	Order Keys Condition Rankedcondition	指定记录是否按它们在数据文件中的列示顺序进行合并、是否使用一个或多个键字段来合并在键字段中包含相同值的记录、是否在满足指定条件时合并记录或者是否要合并主要数据集和所有次要数据集中的每个行对；使用排秩表达式可以按从低到高的顺序对任何多个匹配项进行排序。
condition	字符串	如果 method 设置为 Condition，指定包括或丢弃记录的条件。
key_fields	列表	
common_keys	标志	

表 68. *mergenode* 属性 (续).

mergenode 属性	数据类型	属性描述
join	Inner FullOuter PartialOuter Anti	
outer_join_tag.n	标志	在此属性中，*n* 是"选择数据集"对话框中显示的标记名。注意，可以指定多个标记名，因为任何数量的数据集都无法提供完整记录。
single_large_input	标志	指定是否进行优化，以使一个输入与其他输入相比具有一个相对较大的输入值。
single_large_input_tag	字符串	按"选择较大数据集"对话框中的显示指定标记名。请注意，该属性的用法与 outer_join_tag 属性的用法略有不同（标记与字符串），因为前者只能指定一个输入数据集。
use_existing_sort_keys	标志	指定输入值是否已根据一个或多个键字段进行排序。
existing_sort_keys	[['*string*', 'Ascending'] \ ['*string*', 'Descending']]	指定已排序的字段及其排序方向。
primary_dataset	*string*	如果 method 为 Rankedcondition，请在合并中选择主要数据集。此数据集可以视为外部连接合并的左侧。
rename_duplicate_fields	布尔值	如果方法为 Rankedcondition，并且此项设置为 Y，并且如果生成的合并数据集包含来自不同数据源的多个同名字段，那么来自数据集的各个标记将添加在字段列标题的开头处。
merge_condition	*string*	
ranking_expression	*string*	
Num_matches	*integer*	要返回的匹配项数，基于 merge_condition 和 ranking_expression。最小值为 1，最大值为 100。

rfmaggregatenode 属性

使用"近因、频率和货币 (RFM) 汇总"节点，您可以采用客户的 历史记录事务处理数据，删除所有无用数据以及将所有他们保留的事务处理数据组合成一行，且该行中列出了他们与您上次谈业务的时间、所完成的交易量以及这些交易的总货币价值。

示例

```
node = stream.create("rfmaggregate", "My node")
node.setPropertyValue("relative_to", "Fixed")
node.setPropertyValue("reference_date", "2007-10-12")
node.setPropertyValue("id_field", "CardID")
node.setPropertyValue("date_field", "Date")
node.setPropertyValue("value_field", "Amount")
node.setPropertyValue("only_recent_transactions", True)
node.setPropertyValue("transaction_date_after", "2000-10-01")
```

表 69. *rfmaggregatenode* 属性.

rfmaggregatenode 属性	数据类型	属性描述
relative_to	Fixed Today	指定计算交易近因的日期。
reference_date	日期	仅在 relative_to 中选择 Fixed 时才可用。
contiguous	标志	如果您的数据进行了预先排序，以便所有标识相同的记录一起出现在数据流中，那么选择此选项可以加快处理速度。
id_field	字段	指定该字段以用来识别客户及其交易。
date_field	字段	指定将要用来计算近因的日期字段。
value_field	字段	指定该字段以用来计算货币值。
extension	字符串	为重复汇总字段指定前缀或后缀。
add_as	Suffix Prefix	指定是否应作为前缀或后缀来添加 extension。
discard_low_value_records	标志	启用使用 discard_records_below 设置。
discard_records_below	*number*	可在计算 RFM 总计时，指定一个最小值，凡低于该值的交易详细信息都不再被使用。值单位与所选 value 字段相关。
only_recent_transactions	标志	启用使用 specify_transaction_date 或 transaction_within_last 设置。
specify_transaction_date	标志	
transaction_date_after	日期	只有选中 specify_transaction_date 时才可用。指定交易日期以在分析时包含其之后的记录。
transaction_within_last	*number*	只有选中 transaction_within_last 时才可用。指定从计算相对于以下内容的近因日期字段所返回的周期数和周期类型（天、周、月或年），在此日期之后的记录将被包含在您的分析中。
transaction_scale	Days Weeks Months Years	只有选中 transaction_within_last 时才可用。指定从计算相对于以下内容的近因日期字段所返回的周期数和周期类型（天、周、月或年），在此日期之后的记录将被包含在您的分析中。
save_r2	标志	显示每个客户第二个最近交易的日期。
save_r3	标志	只有选中 save_r2 时才可用。显示每个客户第三个最近交易的日期。

Rprocessnode 属性

通过使用您自己的定制 R 脚本，可以使用"R 变换"节点从 IBM(r) SPSS(r) Modeler 流中获取数据并进行修改。修改数据后，会将数据返回到流中。

示例

```
node = stream.create("rprocess", "My node")
node.setPropertyValue("custom_name", "my_node")
node.setPropertyValue("syntax", """day<-as.Date(modelerData$dob, format="%Y-%m-%d")
next_day<-day + 1
modelerData<-cbind(modelerData,next_day)
var1<-c(fieldName="Next day",fieldLabel="",fieldStorage="date",fieldMeasure="",fieldFormat="",
fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
node.setPropertyValue("convert_datetime", "POSIXct")
```

表 *70. Rprocessnode* 属性.

Rprocessnode 属性	**数据类型**	**属性描述**
syntax	*string*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*flag*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*flag*	
use_batch_size	*flag*	支持使用批处理
batch_size	*integer*	指定要包含在每个批处理中的数据记录数

samplenode 属性

"样本"节点用于选择一部分记录。支持各种样本类型，包括分层、聚类和非随机（结构化）样本。采样对于提高性能以及选择相关记录组或事务组进行分析十分有用。

示例

```
/* Create two Sample nodes to extract   different samples from the same data */
node = stream.create("sample", "My node")
node.setPropertyValue("method", "Simple")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("sample_type", "First")
node.setPropertyValue("first_n", 500)

node = stream.create("sample", "My node")
node.setPropertyValue("method", "Complex")
node.setPropertyValue("stratify_by", ["Sex", "Cholesterol"])
node.setPropertyValue("sample_units", "Proportions")
node.setPropertyValue("sample_size_proportions", "Custom")
node.setPropertyValue("sizes_proportions", [["M", "High", "Default"], ["M", "Normal", "Default"],
 ["F", "High", 0.3], ["F", "Normal", 0.3]])
```

表 *71. samplenode* 属性.

samplenode 属性	**数据类型**	**属性描述**
method	Simple Complex	

表 *71. samplenode* 属性 *(续).*

samplenode 属性	**数据类型**	**属性描述**
mode	Include Discard	包括或丢弃满足指定条件的记录。
sample_type	First OneInN RandomPct	指定抽样方法。
first_n	*integer*	将包括或丢弃直到指定截止点的记录。
one_in_n	*number*	每隔 *n-1* 条记录包括或丢弃一条记录。
rand_pct	*number*	指定要包括或丢弃记录的百分比。
use_max_size	标志	启用使用 maximumiscard_records_below 设置。
maximum_size	*integer*	指定要包括在数据流中或丢弃的最大样本量。此选项是冗余选项，因此指定 First 和 Include 时会被禁用。
set_random_seed	标志	启用随机种子设置。
random_seed	*integer*	指定用作随机种子的值。
complex_sample_type	Random Systematic	
sample_units	Proportions Counts	
sample_size_proportions	Fixed Custom Variable	
sample_size_counts	Fixed Custom Variable	
fixed_proportions	*number*	
fixed_counts	*integer*	
variable_proportions	字段	
variable_counts	字段	
use_min_stratum_size	标志	
minimum_stratum_size	*integer*	仅当抽取复杂样本 Sample units=Proportions 时才应用此选项。
use_max_stratum_size	标志	
maximum_stratum_size	*integer*	仅当抽取复杂样本 Sample units=Proportions 时才应用此选项。
clusters	字段	
stratify_by	*[field1 ... fieldN]*	
specify_input_weight	标志	
input_weight	字段	
new_output_weight	字符串	
sizes_proportions	[[string *string value*][string *string value*]...]	如果 sample_units=proportions 且 sample_size_proportions=Custom，指定层字段值每个可能组合的值。
default_proportion	*number*	

表 *71. samplenode* 属性 *(续).*

samplenode 属性	**数据类型**	**属性描述**
sizes_counts	[[string *string value*][string *string value*]...]	指定层字段值每个可能的组合值。用法与 sizes_proportions 的用法相似，但指定的是整数，而非比例。
default_count	*number*	

selectnode 属性

"选择"节点根据特定条件从数据流中选择或废弃一部分记录。例如，可以选择与特定销售区域相关的记录。

示例

```
node = stream.create("select", "My node")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("condition", "Age < 18")
```

表 *72. selectnode* 属性.

selectnode 属性	**数据类型**	**属性描述**
mode	Include Discard	指定是包括还是丢弃选定记录。
condition	字符串	包括或丢弃记录的条件。

sortnode 属性

"排序"节点根据一个或多个字段的值按升序或降序对记录进行排序。

示例

```
node = stream.create("sort", "My node")
node.setPropertyValue("keys", [["Age", "Ascending"], ["Sex", "Descending"]])
node.setPropertyValue("default_ascending", False)
node.setPropertyValue("use_existing_keys", True)
node.setPropertyValue("existing_keys", [["Age", "Ascending"]])
```

表 *73. sortnode* 属性.

sortnode 属性	**数据类型**	**属性描述**
keys	列表	指定要作为排序依据的字段。如果未指定方向，则会使用缺省值。
default_ascending	标志	指定缺省排序顺序。

表 73. *sortnode* 属性 (续).

sortnode 属性	数据类型	属性描述
use_existing_keys	标志	指定是否使用以前已排序字段的排序顺序来优化现在的排序。
existing_keys		指定已排序的字段及其排序方向。使用的格式与 keys 属性相同。

spacetimeboxes 属性

空间时间限制 (STB) 是进行了地理散列计算的空间位置的扩展。更具体而言，STB 是一个字母数字字符串，它表示形状规则的空间及时间区域。

表 74. *spacetimeboxes* 属性

spacetimeboxes 属性	数据类型	属性描述
mode	*IndividualRecords* *Hangouts*	
latitude_field	字段	
longitude_field	字段	
timestamp_field	字段	

表 *74. spacetimeboxes* 属性 *(续)*

spacetimeboxes 属性	**数据类型**	**属性描述**
densities	*[density, density, density...]*	每个 density 都是一个字符串。例如：STB_GH8_1DAY 请注意，对于哪些密度有效，存在限制。 对于 geohash，可以使用 GH1 到 GH15 中的值。 对于 temporal 部分，可以使用下列值： EVER 1YEAR 1MONTH 1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2HOURS 1HOUR 30MINS 15MINS 10MINS 5MINS 2 MINS 1 MIN 30SECS 15SECS 10SECS 5 SECS 2 SECS 1SEC
field_name_extension	*string*	
add_extension_as	*Prefix* *Suffix*	
hangout_density	密度	单一密度（请参阅上文）
id_field	字段	
qualifying_duration	1DAY 12HOURS 8HOURS 6HOURS 4HOURS 2HOURS 1HOUR 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	必须是字符串。
min_events	*integer*	最小值为 2
qualifying_pct	*integer*	必须介于 1 与 100 之间。

streamingtimeseries 属性

"流式时间序列"节点可在一个步骤内构建时间序列模型并对其进行评分。

注：此"流式时间序列"节点替换了在 SPSS Modeler 第 18 版中不推荐的原始"流式 TS"节点。

表 75. *streamingtimeseries* 属性

streamingtimeseries 属性	**值**	**属性描述**
targets	字段	"流式时间序列"节点可以预测一个或多个目标，可以选择使用一个或多个输入字段作为预测变量。不使用频率字段和权重字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
candidate_inputs	[*field1 ... fieldN*]	模型所使用的输入字段或预测变量字段。
use_period	*flag*	
date_time_field	字段	
input_interval	None 未知 年 季度 月 周 Day 小时 Hour_nonperiod 分钟 Minute_nonperiod 秒 Second_nonperiod	
period_field	字段	
period_start_value	*integer*	
num_days_per_week	*integer*	
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	*integer*	
start_hour_of_day	*integer*	
timestamp_increments	*integer*	
cyclic_increments	*integer*	

表 *75. streamingtimeseries* 属性 *(续)*

streamingtimeseries 属性	**值**	**属性描述**
cyclic_periods	列表	
output_interval	None 年 季度 月 周 Day 小时 分钟 秒	
is_same_interval	*flag*	
cross_hour	*flag*	
aggregate_and_distribute	列表	
aggregate_default	Mean 合计 众数 最小值 最大值	
distribute_default	Mean 合计	
group_default	Mean 合计 众数 最小值 最大值	
missing_imput	Linear_interp Series_mean K_mean K_median Linear_trend	
k_span_points	*integer*	
use_estimation_period	*flag*	
estimation_period	Observations 时间	
date_estimation	列表	仅当使用 date_time_field 时才可用
period_estimation	列表	仅当使用 use_period 时才可用
observations_type	Latest 最早	
observations_num	*integer*	
observations_exclude	*integer*	
method	ExpertModeler Exsmooth Arima	

表 *75. streamingtimeseries* 属性 *(续)*

streamingtimeseries 属性	值	属性描述
expert_modeler_method	ExpertModeler Exsmooth Arima	
consider_seasonal	*flag*	
detect_outliers	*flag*	
expert_outlier_additive	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_innovational	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_transient	*flag*	
expert_outlier_seasonal_additive	*flag*	
expert_outlier_local_trend	*flag*	
expert_outlier_additive_patch	*flag*	
consider_newesmodels	*flag*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative DampedTrendAdditive DampedTrendMultiplicative MultiplicativeTrendAdditive MultiplicativeSeasonal MultiplicativeTrendMultiplicative MultiplicativeTrend	
futureValue_type_method	Compute specify	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima.p	*integer*	
arima.d	*integer*	
arima.q	整数	
arima.sp	整数	
arima.sd	整数	
arima.sq	整数	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*flag*	
tf_arima.p. *fieldname*	整数	用于转换函数。

表 75. *streamingtimeseries* 属性（续）

streamingtimeseries 属性	**值**	**属性描述**
tf_arima.d. *fieldname*	整数	用于转换函数。
tf_arima.q. *fieldname*	整数	用于转换函数。
tf_arima.sp. *fieldname*	整数	用于转换函数。
tf_arima.sd. *fieldname*	整数	用于转换函数。
tf_arima.sq. *fieldname*	整数	用于转换函数。
tf_arima.delay. *fieldname*	整数	用于转换函数。
tf_arima.transformation_type. *fieldname*	None SquareRoot NaturalLog	用于转换函数。
arima_detect_outliers	*flag*	
arima_outlier_additive	*flag*	
arima_outlier_level_shift	*flag*	
arima_outlier_innovational	*flag*	
arima_outlier_transient	*flag*	
arima_outlier_seasonal_additive	*flag*	
arima_outlier_local_trend	*flag*	
arima_outlier_additive_patch	*flag*	
conf_limit_pct	*real*	
events	字段	
forecastperiods	整数	
extend_records_into_future	*flag*	
conf_limits	*flag*	
noise_res	*flag*	

streamingts 属性（不推荐）

注：此原始"流式时间序列"节点在 SPSS Modeler V18 中已不推荐，并且替换为新"流式时间序列"节点，此节点设计为利用 IBM SPSS Analytic Server 能力来处理大数据。"流式 TS"节点在某个步骤中构建时间序列模型并对其进行评估，而不需要"时间区间"节点。

示例

```
node = stream.create("streamingts", "My node")
node.setPropertyValue("deployment_force_rebuild", True)
node.setPropertyValue("deployment_rebuild_mode", "Count")
node.setPropertyValue("deployment_rebuild_count", 3)
node.setPropertyValue("deployment_rebuild_pct", 11)
node.setPropertyValue("deployment_rebuild_field", "Year")
```

表 *76. streamingts* 属性.

streamingts 属性	**数据类型**	**属性描述**
custom_fields	*flag*	如果 custom_fields=false，那么将使用上游"类型"节点的当前设置。如果 custom_fields=true，那么必须指定 targets 和 inputs。
targets	[*field1...fieldN*]	
inputs	[*field1...fieldN*]	
method	ExpertModeler Exsmooth Arima	
calculate_conf	*flag*	
conf_limit_pct	*real*	
use_time_intervals_node	*flag*	如果 use_time_intervals_node=true，那么将使用上游"时间区间"节点的设置。如果 use_time_intervals_node=false，那么必须指定 interval_offset_position、interval_offset 和 interval_type。
interval_offset_position	LastObservation LastRecord	LastObservation 是指**最后一个有效观测值**。LastRecord 是指**从最后一个记录计数**。
interval_offset	*number*	
interval_type	Periods Years Quarters Months WeeksNonPeriodic DaysNonPeriodic HoursNonPeriodic MinutesNonPeriodic SecondsNonPeriodic	
events	字段	
expert_modeler_method	AllModels Exsmooth Arima	
consider_seasonal	*flag*	
detect_outliers	*flag*	
expert_outlier_additive	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_innovational	*flag*	
expert_outlier_transient	*flag*	
expert_outlier_seasonal_additive	*flag*	
expert_outlier_local_trend	*flag*	
expert_outlier_additive_patch	*flag*	
exsmooth_model_type	SimpleHoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	

表 *76. streamingts* 属性 *(续).*

streamingts 属性	**数据类型**	**属性描述**
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*integer*	对于"时间序列"建模节点是同一属性
arima_d	*integer*	对于"时间序列"建模节点是同一属性
arima_q	*integer*	对于"时间序列"建模节点是同一属性
arima_sp	*integer*	对于"时间序列"建模节点是同一属性
arima_sd	*integer*	对于"时间序列"建模节点是同一属性
arima_sq	*integer*	对于"时间序列"建模节点是同一属性
arima_transformation_type	None SquareRoot NaturalLog	对于"时间序列"建模节点是同一属性
arima_include_constant	*flag*	对于"时间序列"建模节点是同一属性
tf_arima_p.*fieldname*	*integer*	对于"时间序列"建模节点是同一属性。用于转换函数。
tf_arima_d.*fieldname*	*integer*	对于"时间序列"建模节点是同一属性。用于转换函数。
tf_arima_q.*fieldname*	*integer*	对于"时间序列"建模节点是同一属性。用于转换函数。
tf_arima_sp.*fieldname*	*integer*	对于"时间序列"建模节点是同一属性。用于转换函数。
tf_arima_sd.*fieldname*	*integer*	对于"时间序列"建模节点是同一属性。用于转换函数。
tf_arima_sq.*fieldname*	*integer*	对于"时间序列"建模节点是同一属性。用于转换函数。
tf_arima_delay.*fieldname*	*integer*	对于"时间序列"建模节点是同一属性。用于转换函数。
tf_arima_transformation_type.*fieldname*	None SquareRoot NaturalLog	
arima_detect_outlier_mode	None Automatic	
arima_outlier_additive	*flag*	
arima_outlier_level_shift	*flag*	
arima_outlier_innovational	*flag*	
arima_outlier_transient	*flag*	
arima_outlier_seasonal_additive	*flag*	
arima_outlier_local_trend	*flag*	
arima_outlier_additive_patch	*flag*	
deployment_force_rebuild	*flag*	
deployment_rebuild_mode	Count Percent	
deployment_rebuild_count	*number*	
deployment_rebuild_pct	*number*	

表 *76. streamingts* 属性 *(续).*

streamingts 属性	**数据类型**	**属性描述**
deployment_rebuild_field	<字段>	

第 11 章 字段操作节点属性

anonymizenode 属性

"匿名化"节点用于转换字段名和字段值在下游的表示方式，从而掩饰原始数据。如果要允许其他用户构建含有敏感数据（例如客户名称或其他详细信息）的模型，那么这种节点十分有用。

示例

```
stream = modeler.script.stream()
varfilenode = stream.createAt("variablefile", "File", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
node = stream.createAt("anonymize", "My node", 192, 96)
# Anonymize node requires the input fields while setting the values
stream.link(varfilenode, node)
node.setKeyedPropertyValue("enable_anonymize", "Age", True)
node.setKeyedPropertyValue("transformation", "Age", "Random")
node.setKeyedPropertyValue("set_random_seed", "Age", True)
node.setKeyedPropertyValue("random_seed", "Age", 123)
node.setKeyedPropertyValue("enable_anonymize", "Drug", True)
node.setKeyedPropertyValue("use_prefix", "Drug", True)
node.setKeyedPropertyValue("prefix", "Drug", "myprefix")
```

表 *77. anonymizenode* 属性

anonymizenode 属性	**数据类型**	**属性描述**
enable_anonymize	*flag*	设置为 True 时，可激活匿名化字段值（相当于在"对值进行匿名化"列中对该字段选择**是**）。
use_prefix	*flag*	设置为 True 时，如果已指定定制前缀，那么将使用该前缀。适用于将通过杂凑法被匿名化的字段，而且相当于在"替换值"对话框中为该字段选择**定制**单选按钮。
prefix	*string*	相当于在"替换值"对话框的文本框中输入前缀。如果未指定其他任何值，那么缺省前缀即是该缺省值。
transformation	Random Fixed	确定通过转换法匿名化的字段的转换参数是随机的还是固定的。
set_random_seed	*flag*	设置为 True 时，将使用指定的种子值（如果 transformation 也设置为 Random）。
random_seed	*integer*	set_random_seed 设置为 True 时，此值为随机数的种子。
scale	*number*	transformation 设置为 Fixed 时，此值用于"转换尺度"。通常，最大尺度值为 10，但可能会被减小以避免溢出。
translate	*number*	transformation 设置为 Fixed 时，此值用于"转换"。通常，最大转换值为 1000，但可能会被减小以避免溢出。

autodataprepnode 属性

"自动数据准备 (ADP)"节点可分析您的数据并标识修正，筛选出存在问题或可能无用的字段，并在适当的情况下派生新的属性，通过智能筛选和抽样技术改进性能。您可以采用完全自动化方式使用此节点，从而允许此节点选择并应用修订，另外也可以在应用修订前预览更改并根据需要接受、拒绝或进行修改。

示例

```
node = stream.create("autodataprep", "My node")
node.setPropertyValue("objective", "Balanced")
node.setPropertyValue("excluded_fields", "Filter")
node.setPropertyValue("prepare_dates_and_times", True)
node.setPropertyValue("compute_time_until_date", True)
node.setPropertyValue("reference_date", "Today")
node.setPropertyValue("units_for_date_durations", "Automatic")
```

表 *78. autodataprepnode* 属性

autodataprepnode 属性	**数据类型**	**属性描述**
objective	Balanced Speed Accuracy Custom	
custom_fields	*flag*	如果为 true，那么允许您为当前节点指定目标字段、输入字段和其他字段。如果为 false，那么将使用上游"类型"节点的当前设置。
target	字段	指定单个目标字段。
inputs	[*field1 ... fieldN*]	模型所使用的输入字段或预测变量字段。
use_frequency	*flag*	
frequency_field	字段	
use_weight	*flag*	
weight_field	字段	
excluded_fields	Filter 无	
if_fields_do_not_match	StopExecution ClearAnalysis	
prepare_dates_and_times	*flag*	控制对所有日期与时间字段的访问
compute_time_until_date	*flag*	
reference_date	Today Fixed	
fixed_date	*date*	
units_for_date_durations	Automatic Fixed	
fixed_date_units	Years Months Days	
compute_time_until_time	*flag*	

表 *78. autodataprepnode* 属性 *(续)*

autodataprepnode 属性	**数据类型**	**属性描述**
reference_time	CurrentTime Fixed	
fixed_time	*time*	
units_for_time_durations	Automatic Fixed	
fixed_date_units	Hours Minutes Seconds	
extract_year_from_date	*flag*	
extract_month_from_date	*flag*	
extract_day_from_date	*flag*	
extract_hour_from_time	*flag*	
extract_minute_from_time	*flag*	
extract_second_from_time	*flag*	
exclude_low_quality_inputs	*flag*	
exclude_too_many_missing	*flag*	
maximum_percentage_missing	*number*	
exclude_too_many_categories	*flag*	
maximum_number_categories	*number*	
exclude_if_large_category	*flag*	
maximum_percentage_category	*number*	
prepare_inputs_and_target	*flag*	
adjust_type_inputs	*flag*	
adjust_type_target	*flag*	
reorder_nominal_inputs	*flag*	
reorder_nominal_target	*flag*	
replace_outliers_inputs	*flag*	
replace_outliers_target	*flag*	
replace_missing_continuous_inputs	*flag*	
replace_missing_continuous_target	*flag*	
replace_missing_nominal_inputs	*flag*	
replace_missing_nominal_target	*flag*	
replace_missing_ordinal_inputs	*flag*	
replace_missing_ordinal_target	*flag*	
maximum_values_for_ordinal	*number*	
minimum_values_for_continuous	*number*	
outlier_cutoff_value	*number*	
outlier_method	Replace Delete	
rescale_continuous_inputs	*flag*	

表 *78. autodataprepnode* 属性 *(续)*

autodataprepnode 属性	**数据类型**	**属性描述**
rescaling_method	MinMax ZScore	
min_max_minimum	*number*	
min_max_maximum	*number*	
z_score_final_mean	*number*	
z_score_final_sd	*number*	
rescale_continuous_target	*flag*	
target_final_mean	*number*	
target_final_sd	*number*	
transform_select_input_fields	*flag*	
maximize_association_with_target	*flag*	
p_value_for_merging	*number*	
merge_ordinal_features	*flag*	
merge_nominal_features	*flag*	
minimum_cases_in_category	*number*	
bin_continuous_fields	*flag*	
p_value_for_binning	*number*	
perform_feature_selection	*flag*	
p_value_for_selection	*number*	
perform_feature_construction	*flag*	
transformed_target_name_extension	*string*	
transformed_inputs_name_extension	*string*	
constructed_features_root_name	*string*	
years_duration_ name_extension	*string*	
months_duration_ name_extension	*string*	
days_duration_ name_extension	*string*	
hours_duration_ name_extension	*string*	
minutes_duration_ name_extension	*string*	
seconds_duration_ name_extension	*string*	
year_cyclical_name_extension	*string*	
month_cyclical_name_extension	*string*	
day_cyclical_name_extension	*string*	
hour_cyclical_name_extension	*string*	
minute_cyclical_name_extension	*string*	
second_cyclical_name_extension	*string*	

astimeintervalsnode 属性

使用"时间间隔"节点可以指定时间间隔并派生用于进行估算或预测的新时间字段。支持全部范围的时间间隔，从秒到年。

表 79. *astimeintervalsnode* 属性

astimeintervalsnode 属性	数据类型	属性描述
time_field	字段	只能接受单个连续字段。该字段将由节点用作用于转换时间区间的汇总键。如果在此处使用了整数字段，那么会将此字段视为时间索引。
dimensions	*[field1 field2 ... fieldn]*	这些字段用于根据字段值创建各个时间序列。
fields_to_aggregate	*[field1 field2 ... fieldn]*	这些字段将在更改时间字段周期的过程中进行汇总。将从数据中过滤掉此选取器中未包含的任何字段，以留下节点。

binningnode 属性

"分箱"节点根据一个或多个现有连续（数字范围）字段的值自动创建新的名义（集合）字段。例如，您可以将连续收入字段转换为一个包含各组收入（作为与均值之间的偏差）的新分类字段。为新字段创建分箱后，即可根据分割点生成"派生"节点。

示例

```
node = stream.create("binning", "My node")
node.setPropertyValue("fields", ["Na", "K"])
node.setPropertyValue("method", "Rank")
node.setPropertyValue("fixed_width_name_extension", "_binned")
node.setPropertyValue("fixed_width_add_as", "Suffix")
node.setPropertyValue("fixed_bin_method", "Count")
node.setPropertyValue("fixed_bin_count", 10)
node.setPropertyValue("fixed_bin_width", 3.5)
node.setPropertyValue("tile10", True)
```

表 80. *binningnode* 属性

binningnode 属性	数据类型	属性描述
fields	*[field1 field2 ... fieldn]*	待转换的连续（数字范围）字段。可以同时对多个字段进行分箱。
method	FixedWidth EqualCount Rank SDev Optimal	用于为新字段分箱（类别）确定分割点的方法。
rcalculate_bins	Always IfNecessary	指定是每次执行节点时都重新计算分箱并将数据放入相关分箱，还是仅将数据添加到现有分箱和任何已添加的新分箱。

表 *80. binningnode* 属性 *(续)*

binningnode 属性	**数据类型**	**属性描述**
fixed_width_name_extension	字符串	缺省扩展名为 *_BIN*。
fixed_width_add_as	Suffix Prefix	指定是将扩展名添加到字段名末尾（后缀）还是开头（前缀）。缺省扩展名为 *income_BIN*。
fixed_bin_method	Width 计数	
fixed_bin_count	*integer*	指定用于确定新字段的固定宽度分箱（类别）数的整数。
fixed_bin_width	*real*	这是用于计算分箱宽度的值（整数或实数）。
equal_count_name_ extension	*string*	缺省扩展名为 *_TILE*。
equal_count_add_as	Suffix Prefix	指定针对使用标准 p-tile 法生成的字段名使用的扩展名（后缀或前缀）。缺省扩展名为 *_TILE* 加上 *N*，其中 *N* 是分位数。
tile4	*flag*	生成四分位数分箱，每个分箱中包含 25% 的观测值。
tile5	*flag*	生成五分位数分箱。
tile10	*flag*	生成十分位数分箱。
tile20	*flag*	生成二十分位数分箱。
tile100	*flag*	生成百分位数分箱。
use_custom_tile	*flag*	
custom_tile_name_extension	*string*	缺省扩展名为 *_TILEN*。
custom_tile_add_as	Suffix Prefix	
custom_tile	*integer*	
equal_count_method	RecordCount ValueSum	RecordCount 方法是为每个分箱分配相同数目的记录，而 ValueSum 方法是使分配记录后每个分箱中值的总和相等。
tied_values_method	Next Current Random	指定要输入的分箱结值数据。
rank_order	Ascending Descending	此属性包括 Ascending（最低值标记为 1）或 Descending（最高值标记为 1）。
rank_add_as	Suffix Prefix	此选项适用于排序、分数排序和百分比排序。
排名	*flag*	
rank_name_extension	*string*	缺省扩展名为 *_RANK*。
rank_fractional	*flag*	对个案进行排秩，其中新字段的值等于排秩值除以非缺失个案的权重之和。分数排序值介于 0-1 之间。
rank_fractional_name_ extension	*string*	缺省扩展名为 *_F_RANK*。
rank_pct	*flag*	每个排秩值除以具有有效值的记录数，再乘以 100。百分比分数秩处于 1-100 范围内。

表 *80. binningnode* 属性 *(续)*

binningnode 属性	**数据类型**	**属性描述**
rank_pct_name_extension	*string*	缺省扩展名为 *_P_RANK*。
sdev_name_extension	*string*	
sdev_add_as	Suffix Prefix	
sdev_count	One Two Three	
optimal_name_extension	*string*	缺省扩展名为 *_OPTIMAL*。
optimal_add_as	Suffix Prefix	
optimal_supervisor_field	字段	作为监督字段选择的字段，为分箱选择的字段与之相关。
optimal_merge_bins	*flag*	指定将所有具有较小观测值计数的分箱添加到更大的相邻分箱。
optimal_small_bin_threshold	*integer*	
optimal_pre_bin	*flag*	表示要进行数据集的预分箱。
optimal_max_bins	*integer*	指定上限以避免创建过大分箱数。
optimal_lower_end_point	Inclusive Exclusive	
optimal_first_bin	Unbounded Bounded	
optimal_last_bin	Unbounded Bounded	

derivenode 属性

"派生"节点修改数据值或者根据一个或多个现有字段创建新字段。它可以创建类型为公式、标志、名义、状态、计数和条件的字段。

示例 1

```
# Create and configure a Flag Derive field nodenode = stream.create("derive", "My node")
node.setPropertyValue("new_name", "DrugX_Flag")
node.setPropertyValue("result_type", "Flag")
node.setPropertyValue("flag_true", "1")
node.setPropertyValue("flag_false", "0")
node.setPropertyValue("flag_expr", "'Drug' == \"drugX\"")

# Create and configure a Conditional Derive field nodenode = stream.create("derive", "My node")
node.setPropertyValue("result_type", "Conditional")
node.setPropertyValue("cond_if_cond", "@OFFSET(\"Age\", 1) = \"Age\"")
node.setPropertyValue("cond_then_expr", "(@OFFSET(\"Age\", 1) = \"Age\" >< @INDEX")
node.setPropertyValue("cond_else_expr", "\"Age\"")
```

示例 2

此脚本假定有两个名为 XPos 和 YPos 的数字列，它们分别表示某个点（例如，某一事件发生的位置）的 X 和 Y 坐标。它将创建"派生"节点，此节点用于根据特定坐标系中表示该点的 X 和 Y 坐标来计算地理空间列：

```
stream = modeler.script.stream()
# Other stream configuration code
node = stream.createAt("derive", "Location", 192, 96)
node.setPropertyValue("new_name", "Location")
node.setPropertyValue("formula_expr", "['XPos', 'YPos']")
node.setPropertyValue("formula_type", "Geospatial")
# Now we have set the general measurement type, define the
# specifics of the geospatial object
node.setPropertyValue("geo_type", "Point")
node.setPropertyValue("has_coordinate_system", True)
node.setPropertyValue("coordinate_system", "ETRS_1989_EPSG_Arctic_zone_5-47")
```

表 *81. derivenode* 属性

derivenode 属性	**数据类型**	**属性描述**
new_name	*string*	新字段的名称。
mode	Single Multiple	指定单个或多个字段。
fields	列表	仅用于在"多重"方式下选择多个字段。
name_extension	*string*	指定新字段名的扩展名。
add_as	Suffix Prefix	将扩展名添加为字段名的前缀（开头）或后缀（末尾）。
result_type	Formula Flag Set State Count Conditional	可创建的六种类型的新字段。
formula_expr	*string*	这是用于在"派生"节点中计算新字段值的表达式。
flag_expr	*string*	
flag_true	*string*	
flag_false	*string*	
set_default	*string*	
set_value_cond	*string*	这是一个结构化属性，用于提供与给定值相关联的条件。
state_on_val	*string*	指定满足 On 条件时新字段的值。
state_off_val	*string*	指定满足 Off 条件时新字段的值。
state_on_expression	*string*	
state_off_expression	*string*	
state_initial	On Off	为新字段的每个记录分配初始值 On 或 Off。可在满足每个条件时更改此值。
count_initial_val	*string*	
count_inc_condition	*string*	
count_inc_expression	*string*	

表 81. derivenode 属性 (续)

derivenode 属性	数据类型	属性描述
count_reset_condition	*string*	
cond_if_cond	*string*	
cond_then_expr	*string*	
cond_else_expr	*string*	
formula_measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	此属性可用于定义与派生字段关联的测量。可以向 setter 函数传递一个字符串或其中一个 MeasureType 值。getter 函数将始终返回 MeasureType 值。
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	对于收集字段（深度为 0 的列表），此属性定义与基础值关联的测量类型。
geo_type	Point MultiPoint LineString MultiLineString 多边形 MultiPolygon	对于地理空间字段，此属性定义该字段表示的地理空间对象的类型。这应该与值的列表深度保持一致
has_coordinate_system	布尔值	对于地理空间字段，此属性定义该字段是否具有坐标系
coordinate_system	*string*	对于地理空间字段，此属性定义该字段的坐标系

ensemblenode 属性

"整体"节点对两个或两个以上模型块进行组合，这样所获得的预测比通过任意一个模型获得的预测更为准确。

示例

```
# Create and configure an Ensemble node
# Use this node with the models in demos\streams\pm_binaryclassifier.str
node = stream.create("ensemble", "My node")
node.setPropertyValue("ensemble_target_field", "response")
node.setPropertyValue("filter_individual_model_output", False)
node.setPropertyValue("flag_ensemble_method", "ConfidenceWeightedVoting")
node.setPropertyValue("flag_voting_tie_selection", "HighestConfidence")
```

表 82. *ensemblenode* 属性.

ensemblenode 属性	数据类型	属性描述
ensemble_target_field	字段	为在整体中使用的所有模型指定目标字段。
filter_individual_model_output	标志	指定是否应抑制各个模型的评分结果。
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting AdjustedPropensityWeightedVoting HighestConfidence AverageRawPropensity AverageAdjustedPropensity	指定用于确定整体评分的方法。仅当选定的目标是标志字段时，才会应用此设置。
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	指定用于确定整体评分的方法。仅当选定的目标是名义字段时，才会应用此设置。
flag_voting_tie_selection	Random HighestConfidence RawPropensity AdjustedPropensity	如果已选定投票方法，那么指定解决结的方法。仅当选定的目标是标志字段时，才会应用此设置。
set_voting_tie_selection	Random HighestConfidence	如果已选定投票方法，那么指定解决结的方法。仅当选定的目标是名义字段时，才会应用此设置。
calculate_standard_error	标志	如果目标字段是连续的，那么缺省情况下将运行标准误差计算，以计算测量值或估算值与真实值之间的差，并显示这些估算值的匹配程度。

fillernode 属性

"填充器"节点用于替换字段值并更改存储。您可以选择基于 CLEM 条件（例如 @BLANK(@FIELD)）的替换值。另外，也可以选择将所有空白值或 Null 值替换为特定值。"填充器"节点经常与"类型"节点配合使用，以替换缺失值。

示例

```
node = stream.create("filler", "My node")
node.setPropertyValue("fields", ["Age"])
node.setPropertyValue("replace_mode", "Always")
node.setPropertyValue("condition", "(\"Age\" > 60) and (\"Sex\" = \"M\"")
node.setPropertyValue("replace_with", "\"old man\"")
```

表 83. *fillernode* 属性

fillernode 属性	数据类型	属性描述
fields	列表	数据集中其值将被检查并替换的字段。

表 *83. fillernode* 属性 *(续)*

fillernode 属性	**数据类型**	**属性描述**
replace_mode	Always Conditional Blank Null BlankAndNull	可以替换所有值、空白值或空值，也可以根据指定条件进行替换。
condition	*string*	
replace_with	*string*	

filternode 属性

"过滤"节点用于过滤（废弃）字段、对字段进行重命名以及将字段从一个源节点映射到另一个节点。

示例

```
node = stream.create("filter", "My node")
node.setPropertyValue("default_include", True)
node.setKeyedPropertyValue("new_name", "Drug", "Chemical")
node.setKeyedPropertyValue("include", "Drug", False)
```

使用 default_include 属性。请注意，设置 default_include 属性的值不会自动包括或排除所有字段，而只是确定针对当前所选字段的缺省行为。在功能上，此属性相当于单击"过滤节点"对话框中的**缺省情况下包括字段**按钮。例如，假设运行以下脚本：

```
node = modeler.script.stream().create("filter", "Filter")
node.setPropertyValue("default_include", False)
# Include these two fields in the list
for f in ["Age", "Sex"]:
    node.setKeyedPropertyValue("include", f, True)
```

这会使节点传递字段 年龄 和 性别 ，而丢弃其他所有字段。现在，假设再次运行相同脚本但指定两个不同字段：

```
node = modeler.script.stream().create("filter", "Filter")
node.setPropertyValue("default_include", False)
# Include these two fields in the list
for f in ["BP", "Na"]:
    node.setKeyedPropertyValue("include", f, True)
```

此时会在过滤器中再添加两个字段，因此总共传递四个字段（ 年龄 、 性别 、 *BP* 、 *Na* ）。换句话说，将 default_include 的值重新设置为 False 不会自动重新设置所有字段。

此外，如果现在通过使用脚本或在"过滤节点"对话框中将 default_include 的值更改为 True，则会使此行为发生颠倒，即，将丢弃而非包括上面列出的四个字段。如果有疑问，可使用"过滤节点"对话框中的控件进行实验，这将有助于理解此交互效应。

表 84. *filternode* 属性

filternode 属性	**数据类型**	**属性描述**
default_include	*flag*	用于指定默认行为是传递还是过滤字段的键控属性： 注意，设置此属性不会自动包括或排除所有字段；它只确定默认情况下是包括还是排除选定字段。有关其他注释请参阅下面的示例。
include	*flag*	这是用于包括和除去字段的键控属性。
new_name	*string*	

historynode 属性

"历史记录"节点创建新字段，这些字段包含先前记录中的字段的数据。"历史记录"节点最常用于顺序数据，例如时间序列数据。在使用"历史记录"节点前，可以使用"排序"节点对数据进行排序。

示例

```
node = stream.create("history", "My node")
node.setPropertyValue("fields", ["Drug"])
node.setPropertyValue("offset", 1)
node.setPropertyValue("span", 3)
node.setPropertyValue("unavailable", "Discard")
node.setPropertyValue("fill_with", "undef")
```

表 85. *historynode* 属性

historynode 属性	**数据类型**	**属性描述**
fields	列表	需要其历史记录的字段。
偏移	*number*	指定要从中提取历史记录字段值的最新记录（当前记录之前的记录）。
span	*number*	指定要从中提取值的以前记录的数目。
unavailable	Discard Leave Fill	处理不含历史记录值的记录时，通常参考没有以前的记录作为历史记录的前几个记录（位于数据集顶部）。
fill_with	字符串 Number	指定要用于无历史记录值可用的记录的值或字符串。

partitionnode 属性

分区节点可生成分区字段，该字段可将数据分割为单独的子集以便在模型构建的训练、测试和验证阶段使用。

示例

```
node = stream.create("partition", "My node")
node.setPropertyValue("create_validation", True)
node.setPropertyValue("training_size", 33)
node.setPropertyValue("testing_size", 33)
node.setPropertyValue("validation_size", 33)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 123)
node.setPropertyValue("value_mode", "System")
```

表 *86. partitionnode* 属性

partitionnode 属性	**数据类型**	**属性描述**
new_name	*string*	由节点生成的分区字段的名称。
create_validation	*flag*	指定是否应创建验证分区。
training_size	*integer*	要分配给训练分区的记录所占的百分比 (0-100)。
testing_size	*integer*	要分配给测试分区的记录所占的百分比 (0-100)。
validation_size	*integer*	要分配给验证分区的记录所占的百分比 (0-100)。如果未创建验证分区，那么忽略此属性。
training_label	*string*	训练分区的标签。
testing_label	*string*	测试分区的标签。
validation_label	*string*	验证分区的标签。如果未创建验证分区，那么忽略此属性。
value_mode	System SystemAndLabel Label	指定用于表示数据中每个分区的值。例如，训练样本可以表示为系统整数 1、标签 Training 或二者的组合 1_Training。
set_random_seed	布尔值	指定是否应使用用户指定的随机种子。
random_seed	*integer*	用户指定的随机种子值。如果要使用此值，set_random_seed 必须设置为 True。
enable_sql_generation	布尔值	指定是否使用 SQL 回送以分配记录到分区。
unique_field		指定输入字段，用以确保以随机但可重复的方式将记录分配到分区。如果要使用此值，enable_sql_generation 必须设置为 True。

reclassifynode 属性

"重新分类"节点将一组分类值转换为另一组值。对于折叠类别或者进行数据重新分组以执行分析而言，重新分类非常有用。

示例

```
node = stream.create("reclassify", "My node")
node.setPropertyValue("mode", "Multiple")
node.setPropertyValue("replace_field", True)
node.setPropertyValue("field", "Drug")
node.setPropertyValue("new_name", "Chemical")
node.setPropertyValue("fields", ["Drug", "BP"])
node.setPropertyValue("name_extension", "reclassified")
node.setPropertyValue("add_as", "Prefix")
```

```
node.setKeyedPropertyValue("reclassify", "drugA", True)
node.setPropertyValue("use_default", True)
node.setPropertyValue("default", "BrandX")
node.setPropertyValue("pick_list", ["BrandX", "Placebo", "Generic"])
```

表 *87. reclassifynode* 属性

reclassifynode 属性	**数据类型**	**属性描述**
mode	Single Multiple	Single 对一个字段的类别进行重新分类。Multiple 将激活用于同时对多个字段进行转换的选项。
replace_field	*flag*	
字段	*string*	仅在 Single 模式下使用。
new_name	*string*	仅在 Single 模式下使用。
fields	*[field1 field2 ... fieldn]*	仅在"多重"方式下使用。
name_extension	*string*	仅在"多重"方式下使用。
add_as	Suffix Prefix	仅在"多重"方式下使用。
reclassify	*string*	字段值的结构化属性。
use_default	*flag*	使用缺省值。
default	*string*	指定缺省值。
pick_list	*[string string ... string]*	允许用户导入已知新值的列表以填充表中的下拉列表。

reordernode 属性

"字段重新排序器"节点定义用于显示下游字段的自然顺序。此顺序将影响字段在各种位置（例如表、列表和字段选择器）的显示方式。处理宽数据集时，此操作有助于使所需字段更为直观。

示例

```
node = stream.create("reorder", "My node")
node.setPropertyValue("mode", "Custom")
node.setPropertyValue("sort_by", "Storage")
node.setPropertyValue("ascending", False)
node.setPropertyValue("start_fields", ["Age", "Cholesterol"])
node.setPropertyValue("end_fields", ["Drug"])
```

表 *88. reordernode* 属性

reordernode 属性	**数据类型**	**属性描述**
mode	Custom Auto	可以自动对值进行排序，也可以指定定制顺序。
sort_by	Name Type 存储	
ascending	*flag*	

表 88. reordernode 属性（续）

reordernode 属性	数据类型	属性描述
start_fields	*[field1 field2 ... fieldn]*	新字段插入到这些字段之后。
end_fields	*[field1 field2 ... fieldn]*	新字段插入到这些字段之前。

reprojectnode 属性

在 SPSS Modeler 中，表达式构建器空间函数、空间时间预测 (STP) 节点及地图可视化节点等项将使用投影坐标系。使用重新投影节点可以更改您导入的使用地理坐标系的任何数据的坐标系。

表 89. reprojectnode 属性

reprojectnode 属性	数据类型	属性描述
reproject_fields	*[field1 field2 ... fieldn]*	列出所有要重新投影的字段。
reproject_type	Streamdefault Specify	选择如何对字段进行重新投影。
coordinate_system	*string*	要对字段应用的坐标系的名称。示例： set reprojectnode.coordinate_system = "WGS_1984_World_Mercator"

restructurenode 属性

"重构"节点将名义字段或标志字段转换为一组字段，这组字段可以使用另一字段的值进行填充。例如，给定一个名为 支付类型的字段，其值为 贷方、现金和 借方，那么将创建三个新字段（贷方、现金、借方），每个字段可能包含实际支付的值。

示例

```
node = stream.create("restructure", "My node")
node.setKeyedPropertyValue("fields_from", "Drug", ["drugA", "drugX"])
node.setPropertyValue("include_field_name", True)
node.setPropertyValue("value_mode", "OtherFields")
node.setPropertyValue("value_fields", ["Age", "BP"])
```

表 90. restructurenode 属性

restructurenode 属性	数据类型	属性描述
fields_from	[*category category category*] all	
include_field_name	*flag*	指示是否在重新结构化的字段名中使用字段名。
value_mode	OtherFields Flags	表示用于为重新结构化字段指定值的模式。如果选择 OtherFields，必须指定要使用哪些字段（参阅下文）。如果选择 Flags，那么值为数值标志。

表 90. *restructurenode* 属性（续）

restructurenode 属性	数据类型	属性描述
value_fields	列表	如果 value_mode 是 OtherFields，那么此属性是必需的。指定使用哪些字段作为值字段。

rfmanalysisnode 属性

通过近因、频率和货币 (RFM) 分析节点，您可以检查客户最近一次购买您产品或服务的时间（近因）、客户购买的频率 （频率）以及客户支付的所有交易金额（货币），确定可能成为最佳客户的数量。

示例

```
node = stream.create("rfmanalysis", "My node")
node.setPropertyValue("recency", "Recency")
node.setPropertyValue("frequency", "Frequency")
node.setPropertyValue("monetary", "Monetary")
node.setPropertyValue("tied_values_method", "Next")
node.setPropertyValue("recalculate_bins", "IfNecessary")
node.setPropertyValue("recency_thresholds", [1, 500, 800, 1500, 2000, 2500])
```

表 91. *rfmanalysisnode* 属性

rfmanalysisnode 属性	数据类型	属性描述
recency	字段	指定近因字段。它有可能是日期、时间戳记或简单的数值。
frequency	字段	指定频率字段。
monetary	字段	指定货币字段。
recency_bins	*integer*	指定要生成的近因分箱数量。
recency_weight	*number*	指定应用于近因数据的权重。缺省值为 100。
frequency_bins	*integer*	指定要生成的频率分箱数量。
frequency_weight	*number*	指定应用于频率数据的权重。缺省值为 10。
monetary_bins	*integer*	指定要生成的货币分箱数量。
monetary_weight	*number*	指定应用于货币数据的权重。缺省值为 1。
tied_values_method	Next Current	指定要输入的分箱结值数据。
recalculate_bins	Always IfNecessary	
add_outliers	*flag*	只有当 recalculate_bins 设为 IfNecessary 时才可用。如果已设置，那么将位于下限分箱以下的记录添加到下限分箱中，并且将最高分箱以上的记录添加到最高分箱中。
binned_field	Recency Frequency Monetary	

表 *91. rfmanalysisnode* 属性 *(续)*

rfmanalysisnode 属性	数据类型	属性描述
recency_thresholds	*value value*	只有当 recalculate_bins 设为 Always 时才可用。指定近因分箱的下限阈值和上限阈值。一个分箱的上限阈值用作下一个分箱的下限阈值 例如，[10 30 60] 可定义两个分箱，第一个分箱的上限阈值和下限阈值分别为 10 和 30，第二个分箱的两个阈值分别为 30 和 60。
frequency_thresholds	*value value*	只有当 recalculate_bins 设为 Always 时才可用。
monetary_thresholds	*value value*	只有当 recalculate_bins 设为 Always 时才可用。

settoflagnode 属性

"设为标志"节点根据针对一个或多个名义字段定义的分类值派生多个标志字段。

示例

```
node = stream.create("settoflag", "My node")
node.setKeyedPropertyValue("fields_from", "Drug", ["drugA", "drugX"])
node.setPropertyValue("true_value", "1")
node.setPropertyValue("false_value", "0")
node.setPropertyValue("use_extension", True)
node.setPropertyValue("extension", "Drug_Flag")
node.setPropertyValue("add_as", "Suffix")
node.setPropertyValue("aggregate", True)
node.setPropertyValue("keys", ["Cholesterol"])
```

表 *92. settoflagnode* 属性

settoflagnode 属性	数据类型	属性描述
fields_from	[*category category category*] all	
true_value	*string*	指定设置标志时节点所使用的真值。缺省值为 T。
false_value	*string*	指定设置标志时节点所使用的假值。缺省值为 F。
use_extension	*flag*	使用扩展名作为新标志字段的后缀或前缀。
extension	*string*	
add_as	Suffix Prefix	指定所添加的扩展名是后缀还是前缀。
aggregate	*flag*	根据键字段将记录分组。如果有任何记录被设置为 true ，那么会启用组中的所有标志字段。
keys	列表	键字段。

statisticstransformnode 属性

"Statistics 转换"节点针对 IBM SPSS Modeler 中的数据源运行选择的 IBM SPSS Statistics 语法命令。此节点需要 IBM SPSS Statistics 的许可副本。

有关此节点属性的信息，请参阅第 315 页的『statisticstransformnode 属性』。

timeintervalsnode 属性（不推荐）

注：此节点在 SPSS Modeler V18 中已不推荐，并且替换为新"时间序列"节点。"时间区间"节点指定区间，创建用于对时间序列数据 进行建模的标签（如果需要）。如果各个值的间隔不均匀，那么此节点可以根据需要填充值或者将值汇总，以使记录之间的区间均匀。

示例

```
node = stream.create("timeintervals", "My node")
node.setPropertyValue("interval_type", "SecondsPerDay")
node.setPropertyValue("days_per_week", 4)
node.setPropertyValue("week_begins_on", "Tuesday")
node.setPropertyValue("hours_per_day", 10)
node.setPropertyValue("day_begins_hour", 7)
node.setPropertyValue("day_begins_minute", 5)
node.setPropertyValue("day_begins_second", 17)
node.setPropertyValue("mode", "Label")
node.setPropertyValue("year_start", 2005)
node.setPropertyValue("month_start", "January")
node.setPropertyValue("day_start", 4)
node.setKeyedPropertyValue("pad", "AGE", "MeanOfRecentPoints")
node.setPropertyValue("agg_mode", "Specify")
node.setPropertyValue("agg_set_default", "Last")
```

表 *93. timeintervalsnode* 属性.

timeintervalsnode 属性	数据类型	属性描述
interval_type	None Periods CyclicPeriods Years Quarters Months DaysPerWeek DaysNonPeriodic HoursPerDay HoursNonPeriodic MinutesPerDay MinutesNonPeriodic SecondsPerDay SecondsNonPeriodic	

表 *93. timeintervalsnode* 属性 *(续).*

timeintervalsnode 属性	**数据类型**	**属性描述**
mode	Label Create	指定是要连续标记记录还是要根据指定日期、时间戳记或时间字段构建序列。
field	字段	当根据数据构建序列时，指定表示每个记录的日期或时间的字段。
period_start	*integer*	指定周期或循环周期的起始区间
cycle_start	*integer*	循环周期的起始周期。
year_start	*integer*	对于适用的区间类型，指第一个区间所属的年份。
quarter_start	*integer*	对于适用的区间类型，指第一个区间所属的季度。
month_start	January February March April May June July August September October November December	
day_start	*integer*	
hour_start	*integer*	
minute_start	*integer*	
second_start	*integer*	
periods_per_cycle	*integer*	对于循环周期，指每个周期中的期间数。
fiscal_year_begins	January February March April May June July August September October November December	对于季度区间，指定财政年度开始的月份。
week_begins_on	Sunday Monday Tuesday Wednesday Thursday Friday Saturday Sunday	对于周期性区间（一周中的天、一天中的小时、一天中的分钟和一天中的秒），指定一周开始的那一天。

表 *93. timeintervalsnode* 属性 *(续)*.

timeintervalsnode 属性	**数据类型**	**属性描述**
day_begins_hour	*integer*	对于周期性区间（一天中的小时、一天中的分钟和一天中的秒），指定一天开始的小时。可以与 day_begins_minute 和 day_begins_second 结合起来使用，以指定一个准确时间，例如 *8:05:01*。请参见下面的使用示例。
day_begins_minute	*integer*	对于周期性区间（一天中的小时、一天中的分钟和一天中的秒），指定一天开始的分钟（例如 *8:05*中的 *5*）。
day_begins_second	*integer*	对于周期性区间（一天中的小时、一天中的分钟和一天中的秒），指定一天开始的秒（例如 *8:05:17* 中的 *17*）。
days_per_week	*integer*	对于周期性区间（一周中的天、一天中的小时、一天中的分钟和一天中的秒），指定一周中的天数。
hours_per_day	*integer*	对于周期性区间（一天中的小时、一天中的分钟和一天中的秒），指定一天中的小时数。
interval_increment	1 2 3 4 5 6 10 15 20 30	对于一天中的分钟和一天中的秒，指定为每个记录增加的分钟数或秒数。
field_name_extension	字符串	
field_name_extension_as_prefix	标志	

表 *93. timeintervalsnode* 属性 *(续).*

timeintervalsnode 属性	**数据类型**	**属性描述**
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	
aggregate	Mean Sum Mode Min Max First Last TrueIfAnyTrue	指定字段的汇总方法。
pad	Blank MeanOfRecentPoints true False	指定字段的填充方法。
agg_mode	All Specify	指定是根据需要使用缺省函数汇总或填充所有字段，还是指定要使用的字段和函数。

表 93. *timeintervalsnode* 属性 *(续)*.

timeintervalsnode 属性	**数据类型**	**属性描述**
agg_range_default	Mean Sum Mode Min Max	指定汇总连续字段时要使用的缺省函数。
agg_set_default	Mode First Last	指定汇总名义字段时要使用的缺省函数。
agg_flag_default	TrueIfAnyTrue Mode First Last	
pad_range_default	Blank MeanOfRecentPoints	指定填充连续字段时要使用的缺省函数。
pad_set_default	Blank MostRecentValue	
pad_flag_default	Blank true False	
max_records_to_create	*integer*	指定填充序列时要创建的最大记录数。
estimation_from_beginning	标志	
estimation_to_end	标志	
estimation_start_offset	*integer*	
estimation_num_holdouts	*integer*	
create_future_records	标志	
num_future_records	*integer*	
create_future_field	标志	
future_field_name	字符串	

transposenode 属性

"转置"节点交换行和列中的数据，以便记录变成字段，字段变成记录。

示例

```
node = stream.create("transpose", "My node")
node.setPropertyValue("transposed_names", "Read")
node.setPropertyValue("read_from_field", "TimeLabel")
node.setPropertyValue("max_num_fields", "1000")
node.setPropertyValue("id_field_name", "ID")
```

表 *94. transposenode* 属性

transposenode 属性	**数据类型**	**属性描述**
transpose_method	*enum*	指定变换方法：正常 (normal)，CASE 到 VAR (casetovar) 或 VAR 到 CASE (vartocase)。
transposed_names	Prefix Read	"正常"变换方法的属性。可以根据指定前缀自动生成新字段名，也可以从数据的现有字段中读取新字段名。
prefix	*string*	"正常"变换方法的属性。
num_new_fields	*integer*	"正常"变换方法的属性。使用前缀时，指定要创建的新字段的最大数目。
read_from_field	字段	"正常"变换方法的属性。从中读取名称的字段。此字段必须是一个实例化字段，否则执行节点时将出错。
max_num_fields	*integer*	"正常"变换方法的属性。当从某个字段中读取名称时，指定上限以避免创建过大的字段数。
transpose_type	数字 字符串 Custom	"正常"变换方法的属性。缺省情况下，只能转置连续（数字范围）字段，但也可以选择数字字段的定制子集或转置所有字符串字段。
transpose_fields	列表	"正常"变换方法的属性。指定使用定制选项时转置的字段。
id_field_name	字段	"正常"变换方法的属性。
index	字段	CASE 到 VAR (casetovar) 变换方法的属性。接受使用多个字段作为索引字段。 field1 ... fieldN
column	字段	CASE 到 VAR (casetovar) 变换方法的属性。接受使用多个字段作为列字段。 field1 ... fieldN
value	字段	CASE 到 VAR (casetovar) 变换方法的属性。接受使用多个字段作为值字段。 field1 ... fieldN
id_variables	字段	VAR 到 CASE (vartocase) 变换方法的属性。接受使用多个字段作为标识变量字段。 field1 ... fieldN
value_variables	字段	VAR 到 CASE (vartocase) 变换方法的属性。接受使用多个字段作为值变量字段。 field1 ... fieldN

typenode 属性

"类型"节点指定字段元数据和属性。例如，您可以指定每个字段的测量级别（连续、名义、有序或标志）、设置用于处理缺失值和系统 Null 值的选项、设置用于建模的字段的角色、指定字段标签和值标签以及为字段指定值。

示例

```
node = stream.createAt("type", "My node", 50, 50)
node.setKeyedPropertyValue("check", "Cholesterol", "Coerce")
node.setKeyedPropertyValue("direction", "Drug", "Input")
node.setKeyedPropertyValue("type", "K", "Range")
node.setKeyedPropertyValue("values", "Drug", ["drugA", "drugB", "drugC", "drugD", "drugX",
 "drugY", "drugZ"])
node.setKeyedPropertyValue("null_missing", "BP", False)
node.setKeyedPropertyValue("whitespace_missing", "BP", False)
node.setKeyedPropertyValue("description", "BP", "Blood Pressure")
node.setKeyedPropertyValue("value_labels", "BP", [["HIGH", "High Blood Pressure"],
 ["NORMAL", "normal blood pressure"]])
```

注意，某些情况下可能需要完全实例化"类型"节点才能使其他节点正常运行，例如，"设为标志"节点的 fields from 属性。可以只连接"表"节点并执行该节点以实例化这些字段：

```
tablenode = stream.createAt("table", "Table node", 150, 50)
stream.link(node, tablenode)
tablenode.run(None)
stream.delete(tablenode)
```

表 *95. typenode* 属性.

typenode 属性	数据类型	属性描述
direction	Input Target Both None Partition Split Frequency RecordID	字段角色的键控属性。 **注：** 现在，不推荐使用值 In 和 Out。在将来的版本中可能取消对这些值的支持。
type	范围 (Range) Flag Set Typeless Discrete OrderedSet Default	字段的测量级别（以前称为字段的"类型"）。 将 type 设置为 Default 会清除所有 values 参数 设置，并且如果 value_mode 的值为 Specify，那么它将重置为 Read。如果 value_mode 设置为 Pass 或 Read， 那么设置 type 不会影响 value_mode。 **注：**内部使用的数据类型不同于类型节点中显示的类型。对应关系如下所示：Range -> Continuous Set - > Nominal OrderedSet -> Ordinal Discrete- > Categorical
storage	未知 字符串 整数 实数 Time 日期 Timestamp	字段存储类型的只读键控属性。

表 *95. typenode* 属性 *(续).*

typenode 属性	数据类型	属性描述
check	None Nullify Coerce Discard Warn Abort	字段类型和范围检查的键控属性。
values	[*value value*]	对于连续型字段而言，第一个是最小值，后一个是最大值。对于名义字段，指定所有值。对标志字段而言，第一个值代表 *false*，后一个值代表 *true*。设置该属性将自动把 value_mode 属性设置为 Specify。
value_mode	Read Pass Read+ Current Specify	确定值的设置方式。注意，不能将此属性直接设置为 Specify；要使用特定值，需设置 values 属性。
extend_values	*flag*	当 value_mode 设置为 Read 时将应用。设为 T 则将新读取的值添加到任意现有字段值。设置为 F 则丢弃现有值并添加新读取值。
enable_missing	*flag*	当设置为 T 时，那么激活对字段缺失值的跟踪。
missing_values	[*value value ...*]	指定表示缺失数据的数据值。
range_missing	*flag*	指定是否为字段定义缺失值（空白）范围。
missing_lower	*string*	range_missing 为 true 时，此属性指定缺失值范围的下限。
missing_upper	*string*	range_missing 为 true 时，此属性指定缺失值范围的上限。
null_missing	*flag*	设置为 T 时，空（在软件中显示为 $null$ 的未定义值）被视为缺失值。
whitespace_ missing	*flag*	设置为 T 时，仅包含空白（空格、制表符和换行符）的值被视为缺失值。
description	*string*	为字段指定说明。
value_labels	*[[Value LabelString] [Value LabelString] ...]*	用于为值对指定标签。
display_places	*integer*	为字段设置显示的小数位数（仅用于以 REAL 存储的字段）。值为 -1 时，将使用流缺省值。
export_places	*integer*	为字段设置导出时的小数位数（仅用于以 REAL 存储的字段）。值为 -1 时，将使用流缺省值。
decimal_separator	DEFAULT PERIOD COMMA	为字段设置十进制分隔符（仅用于以 REAL 存储的字段）。

表 *95. typenode* 属性 *(续).*

typenode 属性	数据类型	属性描述
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	为字段设置日期格式（仅用于以 DATE 或 TIMESTAMP 存储的字段）。
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	为字段设置时间格式（仅用于以 TIME 或 TIMESTAMP 存储的字段）。
number_format	DEFAULT STANDARD SCIENTIFIC CURRENCY	为字段设置数字显示格式。
standard_places	*integer*	为字段设置以标准格式显示时的小数位数。值为 -1 时，将使用流缺省值。请注意，现有的 display_places 通道也会更改此属性，但目前已不再使用。
scientific_places	*integer*	为字段设置以科学计数格式显示时的小数位数。值为 -1 时，将使用流缺省值。
currency_places	*integer*	为字段设置以货币格式显示时的小数位数。值为 -1 时，将使用流缺省值。

表 *95. typenode* 属性 *(续).*

typenode 属性	数据类型	属性描述
grouping_symbol	DEFAULT NONE LOCALE PERIOD COMMA SPACE	为字段设置分组符号。
column_width	*integer*	为字段设置列宽度。值为 -1 标识将列宽度设置为 Auto。
justify	AUTO CENTER LEFT RIGHT	为字段设置列对齐格式。
measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	此键控属性类似于 type，因为它可用于定义与字段关联的测量。区别在于，还可以向 Python 脚本编制中的 setter 函数传递其中一个 MeasureType 值，而 getter 始终返回 MeasureType 值。
collection_ measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	对于收集字段（深度为 0 的列表），此键控属性定义与基础值关联的测量类型。
geo_type	Point MultiPoint LineString MultiLineString 多边形 MultiPolygon	对于地理空间字段，此键控属性定义该字段表示的地理空间对象的类型。这应该与值的列表深度保持一致。
has_coordinate_system	布尔值	对于地理空间字段，此属性定义该字段是否具有坐标系
coordinate_system	*string*	对于地理空间字段，此键控属性定义该字段的坐标系。
custom_storage_type	Unknown / MeasureType.UNKNOWN String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP List / MeasureType.LIST	此键控属性类似于 custom_storage，因为它可用于定义字段的覆盖存储。区别在于，还可以向 Python 脚本编制中的 setter 函数传递其中一个 StorageType 值，而 getter 始终返回 StorageType 值。

表 *95. typenode* 属性 *(续).*

typenode 属性	**数据类型**	**属性描述**
custom_list_storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	对于列表字段，此键控属性指定基础值的存储类型。
custom_list_depth	*integer*	对于列表字段，此键控属性指定字段的深度
max_list_length	*integer*	仅可用于测量级别为地理空间或集合的数据。指定列表可包含的元素数量以设置列表的最大长度。
max_string_length	*integer*	仅可用于无类型数据，生成 SQL 以创建表时使用。输入数据中最大字符串的值；这将在表中生成一个大小足够包含该字符串的列。

第 12 章 图形节点属性

图形节点公共属性

本节介绍图形节点的可用属性，包括公共属性和每种节点类型特有的属性。

表 *96.* 公共图形节点属性

公共图形节点属性	数据类型	属性描述
title	*string*	指定标题。示例："This is a title."
caption	*string*	指定文字说明。示例："This is a caption."
output_mode	Screen File	指定是显示图形节点的输出还是将其写到文件中。
output_format	BMP JPEG PNG HTML output (*.cou*)	指定输出类型。允许每个节点使用的确切输出类型是不同的。
full_filename	*string*	为从图形节点生成的输出指定目标路径和文件名。
use_graph_size	*flag*	控制是否使用下面的宽度和高度属性明确调整图形的大小。只影响输出到屏幕的图形。不适用于"分布"节点。
graph_width	*number*	当 use_graph_size 为 True 时，以像素为单位设置图形宽度。
graph_height	*number*	当 use_graph_size 为 True 时，以像素为单位设置图形高度。

关闭可选字段

通过将属性值设置为 " "（空字符串），可以将可选字段（如图的重叠字段）关闭，如下以示例所示：

```
plotnode.setPropertyValue("color_field", "")
```

指定颜色

使用十六进制字符串（从井号 (#) 开始），可指定标题、标注、背景和标签的颜色。例如，要将图形背景设置为天蓝色，请使用以下语句：

```
mygraphnode.setPropertyValue("graph_background", "#87CEEB")
```

此处，前两位数 87 指定红色内容；中间两位数 CE 指定绿色内容；最后两位数 EB 指定蓝色内容。每位数可获取一个位于范围 0-9 或 A-F 内的值。这些值在一起可以指定红-绿-蓝（即 RGB）颜色。

注：以 RGB 形式指定颜色时，可以使用用户界面中的字段选择器确定正确的颜色代码。只需将鼠标停留在颜色上面就可激活含所需信息的工具提示。

collectionnode 属性

"收集"节点显示一个数字字段的值相对于另一个数字字段的值的分布。（它创建类似于直方图的图形。）图示说明值不断变化的变量或字段时，它 是有用的。使用 3-D 图形表示时，还可以包括一个按类别显示分布的符号轴。

示例

```
node = stream.create("collection", "My node")
# "Plot" tabnode.setPropertyValue("three_D", True)
node.setPropertyValue("collect_field", "Drug")
node.setPropertyValue("over_field", "Age")
node.setPropertyValue("by_field", "BP")
node.setPropertyValue("operation", "Sum")
# "Overlay" sectionnode.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# "Options" tabnode.setPropertyValue("range_mode", "Automatic")
node.setPropertyValue("range_min", 1)
node.setPropertyValue("range_max", 100)
node.setPropertyValue("bins", "ByNumber")
node.setPropertyValue("num_bins", 10)
node.setPropertyValue("bin_width", 5)
```

表 *97. collectionnode* 属性

collectionnode 属性	**数据类型**	**属性描述**
over_field	字段	
over_label_auto	*flag*	
over_label	*string*	
collect_field	字段	
collect_label_auto	*flag*	
collect_label	*string*	
three_D	*flag*	
by_field	字段	
by_label_auto	*flag*	
by_label	*string*	
operation	Sum Mean Min Max SDev	
color_field	*string*	
panel_field	*string*	
animation_field	*string*	
range_mode	Automatic UserDefined	
range_min	*number*	
range_max	*number*	

表 *97. collectionnode* 属性 *(续)*

collectionnode 属性	数据类型	属性描述
bins	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	
use_grid	*flag*	
graph_background	颜色	本节在开头处介绍了标准图形颜色。
page_background	颜色	本节在开头处介绍了标准图形颜色。

distributionnode 属性

"分布"节点显示符号（分类）值（例如抵押类型或性别）的出现次数。通常，您可以使用"分布"节点来显示数据中的不平衡，然后可以在创建模型前使用"均衡"节点来纠正此类不均衡。

示例

```
node = stream.create("distribution", "My node")
# "Plot" tabnode.setPropertyValue("plot", "Flags")
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("normalize", True)
node.setPropertyValue("sort_mode", "ByOccurence")
node.setPropertyValue("use_proportional_scale", True)
```

表 *98. distributionnode* 属性

distributionnode 属性	数据类型	属性描述
plot	SelectedFields Flags	
x_field	字段	
color_field	字段	交叠字段。
normalize	*flag*	
sort_mode	ByOccurence 字母顺序	
use_proportional_scale	*flag*	

evaluationnode 属性

"评估"节点有助于评估和比较预测模型。评估图表显示模型预测特定结果的优劣程度。它根据预测值和预测置信度对记录进行排序。它将记录分成若干个相同大小的组（**分位数**），然后从高到底为每个分位数划分业务标准值。在散点图中，以不同的线条显示多个模型。

示例

```
node = stream.create("evaluation", "My node")
# "Plot" tab
node.setPropertyValue("chart_type", "Gains")
node.setPropertyValue("cumulative", False)
node.setPropertyValue("field_detection_method", "Name")
node.setPropertyValue("inc_baseline", True)
node.setPropertyValue("n_tile", "Deciles")
node.setPropertyValue("style", "Point")
node.setPropertyValue("point_type", "Dot")
node.setPropertyValue("use_fixed_cost", True)
node.setPropertyValue("cost_value", 5.0)
node.setPropertyValue("cost_field", "Na")
node.setPropertyValue("use_fixed_revenue", True)
node.setPropertyValue("revenue_value", 30.0)
node.setPropertyValue("revenue_field", "Age")
node.setPropertyValue("use_fixed_weight", True)
node.setPropertyValue("weight_value", 2.0)
node.setPropertyValue("weight_field", "K")
```

表 *99. evaluationnode* 属性.

evaluationnode 属性	**数据类型**	**属性说明**
chart_type	Gains Response Lift Profit ROI ROC	
inc_baseline	*flag*	
field_detection_method	Metadata Name	
use_fixed_cost	*flag*	
cost_value	数字	
cost_field	*string*	
use_fixed_revenue	*flag*	
revenue_value	数字	
revenue_field	*string*	
use_fixed_weight	*flag*	
weight_value	数字	
weight_field	字段	
n_tile	Quartiles Quintles Deciles Vingtiles 百分位 (Percentiles) 1000-tiles	
cumulative	*flag*	
style	Line Point	

表 *99. evaluationnode* 属性 *(续).*

evaluationnode 属性	数据类型	属性说明
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
export_data	*flag*	
data_filename	字符串	
delimiter	字符串	
new_line	*flag*	
inc_field_names	*flag*	
inc_best_line	*flag*	
inc_business_rule	*flag*	
business_rule_condition	*string*	
plot_score_fields	*flag*	
score_fields	*[field1 ... fieldN]*	
target_field	字段	
use_hit_condition	*flag*	
hit_condition	*string*	
use_score_expression	*flag*	
score_expression	*string*	
caption_auto	*flag*	

graphboardnode 属性

"图形板"节点在单个节点中提供许多不同类型的图形。使用此节点，可以选择要探索的数据字段，然后从适用于选定数据的字段中选择 一个图形。此节点自动过滤掉所有不适用于字段选项的图形类型。

注：如果您设置对图形类型无效的属性（例如，为直方图指定 y_field），该属性将被忽略。

注：在 UI 中许多不同图形类型的"详细信息"选项卡上，有一个**摘要**字段；脚本编制当前不支持此字段。

示例

```
node = stream.create("graphboard", "My node")
node.setPropertyValue("graph_type", "Line")
node.setPropertyValue("x_field", "K")
node.setPropertyValue("y_field", "Na")
```

表 *100. graphboardnode* 属性

graphboard 属性	**数据类型**	**属性描述**
graph_type	2DDotplot 3DArea 3DBar 3DDensity 3DHistogram 3DPie 3DScatterplot Area ArrowMap Bar BarCounts BarCountsMap BarMap BinnedScatter Boxplot Bubble ChoroplethMeans ChoroplethMedians ChoroplethSums ChoroplethValues ChoroplethCounts CoordinateMap CoordinateChoroplethMeans CoordinateChoroplethMedians CoordinateChoroplethSums CoordinateChoroplethValues CoordinateChoroplethCounts Dotplot Heatmap HexBinScatter Histogram Line LineChartMap LineOverlayMap Parallel Path Pie PieCountMap PieCounts PieMap PointOverlayMap PolygonOverlayMap Ribbon Scatterplot SPLOM Surface	标识图形类型。
x_field	字段	为 *x* 轴指定定制标签。只适用于标签。
y_field	字段	为 *y* 轴指定定制标签。只适用于标签。

表 *100. graphboardnode* 属性 *(续)*

graphboard 属性	**数据类型**	**属性描述**
z_field	字段	用于某些 3-D 图。
color_field	字段	在热图中使用。
size_field	字段	在气泡散点图中使用。
categories_field	字段	
values_field	字段	
rows_field	字段	
columns_field	字段	
fields	字段	
start_longitude_field	字段	与参考图中的箭头配合使用。
end_longitude_field	字段	
start_latitude_field	字段	
end_latitude_field	字段	
data_key_field	字段	用于各种图。
panelrow_field	*string*	
panelcol_field	*string*	
animation_field	*string*	
longitude_field	字段	与图中的坐标配合使用。
latitude_field	字段	
map_color_field	字段	

histogramnode 属性

"直方图"节点显示数字字段的值的出现次数。此节点经常用来在进行数据操作和模型构建前探索数据。与"分布"节点相似，"直方图"节点经常用来揭示数据中的不平衡。

示例

```
node = stream.create("histogram", "My node")
# "Plot" tabnode.setPropertyValue("field", "Drug")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# "Options" tabnode.setPropertyValue("range_mode", "Automatic")
node.setPropertyValue("range_min", 1.0)
node.setPropertyValue("range_max", 100.0)
node.setPropertyValue("num_bins", 10)
node.setPropertyValue("bin_width", 10)
node.setPropertyValue("normalize", True)
node.setPropertyValue("separate_bands", False)
```

表 *101. histogramnode* 属性

histogramnode 属性	**数据类型**	**属性描述**
字段	字段	
color_field	字段	
panel_field	字段	
animation_field	字段	
range_mode	Automatic UserDefined	
range_min	*number*	
range_max	*number*	
bins	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	
normalize	*flag*	
控separate_bands	*flag*	
x_label_auto	*flag*	
x_label	*string*	
y_label_auto	*flag*	
y_label	*string*	
use_grid	*flag*	
graph_background	颜色	本节在开头处介绍了标准图形颜色。
page_background	颜色	本节在开头处介绍了标准图形颜色。
normal_curve	*flag*	指出是否应在输出中显示正态分布曲线。

mapvisualization 属性

"地图可视化"节点可以接受多个输入连接，并在地图上将地理空间数据显示为一系列层。每个层都是单个地理空间字段；例如，底层可能是国家或地区的地图，在其之上可能存在一个道路层、一个河流层和一个城镇层。

表 *102. mapvisualization* 属性

mapvisualization 属性	**数据类型**	**属性描述**
tag	*string*	设置输入的标记名称。缺省标记为基于输入连接到节点的顺序的数字（第一个连接标记为 1，第二个连接标记为 2，以此类推。

表 *102. mapvisualization* 属性 *(续)*

mapvisualization 属性	**数据类型**	**属性描述**
layer_field	字段	选择数据集中哪个地理字段显示为地图上的一个图层。缺省选择基于以下排序顺序： • 首先 - 点 • 线串 • 多边形 • 多点 • 多线串 • 最后 - 多多边形 如果存在两个具有同一测量类型的字段，那么缺省情况下将选中按名称的字母顺序排列的第一个字段。
color_type	布尔值	指定是否将标准颜色应用于地理字段的所有特征，或者指定覆盖字段根据数据集中另一字段的值对各个特征进行着色。可能的值包括 standard 或 overlay。缺省值为 standard。
颜色	*string*	如果针对 color_type 选中 standard，那么下拉列表包含的调色板与用户选项"显示"选项卡上的图表类别颜色顺序相同。 缺省为图表类别颜色 1。
color_field	字段	如果针对 color_type 选中 overlay，那么下拉列表包含与选中作为层的地理字段相同的数据集中的所有字段。
symbol_type	布尔值	指定是否将标准符号应用于地理字段的所有记录，或者指定覆盖符号根据数据集中另一字段的点数更改符号图标。可能的值包括 standard 或 overlay。缺省值为 standard。
symbol	*string*	如果针对 symbol_type 选中 standard，那么下拉列表包含可用于在地图上显示点数的符号选择。
symbol_field	字段	如果针对 symbol_type 选中 overlay，那么下拉列表包含与选中作为层的地理字段相同的数据集中的所有名义、有序或分类字段。
size_type	布尔值	指定是否将标准大小应用于地理字段的所有记录，或者指定覆盖大小根据数据集中另一字段的值更改符号图标或线条厚度大小。可能的值包括 standard 或 overlay。缺省值为 standard。
size	*string*	如果针对 size_type 选中 standard，那么对于 point 或 multipoint，下拉列表包含所选符号的大小选择。对于 linestring 或 multilinestring，下拉列表包含线条厚度的选择。
size_field	字段	如果针对 size_type 选中 overlay，那么下拉列表包含与选中作为层的地理字段相同的数据集中的所有字段。

表 *102. mapvisualization* 属性 *(续)*

mapvisualization 属性	**数据类型**	**属性描述**
transp_type	布尔值	指定是否将标准透明度应用于地理字段的所有记录，，或者指定覆盖透明度根据数据集中另一字段的值更改符号、线条或多边形的透明度级别。可能的值包括 standard 或 overlay。缺省值为 standard。
transp	*integer*	如果针对 transp_type 选中 standard，那么下拉列表包含透明度级别的选择，从 0%（不透明）开始按 10% 增量增加值 100%（透明）。设置地图上点、线条或多边形的透明度。 如果针对 size_type 选中 overlay，那么下拉列表包含与选中作为层的地理字段相同的数据集中的所有字段。 对于 points、multipoints、linestrings 和作为底层的 multilinestrings、polygons 和 multipolygons，缺省值为 0%。对于不属于底层的 polygons 和 multipolygons，缺省值为 50% （以避免导致这些多边形下的层变模糊）。
transp_field	字段	如果针对 transp_type 选中 overlay，那么下拉列表包含与选中作为层的地理字段相同的数据集中的所有字段。
data_label_field	字段	指定在地图上要用作为数据标签的字段。例如，如果此设置应用于的层为多边形层，那么数据标签可能是 name 字段 - 其中包含每个多边形的名称。因此在此选中 name 字段会导致在地图上显示这些名称。
use_hex_binning	布尔值	启用六边形分箱，并启用所有汇总下拉列表。缺省情况下关闭此设置。

表 *102. mapvisualization* 属性 *(续)*

mapvisualization 属性	**数据类型**	**属性描述**
color_aggregation 和 transp_aggregation	*string*	如果您为使用了六边形分箱功能的点层选中了覆盖字段，那么该字段中所有的值都必须针对该六边形内的所有点进行汇总。因此，必须为任何要应用于地图的覆盖字段指定汇总函数。 可用汇总函数为： 连续（实数或整数存储）： • 合计 • 均数 • 最小值 • 最大值 • 中位数 • 第一个四分位 • 第三个四分位 连续（时间、日期或时间戳记存储）： • 均数 • 最小值 • 最大值 名义/分类： • 众数 • 最小值 • 最大值 标志： • True（如果任何项为 true） • False（如果任何项为 false）
custom_storage	*string*	设置字段的总体存储类型。缺省值为 List。如果指定 List，那么禁用以下 custom_value_storage 和 list_depth 控件。
custom_value_storage	*string*	设置列表中的元素的存储类型，而不是字段的整体存储类型。缺省值为 Real。

表 102. *mapvisualization* 属性 (续)

mapvisualization 属性	数据类型	属性描述
list_depth	*integer*	设置列表字段的深度。所需的深度取决于地理字段的类型并遵循下列条件： • 点 - 0 • 线串 - 1 • 多边形 - 2 • 多点 - 1 • 多线串 - 2 • 多多边形 - 3 您必须了解要重新转换为列表的地理空间字段的类型以及该类字段的所需深度。如果设置错误，将无法使用该字段。 缺省值为 0，最小值为 0，最大值为 10。

multiplotnode 属性

"多重散点图"节点创建在单个 *X* 字段上显示多个 *Y* 字段的散点图。*Y* 字段被绘制为彩色的线；每条线相当于"样式"设置为**线**且"X 模式"设置为**排序**的散点图节点。在探索多个变量随时间推移的变化情况时，多重散点图非常有用。

示例

```
node = stream.create("multiplot", "My node")
# "Plot" tabnode.setPropertyValue("x_field", "Age")
node.setPropertyValue("y_fields", ["Drug", "BP"])
node.setPropertyValue("panel_field", "Sex")
# "Overlay" sectionnode.setPropertyValue("animation_field", "")
node.setPropertyValue("tooltip", "test")
node.setPropertyValue("normalize", True)
node.setPropertyValue("use_overlay_expr", False)
node.setPropertyValue("overlay_expression", "test")
node.setPropertyValue("records_limit", 500)
node.setPropertyValue("if_over_limit", "PlotSample")
```

表 103. *multiplotnode* 属性

multiplotnode 属性	数据类型	属性描述
x_field	字段	
y_fields	列表	
panel_field	字段	
animation_field	字段	
normalize	*flag*	
use_overlay_expr	*flag*	
overlay_expression	*string*	
records_limit	*number*	

表 *103. multiplotnode* 属性 *(续)*

multiplotnode 属性	数据类型	属性描述
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*flag*	
x_label	*string*	
y_label_auto	*flag*	
y_label	*string*	
use_grid	*flag*	
graph_background	颜色	本节在开头处介绍了标准图形颜色。
page_background	颜色	本节在开头处介绍了标准图形颜色。

plotnode 属性

散点图节点可显示数字字段间的关系。可通过使用点（散点）或线创建散点图。

示例

```
node = stream.create("plot", "My node")
# "Plot" tabnode.setPropertyValue("three_D", True)
node.setPropertyValue("x_field", "BP")
node.setPropertyValue("y_field", "Cholesterol")
node.setPropertyValue("z_field", "Drug")
# "Overlay" sectionnode.setPropertyValue("color_field", "Drug")
node.setPropertyValue("size_field", "Age")
node.setPropertyValue("shape_field", "")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "BP")
node.setPropertyValue("transp_field", "")
node.setPropertyValue("style", "Point")
# "Output" tabnode.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "JPEG")
node.setPropertyValue("full_filename", "C:/temp/graph_output/plot_output.jpeg")
```

表 *104. plotnode* 属性.

plotnode 属性	数据类型	属性描述
x_field	字段	为 *x* 轴指定定制标签。只适用于标签。
y_field	字段	为 *y* 轴指定定制标签。只适用于标签。
three_D	标志	为 *y* 轴指定定制标签。只适用于 3 维图形中的标签。
z_field	字段	
color_field	字段	交叠字段。
size_field	字段	
shape_field	字段	

表 *104. plotnode* 属性 *(续).*

plotnode 属性	**数据类型**	**属性描述**
panel_field	字段	指定用于为每个类别绘制单独图表的名义字段或标志字段。图表一起平铺在一个输出窗口中。
animation_field	字段	指定以图说明数据值类别（通过使用动画创建一系列按顺序显示的图表来说明）时所使用的名义字段或标志字段。
transp_field	字段	指定以图说明数据值类别（通过为每个类别使用不同级别的透明度来说明）时所使用的字段。不适用于线散点图。
overlay_type	None Smoother Function	指定是显示重叠函数还是 LOESS 平滑器。
overlay_expression	字符串	指定当 overlay_type 设置为 Function 时使用的表达式。
style	Point Line	
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
x_mode	Sort Overlay AsRead	
x_range_mode	Automatic UserDefined	
x_range_min	*number*	
x_range_max	*number*	
y_range_mode	Automatic UserDefined	
y_range_min	*number*	
y_range_max	*number*	
z_range_mode	Automatic UserDefined	
z_range_min	*number*	

表 *104. plotnode* 属性 *(续)*.

plotnode 属性	**数据类型**	**属性描述**
z_range_max	*number*	
jitter	标志	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	标志	
x_label	字符串	
y_label_auto	标志	
y_label	字符串	
z_label_auto	标志	
z_label	字符串	
use_grid	标志	
graph_background	颜色	本节在开头处介绍了标准图形颜色。
page_background	颜色	本节在开头处介绍了标准图形颜色。
use_overlay_expr	标志	该属性已由 overlay_type 替代。

timeplotnode 属性

"时间散点图"节点显示一组或多组时间序列数据。通常情况下，您首先要使用"时间区间"节点创建一个 *TimeLabel* 字段，该字段用于为 *x* 轴设置标签。

示例

```
node = stream.create("timeplot", "My node")
node.setPropertyValue("y_fields", ["sales", "men", "women"])
node.setPropertyValue("panel", True)
node.setPropertyValue("normalize", True)
node.setPropertyValue("line", True)
node.setPropertyValue("smoother", True)
node.setPropertyValue("use_records_limit", True)
node.setPropertyValue("records_limit", 2000)
# Appearance settingsnode.setPropertyValue("symbol_size", 2.0)
```

表 *105. timeplotnode* 属性.

timeplotnode 属性	**数据类型**	**属性描述**
plot_series	序列 Models	
use_custom_x_field	标志	
x_field	字段	
y_fields	列表	
panel	标志	

表 *105. timeplotnode* 属性 *(续).*

timeplotnode 属性	**数据类型**	**属性描述**
normalize	标志	
line	标志	
points	标志	
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
smoother	标志	只有将 panel 设置为 True ，才可将平滑其添加到散点图中。
use_records_limit	标志	
records_limit	*integer*	
symbol_size	*number*	指定符号大小。
panel_layout	Horizontal Vertical	

eplotnode 属性

E-Plot (Beta) 节点可显示数字字段间的关系。它类似于散点图节点，但选项不同，并且其输出使用特定于该节点的新绘图界面。使用 beta 级别节点来试用各种新的绘图功能。

表 *106. eplotnode* 属性

eplotnode 属性	**数据类型**	**属性描述**
x_field	字符串	指定要在水平 X 轴上显示的字段。
y_field	字符串	指定要在垂直 Y 轴上显示的字段。
color_field	字符串	指定要用于表示输出中的颜色图重叠的字段（如果需要）。

表 106. *eplotnode* 属性（续）

eplotnode 属性	**数据类型**	**属性描述**
size_field	字符串	指定要用于表示输出中的大小图重叠的字段（如果需要）。
shape_field	字符串	指定要用于表示输出中的形状图重叠的字段（如果需要）。
interested_fields	字符串	指定要包含在输出中的字段。
records_limit	整数	指定要在输出中绘制的最大记录数。 2000 为缺省值。
if_over_limit	布尔值	指定超过 records_limit 时，是使用样本选项还是使用所有数据选项。样本是缺省值，它随机抽取数据，直到达到 records_limit 为止。如果指定使用所有数据以忽略 records_limit 并绘制所有数据点，请注意这会明显降低性能。

tsnenode 属性

t-分布随机邻近嵌入 (t-SNE) (t-SNE) 是一款用于可视化高维数据的工具。此工具可将数据点的亲缘关系转换为概率。此 t-SNE 节点在 SPSS Modeler 中使用 Python 进行实现并且需要 scikit-learn© Python 库。

表 107. *tsnenode* 属性

tsnenode 属性	**数据类型**	**属性描述**
mode_type	字符串	指定 simple 或 expert 方式。
n_components	字符串	嵌入空间的维度（2D 或 3D）。指定 2 或 3。缺省值为 2。
method	字符串	指定 barnes_hut 或 exact。缺省值为 barnes_hut。
init	字符串	初始化嵌入。指定 random 或 pca。缺省值为 random。
target_field	字符串	目标字段名称。它可为输出图形上的颜色映射图。如果未指定目标字段，那么此图形将使用一种颜色。
perplexity	浮点数	perplexity 与其他流形学习算法中使用的最近相邻元素数相关。通常，数据集越大，所需的 perplexity 也越大。考虑选择 5 到 50 之间的值。缺省值为 30。
early_exaggeration	浮点数	控制原始空间中自然集群在嵌入空间中紧密程度以及集群之间的空间。缺省值为 12.0。
learning_rate	浮点数	缺省值为 200。
n_iter	整数	优化的最大迭代次数。设置为至少 250。缺省值为 1000。

表 *107. tsnenode* 属性 *(续)*

tsnenode 属性	数据类型	属性描述
angle	浮点数	从某个点度量的距离节点的角度大小。指定范围为 0-1 的值。缺省值为 0.5。
enable_random_seed	布尔值	设置为 true 以启用 random_seed 参数。缺省值为 false。
random_seed	整数	要使用的随机数种子。缺省值为 None。
n_iter_without_progress	整数	没有进度的最大迭代次数。缺省值为 300。
min_grad_norm	字符串	如果梯度标准值低于此阈值，那么优化将停止。缺省值为 1.0E-7。可能的值为： • 1.0E-1 • 1.0E-2 • 1.0E-3 • 1.0E-4 • 1.0E-5 • 1.0E-6 • 1.0E-7 • 1.0E-8
isGridSearch	布尔值	设置为 true 以执行具有多个不同复杂度的 t-SNE。缺省值为 false。
output_Rename	布尔值	如果要提供定制名称，请指定 true，或者如果要自动对输出命名，请指定 false。缺省值为 false。
output_to	字符串	指定 Screen 或 Output。缺省值为 Screen。
full_filename	字符串	指定输出文件名。
output_file_type	字符串	输出文件格式。指定 HTML 或 Output object。缺省值为 HTML。

webnode 属性

Web 节点说明两个或两个以上符号（分类）字段的值之间的关系强度。此图使用不同粗细的线条来表示连接强度。例如，您可以使用 Web 节点来探索电子商务网站上一组商品的购买之间的关系。

示例

```
node = stream.create("web", "My node")
# "Plot" tabnode.setPropertyValue("use_directed_web", True)
node.setPropertyValue("to_field", "Drug")
node.setPropertyValue("fields", ["BP", "Cholesterol", "Sex", "Drug"])
node.setPropertyValue("from_fields", ["BP", "Cholesterol", "Sex"])
node.setPropertyValue("true_flags_only", False)
node.setPropertyValue("line_values", "Absolute")
node.setPropertyValue("strong_links_heavier", True)
# "Options" tabnode.setPropertyValue("max_num_links", 300)
node.setPropertyValue("links_above", 10)
```

```
node.setPropertyValue("num_links", "ShowAll")
node.setPropertyValue("discard_links_min", True)
node.setPropertyValue("links_min_records", 5)
node.setPropertyValue("discard_links_max", True)
node.setPropertyValue("weak_below", 10)
node.setPropertyValue("strong_above", 19)
node.setPropertyValue("link_size_continuous", True)
node.setPropertyValue("web_display", "Circular")
```

表 *108. webnode* 属性

webnode 属性	**数据类型**	**属性描述**
use_directed_web	*flag*	
fields	列表	
to_field	字段	
from_fields	列表	
true_flags_only	*flag*	
line_values	Absolute OverallPct PctLarger PctSmaller	
strong_links_heavier	*flag*	
num_links	ShowMaximum ShowLinksAbove ShowAll	
max_num_links	*number*	
links_above	*number*	
discard_links_min	*flag*	
links_min_records	*number*	
discard_links_max	*flag*	
links_max_records	*number*	
weak_below	*number*	
strong_above	*number*	
link_size_continuous	*flag*	
web_display	Circular 网络 Directed Grid	
graph_background	颜色	本节在开头处介绍了标准图形颜色。
symbol_size	*number*	指定符号大小。

第 13 章 建模节点属性

公共建模节点属性

以下属性通用于某些或所有建模节点。所有例外情况均根据需要记录在各个建模节点的文档中。

表 *109.* 公共建模节点属性

属性	值	属性描述
custom_fields	*flag*	如果为 true，那么允许您为当前节点指定目标字段、输入字段和其他字段。如果为 false，那么将使用上游"类型"节点的当前设置。
target 或 targets	字段 或 [*field1 ... fieldN*]	根据模型类型指定一个目标字段或多个目标字段。
inputs	[*field1 ... fieldN*]	模型所使用的输入字段或预测变量字段。
partition	字段	
use_partitioned_data	*flag*	如果定义了分区字段，那么此选项可确保仅训练分区的数据用于构建模型。
use_split_data	*flag*	
splits	*[field1 ... fieldN]*	指定一个或多个用于分割建模的字段。仅在 use_split_data 设置为真时有效。
use_frequency	*flag*	特定模型所使用的权重字段和频率字段（参见每种模型类型的说明）。
frequency_field	字段	
use_weight	*flag*	
weight_field	字段	
use_model_name	*flag*	
model_name	*string*	新模型的定制名称。
mode	Simple Expert	

anomalydetectionnode 属性

Anomaly Detection 节点确定不符合"正常"数据格式的异常观测值（离群值）。即使离群值不匹配任何已知格式或用户不清楚自己的查找对象，也可以使用此节点来确定离群值。

示例

```
node = stream.create("anomalydetection", "My node")
node.setPropertyValue("anomaly_method", "PerRecords")
node.setPropertyValue("percent_records", 95)
```

```
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("peer_group_num_auto", True)
node.setPropertyValue("min_num_peer_groups", 3)
node.setPropertyValue("max_num_peer_groups", 10)
```

表 *110. anomalydetectionnode* 属性

anomalydetectionnode 属性	**值**	**属性描述**
inputs	*[field1 ... fieldN]*	"异常检测"模型根据指定的输入字段对记录进行筛选。它们不使用目标字段。另外，也不使用权重字段和频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
mode	Expert Simple	
anomaly_method	IndexLevel PerRecords NumRecords	指定用于将记录标记为异常的分界值的确定方法。
index_level	*number*	指定用于标记异常的最小分界值。
percent_records	*number*	根据训练数据中的记录百分比来设置用于标记记录的阈值。
num_records	*number*	根据训练数据中的记录数目来设置用于标记记录的阈值。
num_fields	*integer*	针对每条异常记录报告的字段数。
impute_missing_values	*flag*	
adjustment_coeff	*number*	此值用于对计算距离时赋予连续型字段和分类字段的相对权重进行平衡。
peer_group_num_auto	*flag*	自动计算对等组数。
min_num_peer_groups	*integer*	指定 peer_group_num_auto 设置为 True 时使用的对等组的最小数。
max_num_per_groups	*integer*	指定对等组的最大数目。
num_peer_groups	*integer*	指定 peer_group_num_auto 设置为 False 时使用的对等组数。
noise_level	*number*	确定创建聚类期间处理离群值的方式。指定值必须为 0 到 0.5 之间。
noise_ratio	*number*	指定分配给应该用于噪声缓存的组件的内存比例。指定值必须为 0 到 0.5 之间。

apriorinode 属性

"先验"节点从数据抽取一组规则，即抽取信息内容最多的规则。Apriori 节点提供五种选择规则的方法并使用复杂的索引模式来高效地处理大数据集。对于较大的问题，Apriori 训练的速度通常较快；它对可保留的规则数量没有任何限制，而且可处理最多带有 32 个前提条件的规则。"先验"要求输入和输出字段均为分类型字段，但因为它专为处理此类型数据而进行优化，因而处理速度快得多。

示例

```
node = stream.create("apriori", "My node")
# "Fields" tabnode.setPropertyValue("custom_fields", True)
node.setPropertyValue("partition", "Test")
# For non-transactional
node.setPropertyValue("use_transactional_data", False)
node.setPropertyValue("consequents", ["Age"])
node.setPropertyValue("antecedents", ["BP", "Cholesterol", "Drug"])
# For transactional
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("content_field", "Drug")
# "Model" tabnode.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Apriori_bp_choles_drug")
node.setPropertyValue("min_supp", 7.0)
node.setPropertyValue("min_conf", 30.0)
node.setPropertyValue("max_antecedents", 7)
node.setPropertyValue("true_flags", False)
node.setPropertyValue("optimize", "Memory")
# "Expert" tabnode.setPropertyValue("mode", "Expert")
node.setPropertyValue("evaluation", "ConfidenceRatio")
node.setPropertyValue("lower_bound", 7)
```

表 *111. apriorinode* 属性

apriorinode 属性	**值**	**属性描述**
consequents	字段	Apriori 模型使用"结果"和"前提条件"代替标准的目标字段和输入字段。不使用权重字段和频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
antecedents	[*field1 ... fieldN*]	
min_supp	*number*	
min_conf	*number*	
max_antecedents	*number*	
true_flags	*flag*	
optimize	Speed Memory	
use_transactional_data	*flag*	
contiguous	*flag*	
id_field	*string*	
content_field	*string*	
mode	Simple Expert	
评估	RuleConfidence DifferenceToPrior ConfidenceRatio InformationDifference NormalizedChiSquare	
lower_bound	*number*	
optimize	Speed Memory	用于指定模型构建是针对速度还是内存进行优化。

associationrulesnode 属性

关联规则节点类似于 Apriori 节点；但是，与 Apriori 节点不同的是，关联规则节点可以处理列表数据。另外，可以将关联规则节点与 IBM SPSS Analytic Server 配合使用以处理大型数据以及利用更快的并行处理。

表 *112. associationrulesnode* 属性

associationrulesnode 属性	数据类型	属性描述
预测	字段	此列表中的字段只能显示为规则的预测变量
条件	*[field1...fieldN]*	此列表中的字段只能显示为规则的条件
max_rule_conditions	*integer*	可以在单条规则中包含的条件的最大数目。最小值为 1，最大值为 9。
max_rule_predictions	*integer*	可以在单条规则中包含的预测的最大数目。最小值为 1，最大值为 5。
max_num_rules	*integer*	可以视为规则构建的组成部分的规则的最大数目。最小值为 1，最大值为 10,000。
rule_criterion_top_n	Confidence Rulesupport Lift Conditionsupport Deployability	用于确定选择模型中前"N"条规则时依据的值的规则条件。
true_flags	布尔值	设置为 *Y* 将确定在规则构建期间仅对标志字段考虑 true 值。
rule_criterion	布尔值	设置为 *Y* 将确定在模型构建期间使用规则条件值来排除规则。
min_confidence	*number*	0.1 到 100 - 模型所生成的规则的最低必需置信度级别的百分比值。如果模型所生成规则的置信度级别低于此处指定的值，那么将丢弃此规则。
min_rule_support	*number*	0.1 到 100 - 模型所生成的规则的最低必需规则支持的百分比值。如果模型所生成规则的规则支持级别低于指定值，那么将丢弃此规则。
min_condition_support	*number*	0.1 to 100 - 模型所生成的规则的最低必需条件支持的百分比值。如果模型所生成规则的条件支持级别低于指定值，那么将丢弃此规则。
min_lift	*integer*	1 到 10 - 表示模型所生成的规则的最低必需提升。如果模型所生成规则的提升级别低于指定值，那么将丢弃此规则。
exclude_rules	布尔值	用于选择一列您不希望模型根据其创建规则的相关字段。 例如：set :gsarsnode.exclude_rules = [[[field1,field2, field3]],[[field4, field5]]] - 其中，以 [] 分隔的每个字段列表是表中的一行。
num_bins	*integer*	设置对连续字段进行分级所依据的自动分级的数目。最小值为 2，最大值为 10。

表 *112. associationrulesnode* 属性（续）

associationrulesnode 属性	数据类型	属性描述
max_list_length	*integer*	应用于其最大长度未知的任何列表字段。列表中此处指定数字之前的所有元素将包含在模型构建中；任何其他元素将被丢弃。最小值为 1，最大值为 100。
output_confidence	布尔值	
output_rule_support	布尔值	
output_lift	布尔值	
output_condition_support	布尔值	
output_deployability	布尔值	
rules_to_display	upto all	要在输出表中显示的规则的最大数目。
display_upto	*integer*	如果在 rules_to_display 中设置了 upto，请设置要在输出表中显示的规则的数目。最小值为 1。
field_transformations	布尔值	
records_summary	布尔值	
rule_statistics	布尔值	
most_frequent_values	布尔值	
most_frequent_fields	布尔值	
word_cloud	布尔值	
word_cloud_sort	Confidence Rulesupport Lift Conditionsupport Deployability	
word_cloud_display	*integer*	最小值为 1，最大值为 20。
max_predictions	*integer*	可以对每个输入应用以进行评分的规则的最大数。
criterion	Confidence Rulesupport Lift Conditionsupport Deployability	选择用于确定规则强度的度量。
allow_repeats	布尔值	确定是否在分数中包含具有相同预测的规则。
check_input	NoPredictions Predictions NoCheck	

autoclassifiernode 属性

"自动分类器"节点用于创建和对比二元结果（是或否，流失或不流失等）的若干不同模型，使用户可以选择给定分析的最佳处理方法。由于支持 多种建模算法，因此可以对用户希望使用的方法、每种方法的特定选项以及对比结果的标准进行选择。节点根据指定的选项生成一组模型并根据用户指定的标准排列最佳候选项的顺序。

示例

```
node = stream.create("autoclassifier", "My node")
node.setPropertyValue("ranking_measure", "Accuracy")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_accuracy_limit", True)
node.setPropertyValue("accuracy_limit", 0.9)
node.setPropertyValue("calculate_variable_importance", True)
node.setPropertyValue("use_costs", True)
node.setPropertyValue("svm", False)
```

表 *113. autoclassifiernode* 属性.

autoclassifiernode 属性	**值**	**属性描述**
target	字段	对于标志目标，"自动分类器"节点需要单个目标字段以及一个或多个输入字段。另外，也可以使用权重和频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
ranking_measure	Accuracy Area_under_curve Profit Lift Num_variables	
ranking_dataset	Training Test	
number_of_models	*integer*	这是要包括在模型块中的模型数。指定整数必须为 1 到 100 之间。
calculate_variable_importance	标志	
enable_accuracy_limit	标志	
accuracy_limit	*integer*	介于 0 与 100 之间的整数。
enable_ area_under_curve _limit	标志	
area_under_curve_limit	*number*	介于 0.0 与 1.0 之间的实数。
enable_profit_limit	标志	
profit_limit	*number*	大于 0 的整数。
enable_lift_limit	标志	
lift_limit	*number*	这是大于 1.0 的实数。
enable_number_of_variables_limit	标志	
number_of_variables_limit	*number*	大于 0 的整数。
use_fixed_cost	标志	
fixed_cost	*number*	这是大于 0.0 的实数。
variable_cost	字段	
use_fixed_revenue	标志	
fixed_revenue	*number*	这是大于 0.0 的实数。
variable_revenue	字段	
use_fixed_weight	标志	
fixed_weight	*number*	这是大于 0.0 的实数。
variable_weight	字段	

表 *113. autoclassifiernode* 属性 *(续).*

autoclassifiernode 属性	**值**	**属性描述**
lift_percentile	*number*	介于 0 与 100 之间的整数。
enable_model_build_time_limit	标志	
model_build_time_limit	*number*	设置为分钟数的整数，用于限制构建各个模型所花费的时间。
enable_stop_after_time_limit	标志	
stop_after_time_limit	*number*	这是设置为小时数的实数，用于限制运行自动分类器的总耗时。
enable_stop_after_valid_model_ produced	标志	
use_costs	标志	
<algorithm>	标志	允许或禁止使用特定算法。
<algorithm>.<property>	字符串	设置特定算法的属性值。请参阅主题『设置算法属性』以获取更多信息。

设置算法属性

对于自动分类器、自动数字和自动聚类节点，可以使用常规格式来设置节点所使用的特定算法的属性：

```
autonode.setKeyedPropertyValue(<algorithm>, <property>, <value>)
```

例如：

```
node.setKeyedPropertyValue("neuralnetwork", "method", "MultilayerPerceptron")
```

用于自动分类器节点的算法名称有 cart、chaid、quest、c50、logreg、decisionlist、bayesnet、discriminant、svm 和 knn。

用于自动数字节点的算法名称有 cart、chaid、neuralnetwork、genlin、svm、regression、linear 和 knn。

"自动聚类"节点的算法名称有 twostep、k-means 和 kohonen。

属性名是各算法节点的文档中记录的标准名称。

包含句点或其他标点符号的算法属性必须包含在单引号中，例如：

```
node.setKeyedPropertyValue("logreg", "tolerance", "1.0E-5")
```

可以为属性分配多个值，例如：

```
node.setKeyedPropertyValue("decisionlist", "search_direction", ["Up", "Down"])
```

要启用或禁用特定算法：

```
node.setPropertyValue("chaid", True)
```

注：如果某些算法选项在"自动分类器"节点中不可用，或者只能指定单个值而不能指定值范围，那么编写脚本时的限制与采用标准方式访问节点时的限制相同。

autoclusternode 属性

"自动聚类"节点估算和比较识别具有类似特征记录组的聚类模型。节点工作方式与其他自动建模节点相同，使您在一次建模运行中即可试验多个选项组合。模型可使用基本测量进行比较，以尝试过滤聚类模型的有效性以及对其进行排序，并提供一个基于特定字段的重要性的测量。

示例

```
node = stream.create("autocluster", "My node")
node.setPropertyValue("ranking_measure", "Silhouette")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_silhouette_limit", True)
node.setPropertyValue("silhouette_limit", 5)
```

表 *114. autoclusternode* 属性

autoclusternode 属性	**值**	**属性描述**
评估	字段	**注：**仅自动聚类节点。标识要计算重要性值的字段。另外，可用于标识聚类对此字段的值进行区分的良好程度，从而标识模型预测此字段的良好程度。
ranking_measure	Silhouette Num_clusters Size_smallest_cluster Size_largest_cluster Smallest_to_largest Importance	
ranking_dataset	Training Test	
summary_limit	*integer*	要在报告中列出的模型的数目。指定整数必须为 1 到 100 之间。
enable_silhouette_limit	*flag*	
silhouette_limit	*integer*	介于 0 与 100 之间的整数。
enable_number_less_limit	*flag*	
number_less_limit	*number*	介于 0.0 与 1.0 之间的实数。
enable_number_greater_limit	*flag*	
number_greater_limit	*number*	大于 0 的整数。
enable_smallest_cluster_limit	*flag*	
smallest_cluster_units	Percentage Counts	
smallest_cluster_limit_percentage	*number*	
smallest_cluster_limit_count	*integer*	大于 0 的整数。
enable_largest_cluster_limit	*flag*	
largest_cluster_units	Percentage Counts	
largest_cluster_limit_percentage	*number*	
largest_cluster_limit_count	*integer*	

表 *114. autoclusternode* 属性 *(续)*

autoclusternode 属性	值	属性描述
enable_smallest_largest_limit	*flag*	
smallest_largest_limit	*number*	
enable_importance_limit	*flag*	
importance_limit_condition	Greater_than Less_than	
importance_limit_greater_than	*number*	介于 0 与 100 之间的整数。
importance_limit_less_than	*number*	介于 0 与 100 之间的整数。
<algorithm>	*flag*	允许或禁止使用特定算法。
<algorithm>.<property>	*string*	设置特定算法的属性值。请参阅主题第 173 页的『设置算法属性』以获取更多信息。

autonumericnode 属性

自动数字节点使用多种不同方法估计和对比模型的连续数字范围结果。此节点和自动分类器节点的工作方式相同，因此可以选择要使用和要在单个建模传递中使用多个选项组合进行测试的算法。受支持的算法包括神经网络、C&R 树、CHAID、线性回归、广义线性回归以及支持向量机 (SVM)。可基于相关度、相对错误或已用变量数对模型进行对比。

示例

```
node = stream.create("autonumeric", "My node")
node.setPropertyValue("ranking_measure", "Correlation")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_correlation_limit", True)
node.setPropertyValue("correlation_limit", 0.8)
node.setPropertyValue("calculate_variable_importance", True)
node.setPropertyValue("neuralnetwork", True)
node.setPropertyValue("chaid", False)
```

表 *115. autonumericnode* 属性

autonumericnode 属性	值	属性描述
custom_fields	*flag*	如果为 True，将使用定制字段设置代替"类型"节点设置。
target	字段	"自动数字"节点要求单个目标字段以及一个或多个输入字段。另外，也可以使用权重和频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
inputs	*[field1 ... field2]*	
partition	字段	
use_frequency	*flag*	
frequency_field	字段	
use_weight	*flag*	
weight_field	字段	

表 115. *autonumericnode* 属性 (续)

autonumericnode 属性	值	属性描述
use_partitioned_data	*flag*	如果定义了分区字段，那么仅将训练数据用于模型构建。
ranking_measure	Correlation NumberOfFields	
ranking_dataset	Test Training	
number_of_models	*integer*	这是要包括在模型块中的模型数。指定整数必须为 1 到 100 之间。
calculate_variable_importance	*flag*	
enable_correlation_limit	*flag*	
correlation_limit	*integer*	
enable_number_of_fields_limit	*flag*	
number_of_fields_limit	*integer*	
enable_relative_error_limit	*flag*	
relative_error_limit	*integer*	
enable_model_build_time_limit	*flag*	
model_build_time_limit	*integer*	
enable_stop_after_time_limit	*flag*	
stop_after_time_limit	*integer*	
stop_if_valid_model	*flag*	
<algorithm>	*flag*	允许或禁止使用特定算法。
<algorithm>.<property>	*string*	设置特定算法的属性值。请参阅主题第 173 页的『设置算法属性』以获取更多信息。

bayesnetnode 属性

通过贝叶斯网络节点，你可以利用对真实世界认知的判断力并结合所观察和记录的证据来构建概率模型。该节点重点应用了树扩展简单贝叶斯 (TAN) 和马尔可夫覆盖网络，这些算法主要用于分类问题。

示例

```
node = stream.create("bayesnet", "My node")
node.setPropertyValue("continue_training_existing_model", True)
node.setPropertyValue("structure_type", "MarkovBlanket")
node.setPropertyValue("use_feature_selection", True)
# Expert tabnode.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("independence", "Pearson")
```

表 116. *bayesnetnode* 属性

bayesnetnode 属性	**值**	**属性描述**
inputs	*[field1 ... fieldN]*	贝叶斯网络模型使用单个目标字段以及一个或多个输入字段。连续字段将自动进行分箱。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
continue_training_existing_model	*flag*	
structure_type	TAN MarkovBlanket	请选择要在构建贝叶斯网络时使用的结构。
use_feature_selection	*flag*	
parameter_learning_method	Likelihood Bayes	指定父节点值已知的节点之间的条件概率表的预测方法。
mode	Expert Simple	
missing_values	*flag*	
all_probabilities	*flag*	
independence	Likelihood Pearson	指定用于确定两个变量的成对观测值是否相互独立的方法。
significance_level	*number*	指定用于确定独立性的分界值。
maximal_conditioning_set	*number*	设置用于独立性测试的条件变量的最大数目。
inputs_always_selected	*[field1 ... fieldN]*	指定构建贝叶斯网络时始终使用的数据集字段。 **注：**始终选择目标字段。
maximum_number_inputs	*number*	指定构建贝叶斯网络时使用的输入字段的最大数目。
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

buildr 属性

"R 构建"节点使您能够输入定制 R 脚本，以执行 IBM SPSS Modeler 中部署的模型构建和模型评分。

示例

```
node = stream.create("buildr", "My node")
node.setPropertyValue("score_syntax", """
result<-predict(modelerModel,newdata=modelerData)
modelerData<-cbind(modelerData,result)
```

```
var1<-c(fieldName="NaPrediction",fieldLabel="",fieldStorage="real",fieldMeasure="",
fieldFormat="",fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

表 *117. buildr* 属性.

buildr 属性	**值**	**属性描述**
build_syntax	字符串	这是用于进行模型构建的 R 脚本语法。
score_syntax	*string*	这是用于进行模型评分的 R 脚本语法。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_datetime	*flag*	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。
convert_missing	标志	此选项用于将缺失值转换为 R NA 值。
output_html	标志	此选项用于在 R 模型块中的选项卡上显示图形。
output_text	*flag*	此选项用于将 R 控制台文本输出写至 R 模型块中的选项卡。

c50node 属性

C5.0 节点构建决策树或规则 集。该模型的工作原理是根据在每个级别提供最大信息收获的字段分割样本。目标字段必须为分类字段。允许进行多次多于两个子组的分割。

示例

```
node = stream.create("c50", "My node")
# "Model" tabnode.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "C5_Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("output_type", "DecisionTree")
node.setPropertyValue("use_xval", True)
node.setPropertyValue("xval_num_folds", 3)
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("favor", "Generality")
node.setPropertyValue("min_child_records", 3)
# "Costs" tabnode.setPropertyValue("use_costs", True)
node.setPropertyValue("costs", [["drugA", "drugX", 2]])
```

表 *118. c50node* 属性

c50node 属性	**值**	**属性描述**
target	字段	C50 模型使用单个目标字段以及一个或多个输入字段。另外，还可以指定权重字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。

表 *118. c50node* 属性 *(续)*

c50node 属性	**值**	**属性描述**
output_type	DecisionTree RuleSet	
group_symbolics	*flag*	
use_boost	*flag*	
boost_num_trials	*number*	
use_xval	*flag*	
xval_num_folds	*number*	
mode	Simple Expert	
favor	Accuracy Generality	优先选择准确性还是通用性。
expected_noise	*number*	
min_child_records	*number*	
pruning_severity	*number*	
use_costs	*flag*	
costs	结构化	这是结构化属性。
use_winnowing	*flag*	
use_global_pruning	*flag*	缺省为"启用（True）。
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

carmanode 属性

CARMA 模型不需要用户指定输入或目标字段即可从数据抽取一组规则。与 Apriori 不同，CARMA 节点提供构建规则设置支持（前项和后项支持），而不仅仅是前项支持。这就意味着生成的规则可以用于更多应用程序，例如用于查找产品或服务（前项）的列表，这些产品或服务的后项为想在节日期间促销的商品。

示例

```
node = stream.create("carma", "My node")
# "Fields" tabnode.setPropertyValue("custom_fields", True)
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Drug"])
node.setPropertyValue("partition", "Test")
# "Model" tabnode.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "age_bp_drug")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("min_supp", 10.0)
node.setPropertyValue("min_conf", 30.0)
node.setPropertyValue("max_size", 5)
# Expert Optionsnode.setPropertyValue("mode", "Expert")
```

```
node.setPropertyValue("use_pruning", True)
node.setPropertyValue("pruning_value", 300)
node.setPropertyValue("vary_support", True)
node.setPropertyValue("estimated_transactions", 30)
node.setPropertyValue("rules_without_antecedents", True)
```

表 *119. carmanode* 属性

carmanode 属性	**值**	**属性描述**
inputs	*[field1 ... fieldn]*	CARMA 模型使用输入字段列表，但不使用目标字段。不使用权重字段和频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
id_field	字段	此字段是用于进行模型构建的标识字段。
contiguous	*flag*	用于指定标识字段中的标识是否连续。
use_transactional_data	*flag*	
content_field	字段	
min_supp	*number(percent)*	与规则支持相关，而不是与前提条件支持相关。缺省值为 20%。
min_conf	*number(percent)*	缺省值为 20%。
max_size	*number*	缺省值为 10。
mode	Simple Expert	缺省值为 Simple。
exclude_multiple	*flag*	排除具有多个结果的规则。缺省值为 False。
use_pruning	*flag*	缺省值为 False。
pruning_value	*number*	缺省值为 500。
vary_support	*flag*	
estimated_transactions	*integer*	
rules_without_antecedents	*flag*	

cartnode 属性

分类和回归 (C&R) 树节点生成可用于预测或分类未来观测值的决策树。该方法通过在每个步骤最大限度降低不纯洁度，使用递归分区来将训练记录分割为组。如果树中某个节点中 100% 的观测值都属于目标字段的一个特定类别，那么该节点将被认定为"纯洁"。目标和输入字段可以是数字范围或分类（名义、有序或标志）；所有分割均为二元分割（即仅分割为两个子组）。

示例

```
node = stream.createAt("cart", "My node", 200, 100)
# "Fields" tabnode.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "BP", "Cholesterol"])
# "Build Options" tab, "Objective" panel
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("tree_directives", """Grow Node Index 0 Children 1 2
Grow Node Index 2 Children 3 4""")
# "Build Options" tab, "Basics" panel
```

```
node.setPropertyValue("prune_tree", False)
node.setPropertyValue("use_std_err_rule", True)
node.setPropertyValue("std_err_multiplier", 3.0)
node.setPropertyValue("max_surrogates", 7)
# "Build Options" tab, "Stopping Rules" panel
node.setPropertyValue("use_percentage", True)
node.setPropertyValue("min_parent_records_pc", 5)
node.setPropertyValue("min_child_records_pc", 3)
# "Build Options" tab, "Advanced" panel
node.setPropertyValue("min_impurity", 0.0003)
node.setPropertyValue("impurity_measure", "Twoing")
# "Model Options" tabnode.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Cart_Drug")
```

表 *120. cartnode* 属性

cartnode 属性	**值**	**属性描述**
target	字段	C&R 树模型需要单个目标字段以及一个或多个输入字段。另外，还可以指定频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
continue_training_existing_model	*flag*	
objective	Standard Boosting Bagging psm	psm 用于非常大的数据集，同时需要 Server 连接。
model_output_type	Single InteractiveBuilder	
use_tree_directives	*flag*	
tree_directives	*string*	指定用于使树生长的伪指令。这些伪指令可以括在三重引号中，以避免转义新行或引号。请注意，伪指令可能对数据或建模选项的细微变化高度敏感，并且可能无法通用于其他数据集。
use_max_depth	Default Custom	
max_depth	*integer*	最大树深度从 0 到 1000。只在 use_max_depth = Custom 时使用。
prune_tree	*flag*	修剪树，以避免过度拟合。
use_std_err	*flag*	使用最大风险差值（标准误差）。
std_err_multiplier	*number*	最大差值。
max_surrogates	*number*	最大代用项。
use_percentage	*flag*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*flag*	

表 120. *cartnode* 属性（续）

cartnode 属性	值	属性描述
costs	结构化	结构化属性。
priors	Data Equal Custom	
custom_priors	结构化	结构化属性。
adjust_priors	*flag*	
trails	*number*	用于推进或组装的组件模型数。
set_ensemble_method	Voting HighestProbability HighestMeanProbability	分类目标的缺省组合规则。
range_ensemble_method	Mean 中位数	连续目标的缺省组合规则。
large_boost	*flag*	对非常大型的数据集应用推进。
min_impurity	*number*	
impurity_measure	Gini Twoing Ordered	
train_pct	*number*	防止过度拟合集合。
set_random_seed	*flag*	复制结果选项。
seed	*number*	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

chaidnode 属性

CHAID 使用卡方统计来生成决策树，以确定最佳的分割。CHAID 与 C&R 树和 QUEST 节点不同，它可以生成非二元树，这意味着有些分割将有多于两个的分支。目标和输入字段可以是数字范围（连续）或分类。Exhaustive CHAID 是 CHAID 的修正版，它对所有分割进行更彻底的检查，但计算时间比较长。

示例

```
filenode = stream.createAt("variablefile", "My node", 100, 100)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node = stream.createAt("chaid", "My node", 200, 100)
stream.link(filenode, node)

node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "CHAID")
```

```
node.setPropertyValue("method", "Chaid")
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("tree_directives", "Test")
node.setPropertyValue("split_alpha", 0.03)
node.setPropertyValue("merge_alpha", 0.04)
node.setPropertyValue("chi_square", "Pearson")
node.setPropertyValue("use_percentage", False)
node.setPropertyValue("min_parent_records_abs", 40)
node.setPropertyValue("min_child_records_abs", 30)
node.setPropertyValue("epsilon", 0.003)
node.setPropertyValue("max_iterations", 75)
node.setPropertyValue("split_merged_categories", True)
node.setPropertyValue("bonferroni_adjustment", True)
```

表 *121. chaidnode* 属性

chaidnode 属性	**值**	**属性描述**
target	字段	CHAID 模型需要单个目标字段以及一个或多个输入字段。另外，还可以指定频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
continue_training_existing_model	*flag*	
objective	Standard Boosting Bagging psm	psm 用于非常大的数据集，同时需要 Server 连接。
model_output_type	Single InteractiveBuilder	
use_tree_directives	*flag*	
tree_directives	*string*	
method	Chaid ExhaustiveChaid	
use_max_depth	Default Custom	
max_depth	*integer*	最大树深度从 0 到 1000。只在 use_max_depth = Custom 时使用。
use_percentage	*flag*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*flag*	
costs	结构化	结构化属性。
trails	*number*	用于推进或组装的组件模型数。
set_ensemble_method	Voting HighestProbability HighestMeanProbability	分类目标的缺省组合规则。
range_ensemble_method	Mean 中位数	连续目标的缺省组合规则。

表 121. *chaidnode* 属性（续）

chaidnode 属性	值	属性描述
large_boost	*flag*	对非常大型的数据集应用推进。
split_alpha	*number*	用于执行分割的显著性水平。
merge_alpha	*number*	用于执行合并的显著性水平。
bonferroni_adjustment	*flag*	使用 Bonferroni 法调整显著性值。
split_merged_categories	*flag*	允许对合并的类别进行再分割。
chi_square	Pearson LR	这是用于计算卡方统计的方法：Pearson 或似然比
epsilon	*number*	期望单元格频率的最小变化值。
max_iterations	*number*	收敛的最大迭代次数。
set_random_seed	*integer*	
seed	*number*	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	
maximum_number_of_models	*integer*	

coxregnode 属性

使用 Cox 回归节点，您可以在已有的检查记录中建立时间事件的生存模型。该模型会生成一个生存函数，该函数可预测在给定时间 (*t*) 内对于所给定的输入变量值相关事件的发生概率。

示例

```
node = stream.create("coxreg", "My node")
node.setPropertyValue("survival_time", "tenure")
node.setPropertyValue("method", "BackwardsStepwise")
# Expert tabnode.setPropertyValue("mode", "Expert")
node.setPropertyValue("removal_criterion", "Conditional")
node.setPropertyValue("survival", True)
```

表 122. *coxregnode* 属性

coxregnode 属性	值	属性描述
survival_time	字段	Cox 回归模型需要单个包含生存时间的字段。
target	字段	Cox 回归模型需要单个目标字段以及一个或多个输入字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
method	Enter Stepwise BackwardsStepwise	

表 *122. coxregnode* 属性 *(续)*

coxregnode 属性	**值**	**属性描述**
groups	字段	
model_type	MainEffects Custom	
custom_terms	[*"BP*Sex" "BP*Age"*]	
mode	Expert Simple	
max_iterations	*number*	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
removal_criterion	LR Wald Conditional	
probability_entry	*number*	
probability_removal	*number*	
output_display	EachStep LastStep	
ci_enable	*flag*	
ci_value	90 95 99	
correlation	*flag*	
display_baseline	*flag*	
survival	*flag*	
hazard	*flag*	
log_minus_log	*flag*	
one_minus_survival	*flag*	
separate_line	字段	

表 122. *coxregnode* 属性（续）

coxregnode 属性	值	属性描述
value	数字或字符串	如果未对某个字段指定值，那么将对该字段使用缺省选项"均值"。

decisionlistnode 属性

决策列表节点可标识子组或段，显示与总体相关的给定二元结果的似然度的高低。例如，您或许在寻找那些最不可能流失的客户或最有可能对某个商业活动作出积极响应的客户。通过定制段和并排预览备选模型来比较结果，您可以将自己的业务知识体现在模型中。决策列表模型由一组规则构成，其中每个规则具备一个条件和一个结果。规则依顺序应用，相匹配的第一个规则将决定结果。

示例

```
node = stream.create("decisionlist", "My node")
node.setPropertyValue("search_direction", "Down")
node.setPropertyValue("target_value", 1)
node.setPropertyValue("max_rules", 4)
node.setPropertyValue("min_group_size_pct", 15)
```

表 123. *decisionlistnode* 属性

decisionlistnode 属性	值	属性描述
target	字段	"决策列表"模型使用单个目标以及一个或多个输入字段。另外，还可以指定频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
model_output_type	Model InteractiveBuilder	
search_direction	Up Down	与查找段相关；其中 Up 相当于"高概率"，而 Down 相当于"低概率"。
target_value	*string*	如果未指定，那么标志将采用 true 值。
max_rules	*integer*	除余数以外的最大段数。
min_group_size	*integer*	最小段大小。
min_group_size_pct	*number*	最小段大小（以百分比表示）。
confidence_level	*number*	为了使输入字段符合添加到段定义的条件，输入字段必须将响应似然度提高（提升）的最小阈值。
max_segments_per_rule	*integer*	
mode	Simple Expert	
bin_method	EqualWidth EqualCount	
bin_count	*number*	
max_models_per_cycle	*integer*	列表的搜索宽度。
max_rules_per_cycle	*integer*	段规则的搜索宽度。
segment_growth	*number*	

表 123. *decisionlistnode* 属性（续）

decisionlistnode 属性	**值**	**属性描述**
include_missing	*flag*	
final_results_only	*flag*	
reuse_fields	*flag*	允许重复使用属性（出现在规则中的输入字段）。
max_alternatives	*integer*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

discriminantnode 属性

判别分析所做的假设比 logistic 回归的假设更严格，但在符合这些假设时，判别分析可以作为 logistic 回归分析的有用替代项或补充。

示例

```
node = stream.create("discriminant", "My node")
node.setPropertyValue("target", "custcat")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "Stepwise")
```

表 124. *discriminantnode* 属性

discriminantnode 属性	**值**	**属性描述**
target	字段	"判别"模型需要单个目标字段以及一个或多个输入字段。不使用权重字段和频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
method	Enter Stepwise	
mode	Simple Expert	
prior_probabilities	AllEqual ComputeFromSizes	
covariance_matrix	WithinGroups SeparateGroups	
平均值	*flag*	"高级输出"对话框中的统计选项。
univariate_anovas	*flag*	
box_m	*flag*	
within_group_covariance	*flag*	
within_groups_correlation	*flag*	

表 *124. discriminantnode* 属性 *(续)*

discriminantnode 属性	**值**	**属性描述**
separate_groups_covariance	*flag*	
total_covariance	*flag*	
fishers	*flag*	
未标准化	*flag*	
casewise_results	*flag*	"高级输出"对话框中的分类选项。
limit_to_first	*number*	缺省值为 10。
summary_table	*flag*	
leave_one_classification	*flag*	
combined_groups	*flag*	
separate_groups_covariance	*flag*	矩阵选项**类协方差** 。
territorial_map	*flag*	
combined_groups	*flag*	散点图选项 **联合组** 。
separate_groups	*flag*	散点图选项 **独立组** 。
summary_of_steps	*flag*	
F_pairwise	*flag*	
stepwise_method	WilksLambda UnexplainedVariance MahalanobisDistance SmallestF RaosV	
V_to_enter	*number*	
criteria	UseValue UseProbability	
F_value_entry	*number*	缺省值为 3.84。
F_value_removal	*number*	缺省值为 2.71。
probability_entry	*number*	缺省值为 0.05。
probability_removal	*number*	缺省值为 0.10。
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

extensionmodelnode 属性

通过扩展模型节点，您可以运行 R 或 Python for Spark 脚本来构建结果并对结果进行评分。

Python for Spark 示例

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_build", "extension_build")
node.setPropertyValue("syntax_type", "Python")

build_script = """
import json
import spss.pyspark.runtime
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import DenseVector
from pyspark.mllib.tree import DecisionTree

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
schema = df.dtypes[:]

target = "Drug"
predictors = ["Age","BP","Sex","Cholesterol","Na","K"]

def metaMap(row,schema):
    col = 0
    meta = []
    for (cname, ctype) in schema:
        if ctype == 'string':
            meta.append(set([row[col]]))
        else:
            meta.append((row[col],row[col]))
        col += 1
    return meta

def metaReduce(meta1,meta2,schema):
    col = 0
    meta = []
    for (cname, ctype) in schema:
        if ctype == 'string':
            meta.append(meta1[col].union(meta2[col]))
        else:
            meta.append((min(meta1[col][0],meta2[col][0]),max(meta1[col][1],meta2[col][1])))
        col += 1
    return meta

metadata = df.rdd.map(lambda row: metaMap(row,schema)).reduce(lambda x,y:metaReduce(x,y,schema))

def setToList(v):
    if isinstance(v,set):
        return list(v)
    return v

metadata = map(lambda x: setToList(x), metadata)
print metadata

lookup = {}
for i in range(0,len(schema)):
    lookup[schema[i][0]] = i

def row2LabeledPoint(dm,lookup,target,predictors,row):
    target_index = lookup[target]
    tval = dm[target_index].index(row[target_index])
    pvals = []
    for predictor in predictors:
        predictor_index = lookup[predictor]
        if isinstance(dm[predictor_index],list):
```

```
            pval = dm[predictor_index].index(row[predictor_index])
        else:
            pval = row[predictor_index]
        pvals.append(pval)
    return LabeledPoint(tval, DenseVector(pvals))

# count number of target classes
predictorClassCount = len(metadata[lookup[target]])

# define function to extract categorical predictor information from datamodel
def getCategoricalFeatureInfo(dm,lookup,predictors):
    info = {}
    for i in range(0,len(predictors)):
        predictor = predictors[i]
        predictor_index = lookup[predictor]
        if isinstance(dm[predictor_index],list):
            info[i] = len(dm[predictor_index])
    return info

# convert dataframe to an RDD containing LabeledPoint
lps = df.rdd.map(lambda row: row2LabeledPoint(metadata,lookup,target,predictors,row))

treeModel = DecisionTree.trainClassifier(
    lps,
    numClasses=predictorClassCount,
    categoricalFeaturesInfo=getCategoricalFeatureInfo(metadata, lookup, predictors),
    impurity='gini',
    maxDepth=5,
    maxBins=100)

_outputPath = cxt.createTemporaryFolder()
treeModel.save(cxt.getSparkContext(), _outputPath)
cxt.setModelContentFromPath("TreeModel", _outputPath)
cxt.setModelContentFromString("model.dm",json.dumps(metadata), mimeType="application/json")\
   .setModelContentFromString("model.structure",treeModel.toDebugString())

"""

node.setPropertyValue("python_build_syntax", build_script)
```

R 示例

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_build_syntax", """modelerModel <- lm(modelerData$Na~modelerData$K,modelerData)
modelerDataModel
modelerModel
""")
```

表 *125. extensionmodelnode* 属性

extensionmodelnode 属性	**值**	**属性描述**
syntax_type	*R* *Python*	指定要运行的脚本 - R 或 Python（R 为缺省值）
r_build_syntax	*string*	用于进行模型构建的 R 脚本编制语法。
r_score_syntax	*string*	用于进行模型评分的 R 脚本编制语法。
python_build_syntax	*string*	用于进行模型构建的 Python 脚本编制语法。
python_score_syntax	*string*	用于进行模型评分的 Python 脚本编制语法。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_missing	*flag*	此选项用于将缺失值转换为 R NA 值。

表 *125. extensionmodelnode* 属性 *(续)*

extensionmodelnode 属性	**值**	**属性描述**
convert_datetime	*flag*	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。
output_html	*flag*	此选项用于在 R 模型块中的选项卡上显示图形。
output_text	*flag*	此选项用于将 R 控制台文本输出写至 R 模型块中的选项卡。

factornode 属性

"PCA/因子"节点提供用于降低数据复杂程度的强大数据降维技术。主成份分析（PCA）可找出输入字段的线性组合，该组合最好地捕获了整个字段集合中的方差，且组合中的各个成分相互正交（相互垂直）。因子分析则尝试识别底层因素，这些因素说明了观测的字段集合内的相关性模式。对于这两种方法，其共同的目标是找到可对原始字段集合中的信息进行有效总结的少量派生字段。

示例

```
node = stream.create("factor", "My node")
# "Fields" tabnode.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["BP", "Na", "K"])
node.setPropertyValue("partition", "Test")
# "Model" tabnode.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Factor_Age")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "GLS")
# Expert optionsnode.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", True)
node.setPropertyValue("matrix", "Covariance")
node.setPropertyValue("max_iterations", 30)
node.setPropertyValue("extract_factors", "ByFactors")
node.setPropertyValue("min_eigenvalue", 3.0)
node.setPropertyValue("max_factor", 7)
node.setPropertyValue("sort_values", True)
node.setPropertyValue("hide_values", True)
node.setPropertyValue("hide_below", 0.7)
# "Rotation" sectionnode.setPropertyValue("rotation", "DirectOblimin")
node.setPropertyValue("delta", 0.3)
node.setPropertyValue("kappa", 7.0)
```

表 *126. factornode* 属性

factornode 属性	**值**	**属性描述**
inputs	[*field1 ... fieldN*]	"PCA/因子"模型使用输入字段列表，但不使用目标字段。不使用权重字段和频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。

表 126. *factornode* 属性（续）

factornode 属性	值	属性描述
method	PC ULS GLS ML PAF Alpha Image	
mode	Simple Expert	
max_iterations	*number*	
complete_records	*flag*	
matrix	Correlation 协方差	
extract_factors	ByEigenvalues ByFactors	
min_eigenvalue	*number*	
max_factor	*number*	
旋转	None Varimax DirectOblimin Equamax Quartimax Promax	
delta	*number*	如果选择 DirectOblimin 作为旋转数据的类型，那么可以指定 delta 的值。 如果未指定一个值，那么将使用 delta 的缺省值。
kappa	*number*	如果选择 Promax 作为旋转数据的类型，那么可以指定 kappa 的值。 如果未指定一个值，那么将使用 kappa 的缺省值。
sort_values	*flag*	
hide_values	*flag*	
hide_below	*number*	

featureselectionnode 属性

"特征选择"节点根据一组条件（例如缺失值百分比）筛选要除去的输入字段，然后，相对于指定目标对余下的输入的重要性进行排秩。例如，假如某个给定数据集有上千个潜在输入，那么哪些输入最有可能用于对患者结果进行建模呢？

示例

```
node = stream.create("featureselection", "My node")
node.setPropertyValue("screen_single_category", True)
node.setPropertyValue("max_single_category", 95)
node.setPropertyValue("screen_missing_values", True)
node.setPropertyValue("max_missing_values", 80)
node.setPropertyValue("criteria", "Likelihood")
node.setPropertyValue("unimportant_below", 0.8)
node.setPropertyValue("important_above", 0.9)
node.setPropertyValue("important_label", "Check Me Out!")
node.setPropertyValue("selection_mode", "TopN")
node.setPropertyValue("top_n", 15)
```

有关创建和应用"特征选择"模型的更详细示例，请参阅第 4 页的『独立脚本示例：生成特征选择模型』。

表 *127. featureselectionnode* 属性

featureselectionnode 属性	**值**	**属性描述**
target	字段	"特征选择"模型相对于指定的目标对预测变量进行排秩。不使用权重字段和频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
screen_single_category	*flag*	如果为 True，那么将筛选相对于记录总数而言同个类别中具有过多记录的字段。
max_single_category	*number*	指定 screen_single_category 为 True 时使用的阈值。
screen_missing_values	*flag*	如果为 True，那么将筛选具有过多缺失值的字段，字段数表示为记录总数的百分比。
max_missing_values	*number*	
screen_num_categories	*flag*	如果为 True，那么将筛选相对于记录总数而言具有过多类别的字段。
max_num_categories	*number*	
screen_std_dev	*flag*	如果为 True，那么将筛选标准差小于或等于指定最小值的字段。
min_std_dev	*number*	
screen_coeff_of_var	*flag*	如果为 True，那么将筛选方差系数小于或等于指定最小值的字段。
min_coeff_of_var	*number*	
criteria	Pearson Likelihood CramersV Lambda	根据分类目标对分类预测变量进行排秩时，指定重要性值所依据的度量。
unimportant_below	*number*	指定用于将变量排秩为"重要"、"边际"或"不重要"的 *p* 阈值。接受从 0.0 到 1.0 的值。
important_above	*number*	接受从 0.0 到 1.0 的值。
unimportant_label	*string*	指定"不重要"排秩的标签。
marginal_label	*string*	
important_label	*string*	

表 127. *featureselectionnode* 属性 (续)

featureselectionnode 属性	值	属性描述
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*flag*	在 selection_mode 设置为 ImportanceLevel 时，指定是否选择"重要"字段。
select_marginal	*flag*	在 selection_mode 设置为 ImportanceLevel 时，指定是否选择"边际"字段。
select_unimportant	*flag*	在 selection_mode 设置为 ImportanceLevel 时，指定是否选择"不重要"字段。
importance_value	*number*	在 selection_mode 设置为 ImportanceValue 时，指定要使用的分界值。接受从 0 到 100 的值。
top_n	*integer*	在 selection_mode 设置为 TopN 时，指定要使用的分界值。接受从 0 到 1000 的值。

genlinnode 属性

"广义线性"模型对一般线性模型进行了扩展，这样因变量通过指定的关联函数与因子和协变量线性相关。而且，该模型还允许因变量为非正态分布。它包括统计模型大部分的功能，其中包括线性回归、logistic 回归、用于计数数据的对数线性模型以及区间删失生存模型。

示例

```
node = stream.create("genlin", "My node")
node.setPropertyValue("model_type", "MainAndAllTwoWayEffects")
node.setPropertyValue("offset_type", "Variable")
node.setPropertyValue("offset_field", "Claimant")
```

表 128. *genlinnode* 属性

genlinnode 属性	值	属性描述
target	字段	广义线性模型要求单个目标字段（必须是一个集合或标志），以及一个或多个输入字段。另外，还可以指定权重字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
use_weight	*flag*	
weight_field	字段	字段类型只有"连续"。
target_represents_trials	*flag*	
trials_type	Variable FixedValue	
trials_field	字段	字段类型包括"连续"、"标志"或"有序"。

表 *128. genlinnode* 属性 *(续)*

genlinnode 属性	**值**	**属性描述**
trials_number	*number*	缺省值为 10。
model_type	MainEffects MainAndAllTwoWayEffects	
offset_type	Variable FixedValue	
offset_field	字段	字段类型只有"连续"。
offset_value	*number*	必须是实数。
base_category	Last First	
include_intercept	*flag*	
mode	Simple Expert	
distribution	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS：逆高斯。 NEGBIN：负二项式。
negbin_para_type	Specify Estimate	
negbin_parameter	*number*	缺省值为 1。必须包含非负实数。
tweedie_parameter	*number*	
link_function	IDENTITY CLOGLOG LOG LOGC LOGIT NEGBIN NLOGLOG ODDSPOWER PROBIT POWER CUMCAUCHIT CUMCLOGLOG CUMLOGIT CUMNLOGLOG CUMPROBIT	CLOGLOG：互补双对数。 LOGC：对数补数。 NEGBIN：负二项式。 NLOGLOG：负双对数。 CUMCAUCHIT：累积 Cauchit。 CUMCLOGLOG：累积互补双对数。 CUMLOGIT：累积 Logit。 CUMNLOGLOG：累积负双对数。 CUMPROBIT累积 Probit。
power	*number*	值必须是非零实数。
method	Hybrid Fisher NewtonRaphson	
max_fisher_iterations	*number*	缺省值为 1；只允许使用正整数。

表 *128. genlinnode* 属性 *(续)*

genlinnode 属性	值	属性描述
scale_method	MaxLikelihoodEstimate Deviance PearsonChiSquare FixedValue	
scale_value	*number*	缺省值为 1；必须大于 0。
covariance_matrix	ModelEstimator RobustEstimator	
max_iterations	*number*	缺省值为 100；只允许使用非负整数。
max_step_halving	*number*	缺省值为 5；只允许使用正整数。
check_separation	*flag*	
start_iteration	*number*	缺省值为 20；只允许使用正整数。
estimates_change	*flag*	
estimates_change_min	*number*	缺省值为 1E-006；只允许使用正数。
estimates_change_type	Absolute Relative	
loglikelihood_change	*flag*	
loglikelihood_change_min	*number*	只允许使用正数。
loglikelihood_change_type	Absolute Relative	
hessian_convergence	*flag*	
hessian_convergence_min	*number*	只允许使用正数。
hessian_convergence_type	Absolute Relative	
case_summary	*flag*	
contrast_matrices	*flag*	
descriptive_statistics	*flag*	
estimable_functions	*flag*	
model_info	*flag*	
iteration_history	*flag*	
goodness_of_fit	*flag*	
print_interval	*number*	缺省值为 1；必须是正整数。
model_summary	*flag*	
lagrange_multiplier	*flag*	
parameter_estimates	*flag*	
include_exponential	*flag*	
covariance_estimates	*flag*	
correlation_estimates	*flag*	
analysis_type	TypeI TypeIII TypeIAndTypeIII	
statistics	Wald LR	

表 *128. genlinnode* 属性 *(续)*

genlinnode 属性	值	属性描述
citype	Wald Profile	
tolerancelevel	*number*	缺省值为 0.0001。
confidence_interval	*number*	缺省值为 95。
loglikelihood_function	Full Kernel	
singularity_tolerance	1E-007 1E-008 1E-009 1E-010 1E-011 1E-012	
value_order	Ascending 降序 (Descending) DataOrder	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

glmmnode 属性

广义线性混合模型 (GLMM) 扩展了线性模型，使得目标可以有非正态分布，通过指定的关联函数与因子和协变量线性相关，并且观测值可能相关。广义线性混合模型涵盖从简单线性回归模型到非正态纵向数据的复杂多级模型的各种模型。

表 *129. glmmnode* 属性.

glmmnode 属性	值	属性描述
residual_subject_spec	结构化	这是指定的分类字段的值组合，此组合唯一地定义数据集中的主体。
repeated_measures	结构化	这些字段用于标识重复观测值。
residual_group_spec	[*field1 ... fieldN*]	这些字段用于定义重复效应协方差参数的独立集合。
residual_covariance_type	对角线 (Diagonal) AR1 ARMA11 COMPOUND_SYMMETRY IDENTITY TOEPLITZ UNSTRUCTURED VARIANCE_COMPONENTS	指定残值的协方差结构。

表 *129. glmmnode* 属性 *(续)*.

glmmnode 属性	值	属性描述
custom_target	标志	指明是使用在上游节点定义的目标 (false) 还是由 target_field 指定的定制目标 (true)。
target_field	字段	要用作目标的字段（如果 custom_target 为 true）。
use_trials	标志	指示目标响应是一组试验中发生的众多事件时，是否使用用于指定试验数的附加字段或值。缺省值为 false。
use_field_or_value	字段 值 (Value)	指示是使用字段（缺省）还是值来指定试验数。
trials_field	字段	此字段用于指定试验数。
trials_value	*integer*	此值用于指定试验数。如果指定此属性，那么最小值为 1。
use_custom_target_reference	标志	指示将定制参考类别用于分类目标。缺省值为 false。
target_reference_value	字符串	要使用的参考类别（如果 use_custom_target_reference 为 true）。
dist_link_combination	名义 (Nominal) Logit GammaLog BinomialLogit PoissonLog BinomialProbit NegbinLog BinomialLogC Custom	目标值的分布的公共模型。选择 Custom 可以指定 target_distribution 所提供的列表中的分布。
target_distribution	正态 (Normal) 二项 Multinomial 伽玛 (Gamma) 逆模型 NegativeBinomial 泊松 (Poisson)	当 dist_link_combination 为 Custom 时目标值的分布。
link_function_type	Identity LogC Log CLOGLOG Logit NLOGLOG PROBIT POWER CAUCHIT	这是用于使目标值与预测变量相关的关联函数。如果 target_distribution 为 Binomial，那么您可以使用所列出的任何关联函数。如果 target_distribution 为 Multinomial 之外的任何值，那么您可以使用 CLOGLOG、CAUCHIT、LOGIT、NLOGLOG 或 PROBIT。如果 target_distribution 为 Binomial 或 Multinomial 之外的任何值，那么您可以使用 IDENTITY、LOG 或 POWER。
link_function_param	*number*	要使用的关联函数参数值。仅当 normal_link_function 或 link_function_type 为 POWER 时才适用。

表 *129. glmmnode* 属性 *(续).*

glmmnode 属性	**值**	**属性描述**
use_predefined_inputs	标志	指示固定效应字段是定义为输入字段的上流 (true) 还是来自 fixed_effects_list (false)。缺省值为 false。
fixed_effects_list	结构化	如果 use_predefined_inputs 为 false，那么指定将输入字段用作固定效应字段。
use_intercept	标志	如果为 true（缺省），那么在模型中包括截距。
random_effects_list	结构化	作为随机效应指定的字段列表。
regression_weight_field	字段	此字段用作分析权重字段。
use_offset	None offset_value offset_field	指示如何指定平移。值 None 表示不使用平移。
offset_value	*number*	use_offset 设置为 offset_value 时使用的平移值。
offset_field	字段	use_offset 设置为 offset_field 时用于平移值的字段。
target_category_order	Ascending 降序 (Descending) Data	分类目标的排序顺序。值 Data 指定使用数据中的排序顺序。缺省值为 Ascending。
inputs_category_order	Ascending 降序 (Descending) Data	分类预测变量的排序顺序。值 Data 指定使用数据中的排序顺序。缺省值为 Ascending。
max_iterations	*integer*	此算法要执行的最大迭代次数。非负整数；缺省值为 100。
confidence_level	*integer*	这是用于计算模型系数的区间估计值的置信度级别。非负整数；最大值为 100，缺省值为 95。
degrees_of_freedom_method	Fixed Varied	指定如何计算自由度以进行显著性检验。
test_fixed_effects_coeffecients	Model Robust	这是用于计算参数估计协方差矩阵的方法。
use_p_converge	*flag*	用于参数收敛的选项。
p_converge	*number*	空白或任何正值。
p_converge_type	Absolute Relative	
use_l_converge	*flag*	用于对数似然收敛的选项。
l_converge	*number*	空白或任何正值。
l_converge_type	Absolute Relative	
use_h_converge	*flag*	用于 Hessian 收敛的选项。
h_converge	*number*	空白或任何正值。
h_converge_type	Absolute Relative	
max_fisher_steps	*integer*	

表 129. *glmmnode* 属性 *(续)*.

glmmnode 属性	值	属性描述
singularity_tolerance	*number*	
use_model_name	标志	指示是为模型指定定制名称 (true) 还是使用系统生成的名称 (false)。缺省值为 false。
model_name	字符串	如果 use_model_name 为 true，那么指定使用的模型名称。
confidence	onProbability onIncrease	计算评分置信度值的基础：最高预测概率或者最高与次高预测概率之差。
score_category_probabilities	标志	如果为 true，那么为分类目标生成预测概率。缺省值为 false。
max_categories	*integer*	如果 score_category_probabilities 为 true，那么指定保存最大类别数。
score_propensity	标志	如果为 true，那么为标记目标字段生成倾向评分，指示字段结果为"true"的可能性。
emeans	*structure*	对于固定效应列表中的每个分类字段，指定是否生成估计边际均值。
covariance_list	*structure*	对于固定效应列表中的每个连续字段，指定计算估计边际均值时是使用均值还是定制值。
mean_scale	Original 已转换 (Transformed)	指定是根据目标的原始尺度（缺省）还是根据关联函数转换来计算估计边际均值。
comparison_adjustment_method	LSD SEQBONFERRONI SEQSIDAK	对多个对比执行假设检验时使用的调整方法。

gle 属性

GLE 扩展了线性模型，使得目标可以有非正态分布，通过指定的关联函数与因子和协变量线性相关，并且观测值可能相关。广义线性混合模型涵盖从简单线性回归模型到非正态纵向数据的复杂多级模型的各种模型。

表 130. *gle* 属性

gle 属性	值	属性描述
custom_target	*flag*	指明是使用在上游节点定义的目标 (false) 还是由 target_field 指定的定制目标 (true)。
target_field	字段	要用作目标的字段（如果 custom_target 为 true）。
use_trials	*flag*	指示目标响应是一组试验中发生的众多事件时，是否使用用于指定试验数的附加字段或值。缺省值为 false。
use_trials_field_or_value	字段 Value	指示是使用字段（缺省）还是值来指定试验数。
trials_field	字段	此字段用于指定试验数。

表 *130. gle* 属性 *(续)*

gle 属性	值	属性描述
trials_value	*integer*	此值用于指定试验数。如果指定此属性，那么最小值为 1。
use_custom_target_reference	*flag*	指示将定制参考类别用于分类目标。缺省值为 false。
target_reference_value	*string*	要使用的参考类别（如果 use_custom_target_reference 为 true）。
dist_link_combination	NormalIdentity GammaLog PoissonLog NegbinLog TweedieIdentity NominalLogit BinomialLogit BinomialProbit BinomialLogC CUSTOM	目标值的分布的公共模型。 选择 CUSTOM 可以指定 target_distribution 所提供的列表中的分布。
target_distribution	正态 (Normal) 二项 Multinomial 伽玛 (Gamma) INVERSE_GAUSS NEG_BINOMIAL Poisson TWEEDIE UNKNOWN	当 dist_link_combination 为 Custom 时目标值的分布。

表 *130. gle* 属性 *(续)*

gle 属性	值	属性描述
link_function_type	UNKNOWN IDENTITY LOG LOGIT PROBIT COMPL_LOG_LOG POWER LOG_COMPL NEG_LOG_LOG ODDS_POWER NEG_BINOMIAL GEN_LOGIT CUMUL_LOGIT CUMUL_PROBIT CUMUL_COMPL_LOG_LOG CUMUL_NEG_LOG_LOG CUMUL_CAUCHIT	这是用于使目标值与预测变量相关的关联函数。如果 target_distribution 为 Binomial，那么您可以使用： UNKNOWN IDENTITY LOG LOGIT PROBIT COMPL_LOG_LOG POWER LOG_COMPL NEG_LOG_LOG ODDS_POWER 如果 target_distribution 为 NEG_BINOMIAL，那么您可以使用： NEG_BINOMIAL。 如果 target_distribution 为 UNKNOWN，那么您可以使用： GEN_LOGIT CUMUL_LOGIT CUMUL_PROBIT CUMUL_COMPL_LOG_LOG CUMUL_NEG_LOG_LOG CUMUL_CAUCHIT
link_function_param	*number*	要使用的 Tweedie 参数值。仅当 normal_link_function 或 link_function_type 为 POWER 时才适用。
tweedie_param	*number*	要使用的关联函数参数值。仅在 dist_link_combination 设置为 TweedieIdentity，或者 link_function_type 为 TWEEDIE 时适用。
use_predefined_inputs	*flag*	指示模型效应字段是定义为输入字段的上流 (true) 还是来自 fixed_effects_list (false)。
model_effects_list	结构化	如果 use_predefined_inputs 为 false，那么指定将输入字段用作模型效应字段。
use_intercept	*flag*	如果为 true（缺省），那么在模型中包括截距。
regression_weight_field	字段	此字段用作分析权重字段。
use_offset	None 值 (Value) 变量	指示如何指定平移。值 None 表示不使用平移。
offset_value	*number*	use_offset 设置为 offset_value 时使用的平移值。
offset_field	字段	use_offset 设置为 offset_field 时用于平移值的字段。
target_category_order	Ascending Descending	分类目标的排序顺序。缺省值为 Ascending。
inputs_category_order	Ascending Descending	分类预测变量的排序顺序。缺省值为 Ascending。

表 *130. gle* 属性 *(续)*

gle 属性	值	属性描述
max_iterations	*integer*	此算法要执行的最大迭代次数。非负整数；缺省值为 100。
confidence_level	*number*	这是用于计算模型系数的区间估计值的置信度级别。非负整数；最大值为 100，缺省值为 95。
test_fixed_effects_coeffecients	Model Robust	这是用于计算参数估计协方差矩阵的方法。
detect_outliers	*flag*	在为 true 时，算法查找除多项分布外所有分布的影响离群值。
conduct_trend_analysis	*flag*	在为 true 时，算法执行散射绘图的趋势分析。
estimation_method	FISHER_SCORING NEWTON_RAPHSON HYBRID	指定最大发生可能性估计算法。
max_fisher_iterations	*integer*	如果使用 FISHER_SCORING estimation_method，那么为最大迭代数。最小值为 0，最大值为 20。
scale_parameter_method	MLE FIXED DEVIANCE PEARSON_CHISQUARE	指定用于尺度参数估算的方法。
scale_value	*number*	仅在 scale_parameter_method 设置为 Fixed 时才可用。
negative_binomial_method	MLE FIXED	指定用于负二项式辅助参数估算的方法。
negative_binomial_value	*number*	仅在 negative_binomial_method 设置为 Fixed 时才可用。
use_p_converge	*flag*	用于参数收敛的选项。
p_converge	*number*	空白或任何正值。
p_converge_type	*flag*	True = 绝对，False = 相对
use_l_converge	*flag*	用于对数似然收敛的选项。
l_converge	*number*	空白或任何正值。
l_converge_type	*flag*	True = 绝对，False = 相对
use_h_converge	*flag*	用于 Hessian 收敛的选项。
h_converge	*number*	空白或任何正值。
h_converge_type	*flag*	True = 绝对，False = 相对
max_iterations	*integer*	此算法要执行的最大迭代次数。非负整数；缺省值为 100。
sing_tolerance	*integer*	
use_model_selection	*flag*	启用参数阈值和模型选择方法控制。
method	LASSO ELASTIC_NET FORWARD_STEPWISE RIDGE	确定模型选择方法，或者如果使用 Ridge，那么使用规则化方法。

表 *130. gle* 属性 *(续)*

gle 属性	值	属性描述
detect_two_way_interactions	*flag*	在为 True 时，模型将自动检测输入字段之间的双向交互。 仅在模型仅为主效应（也就是，用户未创建任何更高阶效应）并且选中的 method 为"向前步进"、"套索"或"弹性网络"时才应启用此控制。
automatic_penalty_params	*flag*	仅在模型选择 method 为"套索"或"弹性网络"时才可用。 使用此函数以输入与"套索"或"弹性网络"变量选择方法相关联的惩罚参数。 如果为 True，那么将使用缺省值。如果为 False，那么将启用惩罚参数并且可输入定制值。
lasso_penalty_param	*number*	仅在模型选择 method 为"套索"或"弹性网络"并且 automatic_penalty_params 为 False 时才可用。指定"套索"的惩罚参数值。
elastic_net_penalty_param1	*number*	仅在模型选择 method 为"套索"或"弹性网络"并且 automatic_penalty_params 为 False 时才可用。指定"弹性网络"参数 1 的惩罚参数值。
elastic_net_penalty_param2	*number*	仅在模型选择 method 为"套索"或"弹性网络"并且 automatic_penalty_params 为 False 时才可用。指定"弹性网络"参数 2 的惩罚参数值。
probability_entry	*number*	仅在选中的 method 为"向前步进"时才可用。指定影响包含的 F 统计信息条件的显著性水平。
probability_removal	*number*	仅在选中的 method 为"向前步进"时才可用。指定影响删除的 F 统计信息条件的显著性水平。
use_max_effects	*flag*	仅在选中的 method 为"向前步进"时才可用。 启用 max_effects 控制。 在为 False 时，包含的效应的缺省数值应等于提供给模型的效应总数减去截距。
max_effects	*integer*	指定使用向前步进构建方法时最小效应数。
use_max_steps	*flag*	启用 max_steps 控制。 在为 False 时，缺省步骤数应该是提供给模型的效应数的三倍减去截距。
max_steps	*integer*	指定在使用"向前步进"构建 method 时将采用的最大步骤数。
use_model_name	*flag*	指示是为模型指定定制名称 (true) 还是使用系统生成的名称 (false)。缺省值为 false。
model_name	*string*	如果 use_model_name 为 true，那么指定使用的模型名称。
usePI	*flag*	如果为 true，那么将计算预测变量重要性。

kmeansnode 属性

K-Means 节点将数据集聚类到不同分组（或聚类）。此方法将定义固定的聚类数量，将记录迭代分配给聚类，以及调整聚类中心，直到进一步优化无法再改进模型。*k*-means 节点作为一种非监督学习机制，它并不试图预测结果，而是揭示隐含在输入字段集中的模式。

示例

```
node = stream.create("kmeans", "My node")
# "Fields" tabnode.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Cholesterol", "BP", "Drug", "Na", "K", "Age"])
# "Model" tabnode.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Kmeans_allinputs")
node.setPropertyValue("num_clusters", 9)
node.setPropertyValue("gen_distance", True)
node.setPropertyValue("cluster_label", "Number")
node.setPropertyValue("label_prefix", "Kmeans_")
node.setPropertyValue("optimize", "Speed")
# "Expert" tabnode.setPropertyValue("mode", "Expert")
node.setPropertyValue("stop_on", "Custom")
node.setPropertyValue("max_iterations", 10)
node.setPropertyValue("tolerance", 3.0)
node.setPropertyValue("encoding_value", 0.3)
```

表 *131. kmeansnode* 属性

kmeansnode 属性	**值**	**属性描述**
inputs	[*field1 ... fieldN*]	K-means 模型在一系列输入字段上执行聚类分析，但并不使用目标字段。不使用权重字段和频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
num_clusters	*number*	
gen_distance	*flag*	
cluster_label	字符串 Number	
label_prefix	*string*	
mode	Simple Expert	
stop_on	Default Custom	
max_iterations	*number*	
tolerance	*number*	
encoding_value	*number*	
optimize	Speed Memory	用于指定模型构建是针对速度还是内存进行优化。

knnnode 属性

The *k*-最近相邻元素（KNN）节点将新的个案关联到预测变量空间中与其最邻近的 *k* 个对象的类别或值（其中 *k* 为整数）。类似个案相互靠近，而不同个案相互远离。

示例

```
node = stream.create("knn", "My node")
# "目标"选项卡node.setPropertyValue("objective", "Custom")
# "设置"选项卡 - "相邻元素"面板node.setPropertyValue("automatic_k_selection", False)
node.setPropertyValue("fixed_k", 2)
node.setPropertyValue("weight_by_importance", True)
# "设置"选项卡 - "分析"面板node.setPropertyValue("save_distances", True)
```

表 *132. knnnode* 属性

knnnode 属性	值	属性描述
analysis	PredictTarget IdentifyNeighbors	
objective	Balance Speed Accuracy Custom	
normalize_ranges	*flag*	
use_case_labels	*flag*	此复选框用于启用下一个选项。
case_labels_field	字段	
identify_focal_cases	*flag*	此复选框用于启用下一个选项。
focal_cases_field	字段	
automatic_k_selection	*flag*	
fixed_k	*integer*	只有当 automatic_k_selectio 为 False 时才启用。
minimum_k	*integer*	只有当 automatic_k_selectio 为 True 时才启用。
maximum_k	*integer*	
distance_computation	Euclidean CityBlock	
weight_by_importance	*flag*	
range_predictions	Mean 中位数	
perform_feature_selection	*flag*	
forced_entry_inputs	[*field1 ... fieldN*]	
stop_on_error_ratio	*flag*	
number_to_select	*integer*	
minimum_change	*number*	
validation_fold_assign_by_field	*flag*	

表 132. *knnnode* 属性 (续)

knnnode 属性	值	属性描述
number_of_folds	*integer*	只有当 validation_fold_assign_by_field 为 False 时才启用。
set_random_seed	*flag*	
random_seed	*number*	
folds_field	字段	只有当 validation_fold_assign_by_field 为 True 时才启用。
all_probabilities	*flag*	
save_distances	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

kohonennode 属性

Kohonen 节点会生成一种神经网络，此神经网络可用于将数据集聚类到各个差异组。此网络训练完成后，相似的记录应在输出映射中紧密地聚集，差异大的记录则应彼此远离。您可以通过查看模型块 中每个单元所捕获观测值的数量来找出规模较大的单元。这将让您对聚类的相应数量有所估计。

示例

```
node = stream.create("kohonen", "My node")
# "Model" tabnode.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Symbolic Cluster")
node.setPropertyValue("stop_on", "Time")
node.setPropertyValue("time", 1)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 12345)
node.setPropertyValue("optimize", "Speed")
# "Expert" tabnode.setPropertyValue("mode", "Expert")
node.setPropertyValue("width", 3)
node.setPropertyValue("length", 3)
node.setPropertyValue("decay_style", "Exponential")
node.setPropertyValue("phase1_neighborhood", 3)
node.setPropertyValue("phase1_eta", 0.5)
node.setPropertyValue("phase1_cycles", 10)
node.setPropertyValue("phase2_neighborhood", 1)
node.setPropertyValue("phase2_eta", 0.2)
node.setPropertyValue("phase2_cycles", 75)
```

表 133. *kohonennode* 属性

kohonennode 属性	值	属性描述
inputs	[*field1 ... fieldN*]	Kohonen 模型使用输入字段的列表，但不使用目标。不使用频率字段和权重字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。

表 133. *kohonennode* 属性（续）

kohonennode 属性	值	属性描述
continue	*flag*	
show_feedback	*flag*	
stop_on	Default Time	
time	*number*	
optimize	Speed Memory	用于指定模型构建是针对速度还是内存进行优化。
cluster_label	*flag*	
mode	Simple Expert	
width	*number*	
长度	*number*	
decay_style	线性 (Linear) Exponential	
phase1_neighborhood	*number*	
phase1_eta	*number*	
phase1_cycles	*number*	
phase2_neighborhood	*number*	
phase2_eta	*number*	
phase2_cycles	*number*	

linearnode 属性

线性回归模型根据目标与一个或多个预测变量之间的线性关系预测连续目标。

示例

```
node = stream.create("linear", "My node")
# Build Options tab - Objectives panel
node.setPropertyValue("objective", "Standard")
# Build Options tab - Model Selection panel
node.setPropertyValue("model_selection", "BestSubsets")
node.setPropertyValue("criteria_best_subsets", "ASE")
# Build Options tab - Ensembles panel
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

表 134. *linearnode* 属性.

linearnode 属性	值	属性描述
target	字段	指定单个目标字段。

表 *134. linearnode* 属性 *(续).*

linearnode 属性	**值**	**属性描述**
inputs	[*field1 ... fieldN*]	模型使用的预测变量字段。
continue_training_existing_model	标志	
objective	Standard Bagging Boosting psm	psm 用于非常大的数据集，同时需要 Server 连接。
use_auto_data_preparation	标志	
confidence_level	*number*	
model_selection	ForwardStepwise BestSubsets None	
criteria_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	
probability_entry	*number*	
probability_removal	*number*	
use_max_effects	标志	
max_effects	*number*	
use_max_steps	标志	
max_steps	*number*	
criteria_best_subsets	AICC AdjustedRSquare ASE	
combining_rule_continuous	Mean Median	
component_models_n	*number*	
use_random_seed	标志	
random_seed	*number*	
use_custom_model_name	标志	
custom_model_name	字符串	
use_custom_name	标志	
custom_name	字符串	
tooltip	字符串	
keywords	字符串	
annotation	字符串	

linearasnode 属性

线性回归模型根据目标与一个或多个预测变量之间的线性关系预测连续目标。

表 135. *linearasnode* 属性

linearasnode 属性	值	属性描述
target	字段	指定单个目标字段。
inputs	[*field1 ... fieldN*]	模型使用的预测变量字段。
weight_field	字段	模型使用的分析字段。
custom_fields	*flag*	缺省值为 TRUE。
intercept	*flag*	缺省值为 TRUE。
detect_2way_interaction	*flag*	是否考虑双向交互。 缺省值为 TRUE。
cin	*number*	用于计算模型系数的估算的置信区间。 请指定大于 0 且小于 100 的值。 缺省值为 95。
factor_order	ascending descending	分类预测变量的排序顺序。 缺省值为 ascending。
var_select_method	ForwardStepwise BestSubsets none	要使用的模型选择方法。 缺省值为 ForwardStepwise。
criteria_for_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	用于确定应向模型中添加效应还是应从模型中移除效应的统计信息。 缺省值为 AdjustedRSquare。
pin	*number*	将向模型中添加具有小于此指定 pin 阈值的最小 p 值的效应。 缺省值为 0.05。
pout	*number*	将移除模型中具有大于此指定 pout 阈值的 p 值的任何效应。 缺省值为 0.10。
use_custom_max_effects	*flag*	是否在最终模型中使用最大效应数。 缺省值为 FALSE。
max_effects	*number*	要在最终模型中使用的最大效应数。 缺省值为 1。
use_custom_max_steps	*flag*	是否使用最大步骤数。缺省值为 FALSE。
max_steps	*number*	分步算法停止之前的最大步骤数。 缺省值为 1。
criteria_for_best_subsets	AICC AdjustedRSquare ASE	要使用的条件方式。 缺省值为 AdjustedRSquare。

logregnode 属性

Logistic 回归是一种统计方法，它可根据输入字段的值对记录进行分类。它类似于线性回归，但采用的是类别目标字段而非数字范围。

多项式示例

```
node = stream.create("logreg", "My node")
# "Fields" tabnode.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tabnode.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Log_reg Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Stepwise")
node.setPropertyValue("logistic_procedure", "Multinomial")
node.setPropertyValue("multinomial_base_category", "BP")
node.setPropertyValue("model_type", "FullFactorial")
node.setPropertyValue("custom_terms", [["BP", "Sex"], ["Age"], ["Na", "K"]])
node.setPropertyValue("include_constant", False)
# "Expert" tabnode.setPropertyValue("mode", "Expert")
node.setPropertyValue("scale", "Pearson")
node.setPropertyValue("scale_value", 3.0)
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("tolerance", "1.0E-7")
# "Convergence..." section
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("max_steps", 3)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
node.setPropertyValue("delta", 0.03)
# "Output..." section
node.setPropertyValue("summary", True)
node.setPropertyValue("likelihood_ratio", True)
node.setPropertyValue("asymptotic_correlation", True)
node.setPropertyValue("goodness_fit", True)
node.setPropertyValue("iteration_history", True)
node.setPropertyValue("history_steps", 3)
node.setPropertyValue("parameters", True)
node.setPropertyValue("confidence_interval", 90)
node.setPropertyValue("asymptotic_covariance", True)
node.setPropertyValue("classification_table", True)
# "Stepping" optionsnode.setPropertyValue("min_terms", 7)
node.setPropertyValue("use_max_terms", True)
node.setPropertyValue("max_terms", 10)
node.setPropertyValue("probability_entry", 3)
node.setPropertyValue("probability_removal", 5)
node.setPropertyValue("requirements", "Containment")
```

二项式示例

```
node = stream.create("logreg", "My node")
# "Fields" tabnode.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Cholesterol")
node.setPropertyValue("inputs", ["BP", "Drug", "Age"])
```

```
node.setPropertyValue("partition", "Test")
# "Model" tabnode.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Log_reg Cholesterol")
node.setPropertyValue("multinomial_base_category", "BP")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("binomial_method", "Forwards")
node.setPropertyValue("logistic_procedure", "Binomial")
node.setPropertyValue("binomial_categorical_input", "Sex")
node.setKeyedPropertyValue("binomial_input_contrast", "Sex", "Simple")
node.setKeyedPropertyValue("binomial_input_category", "Sex", "Last")
node.setPropertyValue("include_constant", False)
# "Expert" tabnode.setPropertyValue("mode", "Expert")
node.setPropertyValue("scale", "Pearson")
node.setPropertyValue("scale_value", 3.0)
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("tolerance", "1.0E-7")
# "Convergence..." section
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
# "Output..." section
node.setPropertyValue("binomial_output_display", "at_each_step")
node.setPropertyValue("binomial_goodness_of_fit", True)
node.setPropertyValue("binomial_iteration_history", True)
node.setPropertyValue("binomial_parameters", True)
node.setPropertyValue("binomial_ci_enable", True)
node.setPropertyValue("binomial_ci", 85)
# "Stepping" optionsnode.setPropertyValue("binomial_removal_criterion", "LR")
node.setPropertyValue("binomial_probability_removal", 0.2)
```

表 *136. logregnode* 属性.

logregnode 属性	**值**	**属性描述**
target	字段	Logistic 回归模型需要一个目标字段以及一个或多个输入字段。不使用频率字段和权重字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
logistic_procedure	二项 Multinomial	
include_constant	标志	
mode	Simple Expert	
method	Enter Stepwise Forwards Backwards BackwardsStepwise	
binomial_method	Enter Forwards Backwards	
model_type	MainEffects FullFactorial Custom	将 FullFactorial 指定为模型类型时，即使指定了步进方法，步进方法也不会运行。而是使用 Enter 方法。 如果将模型类型设置为 Custom，但未指定定制字段，那么将构建主效应模型。

表 *136. logregnode* 属性 *(续).*

logregnode 属性	**值**	**属性描述**
custom_terms	[*[BP Sex][BP][Age]*]	
multinomial_base_category	字符串	指定如何确定参考类别。
binomial_categorical_input	字符串	
binomial_input_contrast	Indicator Simple Difference Helmert Repeated Polynomial Deviation	这是分类输入的键控属性，用于指定如何确定对比。
binomial_input_category	First Last	这是分类输入的键控属性，用于指定如何确定参考类别。
scale	None UserDefined Pearson Deviance	
scale_value	*number*	
all_probabilities	标志	
tolerance	1.0E-5 1.0E-6 1.0E-7 1.0E-8 1.0E-9 1.0E-10	
min_terms	*number*	
use_max_terms	标志	
max_terms	*number*	
entry_criterion	Score LR	
removal_criterion	LR Wald	
probability_entry	*number*	
probability_removal	*number*	
binomial_probability_entry	*number*	
binomial_probability_removal	*number*	
requirements	HierarchyDiscrete HierarchyAll Containment None	
max_iterations	*number*	
max_steps	*number*	

表 *136. logregnode* 属性 *(续)*.

logregnode 属性	值	属性描述
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
delta	*number*	
iteration_history	标志	
history_steps	*number*	
summary	标志	
likelihood_ratio	标志	
asymptotic_correlation	标志	
goodness_fit	标志	
parameters	标志	
confidence_interval	*number*	
asymptotic_covariance	标志	
classification_table	标志	
stepwise_summary	标志	
info_criteria	标志	
monotonicity_measures	标志	
binomial_output_display	at_each_step at_last_step	
binomial_goodness_of_fit	标志	
binomial_parameters	标志	
binomial_iteration_history	标志	
binomial_classification_plots	标志	
binomial_ci_enable	标志	
binomial_ci	*number*	
binomial_residual	outliers all	
binomial_residual_enable	标志	
binomial_outlier_threshold	*number*	
binomial_classification_cutoff	*number*	
binomial_removal_criterion	LR Wald Conditional	

表 136. *logregnode* 属性 (续).

logregnode 属性	值	属性描述
calculate_variable_importance	标志	
calculate_raw_propensities	标志	

lsvmnode 属性

使用线性支持向量机 (LSVM) 节点，可以将数据分为两组，而无需过度拟合。LSVM 为线性，并且可以与宽数据集配合使用，例如，那些含有大量记录的数据集。

表 137. *lsvmnode* 属性

lsvmnode 属性	值	属性描述
intercept	*flag*	在模型中包括截距。缺省值为 True。
target_order	Ascending Descending	指定分类目标的排序循序。对于连续目标忽略。缺省值为 Ascending。
precision	*number*	仅当目标字段的测量级别为 Continuous 时才使用。指定与回归损失的敏感性相关的参数。最小值为 0，并且无最大值。缺省值为 0.1。
exclude_missing_values	*flag*	在为 True 时，如果任何单个值丢失，那么将排除记录。缺省值为 False。
penalty_function	L1 L2	指定使用的惩罚函数的类型。缺省值为 L2。
lambda	*number*	惩罚（规则化）参数。
calculate_variable_importance	*flag*	对于生成相应的的重要性度量的模型，此选项显示一个图表，其中指示估计模型期间每个预测变量的相对重要性。请注意，对于某些模型，计算变量重要性（特别对较大数据集进行操作时）可能需要花较长时间，因此默认情况下，对于某些模型，变量重要性均处于关闭状态。变量重要性对于决策列表模型不可用。

neuralnetnode 属性

要点：在此发行版中提供了具有增强功能的新版本的神经网络建模节点，并将在下一节 (*neuralnetwork*) 中进行介绍。尽管您仍然可以使用先前版本来构建模型并对其评分，但我们建议您将脚本更新为使用新版本。这里保留了先前版本的详细信息以供您参考。

示例

```
node = stream.create("neuralnet", "My node")
# "Fields" tabnode.setPropertyValue("custom_fields", True)
node.setPropertyValue("targets", ["Drug"])
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
# "Model" tabnode.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Dynamic")
node.setPropertyValue("train_pct", 30)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 12345)
node.setPropertyValue("stop_on", "Time")
node.setPropertyValue("accuracy", 95)
node.setPropertyValue("cycles", 200)
node.setPropertyValue("time", 3)
node.setPropertyValue("optimize", "Speed")
# "Multiple Method Expert Options" section
node.setPropertyValue("m_topologies", "5 30 5; 2 20 3, 1 10 1")
node.setPropertyValue("m_non_pyramids", False)
node.setPropertyValue("m_persistence", 100)
```

表 *138. neuralnetnode* 属性

neuralnetnode 属性	**值**	**属性描述**
targets	[*field1 ... fieldN*]	"神经网络"节点需要一个或多个目标字段以及一个或多个输入字段。将忽略频率和权重字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
method	Quick Dynamic Multiple Prune ExhaustivePrune RBFN	
prevent_overtrain	*flag*	
train_pct	*number*	
set_random_seed	*flag*	
random_seed	*number*	
mode	Simple Expert	
stop_on	Default Accuracy Cycles Time	停止方式。
准确性	*number*	停止准确性。
cycles	*number*	训练周期数。
time	*number*	训练时间（分钟数）。
continue	*flag*	
show_feedback	*flag*	
binary_encode	*flag*	
use_last_model	*flag*	
gen_logfile	*flag*	
logfile_name	*string*	

表 *138. neuralnetnode* 属性 *(续)*

neuralnetnode 属性	**值**	**属性描述**
alpha	*number*	
initial_eta	*number*	
high_eta	*number*	
low_eta	*number*	
eta_decay_cycles	*number*	
hid_layers	One Two Three	
hl_units_one	*number*	
hl_units_two	*number*	
hl_units_three	*number*	
持久性	*number*	
m_topologies	*string*	
m_non_pyramids	*flag*	
m_persistence	*number*	
p_hid_layers	One Two Three	
p_hl_units_one	*number*	
p_hl_units_one	*number*	
p_hl_units_three	*number*	
p_persistence	*number*	
p_hid_rate	*number*	
p_hid_rate	*number*	
p_inp_rate	*number*	
p_inp_pers	*number*	
p_overall_pers	*number*	
r_persistence	*number*	
r_num_clusters	*number*	
r_eta_auto	*flag*	
r_alpha	*number*	
r_eta	*number*	
optimize	Speed Memory	用于指定模型构建是针对速度还是内存进行优化。
calculate_variable_importance	*flag*	注：此属性取代了先前版本中使用的 sensitivity_analysis 属性。仍然支持旧属性，但建议使用 calculate_variable_importance。
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

表 138. *neuralnetnode* 属性（续）

neuralnetnode 属性	值	属性描述
adjusted_propensity_partition	Test Validation	

neuralnetworknode 属性

神经网络节点使用的模型是对人类大脑处理信息的方式简化了的模型。此模型通过模拟大量类似于神经元的抽象形式的互连简单处理单元而运行。神经网络是功能强大的一般函数估计器，只需要最少的统计或数学知识就可以对其进行训练或应用。

示例

```
node = stream.create("neuralnetwork", "My node")
# Build Options tab - Objectives panel
node.setPropertyValue("objective", "Standard")
# Build Options tab - Ensembles panel
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

表 139. *neuralnetworknode* 属性

neuralnetworknode 属性	值	属性描述
targets	[*field1 ... fieldN*]	指定目标字段。
inputs	[*field1 ... fieldN*]	模型使用的预测变量字段。
splits	[*field1 ... fieldN*	指定一个或多个用于分割建模的字段。
use_partition	*flag*	如果定义了分区字段，那么此选项可确保仅训练分区的数据用于构建模型。
continue	*flag*	继续训练现有模型。
objective	Standard Bagging Boosting psm	psm 用于非常大的数据集，同时需要 Server 连接。
method	MultilayerPerceptron RadialBasisFunction	
use_custom_layers	*flag*	
first_layer_units	*number*	
second_layer_units	*number*	
use_max_time	*flag*	
max_time	*number*	
use_max_cycles	*flag*	
max_cycles	*number*	
use_min_accuracy	*flag*	
min_accuracy	*number*	
combining_rule_categorical	Voting HighestProbability HighestMeanProbability	

表 139. *neuralnetworknode* 属性（续）

neuralnetworknode 属性	值	属性描述
combining_rule_continuous	Mean Median	
component_models_n	*number*	
overfit_prevention_pct	*number*	
use_random_seed	*flag*	
random_seed	*number*	
missing_values	listwiseDeletion missingValueImputation	
use_model_name	布尔值	
model_name	*string*	
confidence	onProbability onIncrease	
score_category_probabilities	*flag*	
max_categories	*number*	
score_propensity	*flag*	
use_custom_name	*flag*	
custom_name	*string*	
tooltip	*string*	
keywords	*string*	
annotation	*string*	

questnode 属性

QUEST 节点可提供用于构建决策树的二元分类法，此方法的设计目的是减少大型 C&R 树分析所需的处理时间，同时也减少在分类树方法中发现的趋势以便支持允许有多个分割的输入。输入字段可以是数字范围（连续），但目标字段必须是分类。所有分割都是二元的。

示例

```
node = stream.create("quest", "My node")
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("max_surrogates", 5)
node.setPropertyValue("split_alpha", 0.03)
node.setPropertyValue("use_percentage", False)
node.setPropertyValue("min_parent_records_abs", 40)
node.setPropertyValue("min_child_records_abs", 30)
node.setPropertyValue("prune_tree", True)
node.setPropertyValue("use_std_err", True)
node.setPropertyValue("std_err_multiplier", 3)
```

表 *140. questnode* 属性

questnode 属性	**值**	**属性描述**
target	字段	QUEST 模型需要一个目标字段以及一个或多个输入字段。另外，还可以指定频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
continue_training_existing_model	*flag*	
objective	Standard Boosting Bagging psm	psm 用于非常大的数据集，同时需要 Server 连接。
model_output_type	Single InteractiveBuilder	
use_tree_directives	*flag*	
tree_directives	*string*	
use_max_depth	Default Custom	
max_depth	*integer*	最大树深度从 0 到 1000。只在 use_max_depth = Custom 时使用。
prune_tree	*flag*	修剪树，以避免过度拟合。
use_std_err	*flag*	使用最大风险差值（标准误差）。
std_err_multiplier	*number*	最大差值。
max_surrogates	*number*	最大代用项。
use_percentage	*flag*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*flag*	
costs	结构化	结构化属性。
priors	Data Equal Custom	
custom_priors	结构化	结构化属性。
adjust_priors	*flag*	
trails	*number*	用于推进或组装的组件模型数。
set_ensemble_method	Voting HighestProbability HighestMeanProbability	分类目标的缺省组合规则。
range_ensemble_method	Mean 中位数	连续目标的缺省组合规则。
large_boost	*flag*	对非常大型的数据集应用推进。
split_alpha	*number*	用于执行分割的显著性水平。
train_pct	*number*	防止过度拟合集合。

表 *140. questnode* 属性 *(续)*

questnode 属性	值	属性描述
set_random_seed	*flag*	复制结果选项。
seed	*number*	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

randomtrees 属性

"随机树"节点类似于现有 C&RT 节点；但是，"随机树"节点旨在处理大数据以创建单个树，并在 SPSS Modeler V17 中添加的输出查看器中显示生成的模型。"随机树"树节点生成用于预测或分类未来观测的决策树。该方法通过在每个步骤最大限度降低不纯洁度，使用递归分区来将训练记录分割为组。如果树中某个节点中 100% 的观测值都属于目标字段的一个特定类别，那么该节点将被认定为纯洁。目标和输入字段可以是数字范围或分类（名义、有序或标志）；所有分割均为二元分割（即仅分割为两个子组）。

表 *141. randomtrees* 属性

randomtrees 属性	值	属性描述
target	字段	在"随机树"节点中，模型需要单个目标以及一个或多个输入字段。另外，还可以指定频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
number_of_models	*integer*	确定要作为整体建模的一部分构建的模型数量。
use_number_of_predictors	*flag*	确定是否使用 number_of_predictors。
number_of_predictors	*integer*	指定在构建拆分模型时要使用的预测变量数。
use_stop_rule_for_accuracy	*flag*	确定在无法提高准确性是否停止模型构建。
sample_size	*number*	减少该值可提高处理非常大的数据集时的性能。
handle_imbalanced_data	*flag*	如果模型的目标是特定标志成果，并且期望的成果占非期望的成果的比率非常小，那么数据不平衡，并且模型执行的引导程序采样可能影响模型精确性。支持不平衡的数据处理，从而使模型将捕获期望大部分期望的成果并生成更强大的模型。
use_weighted_sampling	*flag*	在为 *False* 时，将以相同的概率随机选择每个节点的变量。在为 *True* 时，将响应加权并选择变量。
max_node_number	*integer*	单个树中允许的最大节点数。如果在下一次拆分时将超过此数值，那么树增长停止。
max_depth	*integer*	增强停止前最大树深度。

表 141. randomtrees 属性（续）

randomtrees 属性	值	属性描述
min_child_node_size	*integer*	确定在拆分父节点后子节点中允许的最小记录数。如果子节点包含的记录数小于此处指定的数值，那么将不会拆分父节点
use_costs	*flag*	
costs	结构化	结构化属性。格式是由 3 个值组成的列表：实际值、预测值和成本（如果预测错误）。 例如： tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])
default_cost_increase	none linear square 定制	**注：**仅对有序目标启用。 设置成本矩阵中的缺省值。
max_pct_missing	*integer*	如果任何输入中缺失值的百分比大于此处指定的值，那么将排除输入。最小值为 0，最大值为 100。
exclude_single_cat_pct	*integer*	如果一个类别值表示记录的百分比高于此处指定的值，那么将从模型构建中排除整个字段。最小值为 1，最大值为 99。
max_category_number	*integer*	如果字段中类别数量超过该值，那么将从模型构建中排除此字段。最小值为 2。
min_field_variation	*number*	如果连续字段的变体系数小于该值，那么将从模型构建中排除此字段。
num_bins	*integer*	仅当数据由连续输入组成时才使用。 设置要用于输入的等频箱数；选项为：2、4、5、10、20、25、50 或 100。
topN	*integer*	指定要报告的规则数量。缺省值为 50，最小值为 1，最大值为 1000。

regressionnode 属性

线性回归是一种通过拟合直线或平面以实现汇总数据和预测的普通统计方法，它可使预测值和实际输出值之间的差异最小化。

注：在未来的版本中，线性节点将替换回归节点。我们建议您从现在开始使用线性模型进行线性回归。

示例

```
node = stream.create("regression", "My node")
# "Fields" tabnode.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Age")
node.setPropertyValue("inputs", ["Na", "K"])
node.setPropertyValue("partition", "Test")
node.setPropertyValue("use_weight", True)
```

```
node.setPropertyValue("weight_field", "Drug")
# "Model" tabnode.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Regression Age")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Stepwise")
node.setPropertyValue("include_constant", False)
# "Expert" tabnode.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", False)
node.setPropertyValue("tolerance", "1.0E-3")
# "Stepping..." section
node.setPropertyValue("stepping_method", "Probability")
node.setPropertyValue("probability_entry", 0.77)
node.setPropertyValue("probability_removal", 0.88)
node.setPropertyValue("F_value_entry", 7.0)
node.setPropertyValue("F_value_removal", 8.0)
# "Output..." section
node.setPropertyValue("model_fit", True)
node.setPropertyValue("r_squared_change", True)
node.setPropertyValue("selection_criteria", True)
node.setPropertyValue("descriptives", True)
node.setPropertyValue("p_correlations", True)
node.setPropertyValue("collinearity_diagnostics", True)
node.setPropertyValue("confidence_interval", True)
node.setPropertyValue("covariance_matrix", True)
node.setPropertyValue("durbin_watson", True)
```

表 *142. regressionnode* 属性

regressionnode 属性	**值**	**属性描述**
target	字段	"回归"模型需要单个目标字段以及一个或多个输入字段。另外，还可以指定权重字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
method	Enter Stepwise Backwards Forwards	
include_constant	*flag*	
use_weight	*flag*	
weight_field	字段	
mode	Simple Expert	
complete_records	*flag*	

表 142. *regressionnode* 属性 (续)

regressionnode 属性	值	属性描述
tolerance	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 1.0E-9 1.0E-10 1.0E-11 1.0E-12	请使用双引号将自变量括起来。
stepping_method	useP useF	useP：使用 F 的概率 useF：使用 F 值
probability_entry	*number*	
probability_removal	*number*	
F_value_entry	*number*	
F_value_removal	*number*	
selection_criteria	*flag*	
confidence_interval	*flag*	
covariance_matrix	*flag*	
collinearity_diagnostics	*flag*	
regression_coefficients	*flag*	
exclude_fields	*flag*	
durbin_watson	*flag*	
model_fit	*flag*	
r_squared_change	*flag*	
p_correlations	*flag*	
descriptives	*flag*	
calculate_variable_importance	*flag*	

sequencenode 属性

序列节点可发现连续数据或与时间有关的数据中的关联规则。序列是一系列可能会以可预测顺序发生的项目集合。例如，一个购买了剃刀和须后水的顾客可能在下次购物时购买剃须膏。序列节点基于 CARMA 关联规则算法，该算法使用一个有效的两次传递方法查找序列。

示例

```
node = stream.create("sequence", "My node")
# "Fields" tabnode.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("use_time_field", True)
```

```
node.setPropertyValue("time_field", "Date1")
node.setPropertyValue("content_fields", ["Drug", "BP"])
node.setPropertyValue("partition", "Test")
# "Model" tabnode.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Sequence_test")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("min_supp", 15.0)
node.setPropertyValue("min_conf", 14.0)
node.setPropertyValue("max_size", 7)
node.setPropertyValue("max_predictions", 5)
# "Expert" tabnode.setPropertyValue("mode", "Expert")
node.setPropertyValue("use_max_duration", True)
node.setPropertyValue("max_duration", 3.0)
node.setPropertyValue("use_pruning", True)
node.setPropertyValue("pruning_value", 4.0)
node.setPropertyValue("set_mem_sequences", True)
node.setPropertyValue("mem_sequences", 5.0)
node.setPropertyValue("use_gaps", True)
node.setPropertyValue("min_item_gap", 20.0)
node.setPropertyValue("max_item_gap", 30.0)
```

表 *143. sequencenode* 属性

sequencenode 属性	**值**	**属性描述**
id_field	字段	要创建序列规则集，您需要指定一个标识字段以及一个可选的时间字段，以及一个或多个内容字段。不使用权重字段和频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
time_field	字段	
use_time_field	*flag*	
content_fields	[*field1 ... fieldn*]	
contiguous	*flag*	
min_supp	*number*	
min_conf	*number*	
max_size	*number*	
max_predictions	*number*	
mode	Simple Expert	
use_max_duration	*flag*	
max_duration	*number*	
use_gaps	*flag*	
min_item_gap	*number*	
max_item_gap	*number*	
use_pruning	*flag*	
pruning_value	*number*	
set_mem_sequences	*flag*	
mem_sequences	*integer*	

slrmnode 属性

自学响应模型（SLRM）节点可用于构建一个包含单个新观测值或少量新观测值的模型，通过此模型，无需使用全部数据对模型进行重新训练即可对模型进行重新评估。

示例

```
node = stream.create("slrm", "My node")
node.setPropertyValue("target", "Offer")
node.setPropertyValue("target_response", "Response")
node.setPropertyValue("inputs", ["Cust_ID", "Age", "Ave_Bal"])
```

表 144. *slrmnode* 属性

slrmnode 属性	值	属性描述
target	字段	目标字段必须是名义字段或标志字段。另外，还可以指定频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
target_response	字段	类型必须是标志。
continue_training_existing_model	*flag*	
target_field_values	*flag*	Use all：使用来自源的全部值。 Specify：选择所需的值。
target_field_values_specify	*[field1 ... fieldN]*	
include_model_assessment	*flag*	
model_assessment_random_seed	*number*	必须是实数。
model_assessment_sample_size	*number*	必须是实数。
model_assessment_iterations	*number*	迭代数。
display_model_evaluation	*flag*	
max_predictions	*number*	
randomization	*number*	
scoring_random_seed	*number*	
sort	Ascending Descending	指定是先显示评分最高还是最低的报价。
model_reliability	*flag*	
calculate_variable_importance	*flag*	

statisticsmodelnode 属性

"Statistics 模型"节点使您能够通过运行将会生成 PMML 的 IBM SPSS Statistics 过程来分析和处理数据。此节点需要 IBM SPSS Statistics 的许可副本。

有关此节点属性的信息，请参阅第 316 页的『statisticsmodelnode 属性』。

stpnode 属性

空间时间预测 (STP) 节点使用包含位置数据、用于预测的输入字段（预测变量）、时间字段和目标字段的数据。此数据中的每个位置都包含很多行，这些行表示每次进行测量时每个预测变量的值。对此数据进行分析之后，可以使用它预测分析中使用的模型数据内任何位置的目标值。

表 *145. stpnode* 属性

stpnode 属性	**数据类型**	**属性描述**
字段选项卡		
target	字段	此为目标字段。
位置	字段	模型的位置字段。仅允许使用地理空间字段。
location_label	字段	分类字段，用于在输出中为 location 中所选位置添加标签
time_field	字段	模型的时间字段。仅允许使用具有连续测量的字段，并且存储类型必须为时间、日期、时间戳记或整数。
inputs	*[field1 ... fieldN]*	输入字段的列表。
时间间隔选项卡		
interval_type_timestamp	Years Quarters Months Weeks Days Hours Minutes Seconds	
interval_type_date	Years Quarters Months Weeks Days	
interval_type_time	Hours Minutes Seconds	限制创建 STP 用于计算的时间索引时需要考虑的每周天数
interval_type_integer	Periods （仅时间索引字段，整数存储）	数据集将转换为的时间区间。可用选项取决于选择用作模型的 time_field 的字段的存储类型。
period_start	*integer*	

表 *145. stpnode* 属性 *(续)*

stpnode 属性	**数据类型**	**属性描述**
start_month	January February March April May June July August September October November December	模型将开始建立索引的起始月份（例如，如果此项设置为三月，但数据集中第一条记录为一月，那么此模型将跳过前两条记录并从三月开始建立索引）。
week_begins_on	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	STP 根据数据创建的时间索引的起始点
days_per_week	*integer*	最小值为 1，最大值为 7，增量为 1
hours_per_day	*integer*	模型在某天中占用的小时数。如果此项设置为 10，那么模型将从 day_begins_at 时间开始建立索引并持续 10 个小时，然后跳至与 day_begins_at 值匹配的下一个值等等。
day_begins_at	00:00 01:00 02:00 03:00 ... 23:00	设置模型开始建立索引的起始小时值。
interval_increment	1 2 3 4 5 6 10 12 15 20 30	此增量设置针对分钟或秒。此项决定了模型根据数据创建索引的位置。因此，在增量为 30 并且时间间隔类型为秒的情况下，模型每 30 秒根据数据创建一个索引。

表 *145. stpnode* 属性 *(续)*

stpnode 属性	**数据类型**	**属性描述**
data_matches_interval	布尔值	如果设置为 N，那么在构建模型前，会将数据转换为常规 interval_type。 如果数据已使用正确格式，并且 interval_type 和所有关联设置都与数据相匹配，请将此项设置为 Y 以阻止转换或汇总数据。 将此项设置为 Y 会禁用所有汇总控制。
agg_range_default	Sum Mean 最小值 最大值 中位数 第一个四分位 第三个四分位	此项决定用于连续字段的缺省汇总方法。未明确包含在定制汇总中的所有连续字段将使用此处指定的方法进行汇总。
custom_agg	[[field, aggregation method],[]..] 演示： [['x5' 'FirstQuartile']['x4' 'Sum']]	结构化属性： 脚本参数：custom_agg 例如： set :stpnode.custom_agg = [[field1 function] [field2 function]] 其中 function 是要与该字段配合使用的汇总函数。
基本选项卡		
include_intercept	*flag*	
max_autoregressive_lag	*integer*	最小值为 1，最大值为 5，增量为 1。此项为预测所需的先前记录数。因此，如果设置为 5，那么将使用先前的 5 条记录创建新预测。此处根据构建数据指定的记录数将合并到模型中，因此用户无需在对模型进行评分时再次提供数据。
estimation_method	非参数 非参数	用于对空间协方差矩阵进行建模的方法
parametric_model	Gaussian Exponential PoweredExponential	参数空间协方差模型的有序参数
exponential_power	*number*	PoweredExponential 模型的幂级。最小值为 1，最大值为 2。
高级选项卡		
max_missing_values	*integer*	允许在模型中使用的具有缺失值的记录的最大百分比。
显著水平	*number*	模型构建中假设测试的显著性水平。指定 STP 模型估算中所有检验（包括两项拟合优度检验、效应 F 检验及系数 T 检验）的显著性值。
输出选项卡		

表 *145. stpnode* 属性 *(续)*

stpnode 属性	**数据类型**	**属性描述**
model_specifications	*flag*	
temporal_summary	*flag*	
location_summary	*flag*	决定"位置摘要"表是否包括在模型输出中。
model_quality	*flag*	
test_mean_structure	*flag*	
mean_structure_coefficients	*flag*	
autoregressive_coefficients	*flag*	
test_decay_space	*flag*	
parametric_spatial_covariance	*flag*	
correlations_heat_map	*flag*	
correlations_map	*flag*	
location_clusters	*flag*	
similarity_threshold	*number*	阈值，在此阈值处输出集群被认为足够相似而能够合并到单个集群中。
max_number_clusters	*integer*	可以在模型输出中包含的集群数的上限。
模型选项选项卡		
use_model_name	*flag*	
model_name	*string*	
uncertainty_factor	*number*	最小值为 0，最大值为 100。决定应用于未来预测的不确定性（错误）增大。它是预测的上限和下限。

svmnode 属性

使用支持向量机 (SVM) 节点，可以将数据分为两组，而无需过度拟合。SVM 可以与宽数据集配合使用，例如那些含有大量输入字段的数据集。

示例

```
node = stream.create("svm", "My node")
# Expert tabnode.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("kernel", "Polynomial")
node.setPropertyValue("gamma", 1.5)
```

表 *146. svmnode* 属性.

svmnode 属性	**值**	**属性描述**
all_probabilities	标志	

表 *146. svmnode* 属性 *(续)*.

svmnode 属性	值	属性描述
stopping_criteria	1.0E-1 1.0E-2 1.0E-3 (default) 1.0E-4 1.0E-5 1.0E-6	确定何时停止优化算法。
regularization	*number*	也称为 C 参数。
precision	*number*	仅当目标字段的测量级别为 Continuous 时才使用。
kernel	RBF（缺省） Polynomial Sigmoid Linear	用于转换的内核函数的类型。
rbf_gamma	*number*	仅在 kernel 为 RBF 时使用。
gamma	*number*	仅在 kernel 为 Polynomial 或 Sigmoid 时使用。
bias	*number*	
degree	*number*	仅在 kernel 为 Polynomial 时使用。
calculate_variable_importance	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	
adjusted_propensity_partition	Test Validation	

tcmnode 属性

时间因果建模尝试在时间序列数据中发现关键的因果关系。在时间因果建模中，您指定一组目标序列，并指定这些目标的一组候选输入。然后，此过程将为每个目标构建自回归时间序列模型，并且仅包含与目标具有最重要的因果关系的输入。

表 *147. tcmnode* 属性

tcmnode 属性	值	属性描述
custom_fields	布尔值	
dimensionlist	[*dimension1 ... dimensionN*]	
data_struct	Multiple Single	
metric_fields	字段	
both_target_and_input	[*f1 ... fN*]	
targets	[*f1 ... fN*]	

表 147. tcmnode 属性 (续)

tcmnode 属性	值	属性描述
candidate_inputs	[*f1 ... fN*]	
forced_inputs	[*f1 ... fN*]	
use_timestamp	Timestamp Period	
input_interval	None 未知 年 季度 月 周 Day 小时 Hour_nonperiod 分钟 Minute_nonperiod 秒 Second_nonperiod	
period_field	*string*	
period_start_value	*integer*	
num_days_per_week	*integer*	
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	*integer*	
start_hour_of_day	*integer*	
timestamp_increments	*integer*	
cyclic_increments	*integer*	
cyclic_periods	列表	
output_interval	None 年 季度 月 周 Day 小时 分钟 秒	
is_same_interval	相同 Notsame	
cross_hour	布尔值	
aggregate_and_distribute	列表	

表 *147. tcmnode* 属性 *(续)*

tcmnode 属性	**值**	**属性描述**
aggregate_default	Mean 合计 众数 最小值 最大值	
distribute_default	Mean 合计	
group_default	Mean 合计 众数 最小值 最大值	
missing_imput	Linear_interp Series_mean K_mean K_meridian Linear_trend 无	
k_mean_param	*integer*	
k_median_param	*integer*	
missing_value_threshold	*integer*	
conf_level	*integer*	
max_num_predictor	*integer*	
max_lag	*integer*	
epsilon	*number*	
threshold	*integer*	
is_re_est	布尔值	
num_targets	*integer*	
percent_targets	*integer*	
fields_display	列表	
series_display	列表	
network_graph_for_target	布尔值	
sign_level_for_target	*number*	
fit_and_outlier_for_target	布尔值	
sum_and_para_for_target	布尔值	
impact_diag_for_target	布尔值	
impact_diag_type_for_target	作用 原因 Both	
impact_diag_level_for_target	*integer*	
series_plot_for_target	布尔值	
res_plot_for_target	布尔值	
top_input_for_target	布尔值	

表 *147. tcmnode* 属性 *(续)*

tcmnode 属性	**值**	**属性描述**
forecast_table_for_target	布尔值	
same_as_for_target	布尔值	
network_graph_for_series	布尔值	
sign_level_for_series	*number*	
fit_and_outlier_for_series	布尔值	
sum_and_para_for_series	布尔值	
impact_diagram_for_series	布尔值	
impact_diagram_type_for_series	作用 原因 Both	
impact_diagram_level_for_series	*integer*	
series_plot_for_series	布尔值	
residual_plot_for_series	布尔值	
forecast_table_for_series	布尔值	
outlier_root_cause_analysis	布尔值	
causal_levels	*integer*	
outlier_table	交互 轴 Both	
rmsp_error	布尔值	
bic	布尔值	
r_square	布尔值	
outliers_over_time	布尔值	
series_transormation	布尔值	
use_estimation_period	布尔值	
estimation_period	时间 观测值	
observations	列表	
observations_type	Latest 最早	
observations_num	*integer*	
observations_exclude	*integer*	
extend_records_into_future	布尔值	
forecastperiods	*integer*	
max_num_distinct_values	*integer*	
display_targets	FIXEDNUMBER PERCENTAGE	
goodness_fit_measure	ROOTMEAN BIC RSQUARE	
top_input_for_series	布尔值	

表 147. tcmnode 属性 (续)

tcmnode 属性	值	属性描述
aic	布尔值	
rmse	布尔值	

ts 属性

"时间序列"节点估计时间序列数据的指数平滑模型、单变量自回归整合移动平均值 (ARIMA) 模型和多变量 ARIMA（即转换函数）模型，并生成未来性能的预测数据。此"时间序列"节点类似于在 SPSS Modeler 第 18 版中不推荐的旧"时间序列"节点。但是，此更新的"时间序列"节点设计为利用 IBM SPSS Analytic Server 能力来处理大数据，并在 SPSS Modeler 第 17 版中添加的输出查看器中显示产生的模型。

表 148. ts 属性

ts 属性	值(V)	属性说明
targets	*field*	"时间序列"节点可以预测一个或多个目标，可以选择使用一个或多个输入字段作为预测变量。不使用频率字段和权重字段。请参阅主题第 167 页的『公共建模节点属性』，了解更多信息。
candidate_inputs	[*field1 ... fieldN*]	模型所使用的输入或预测变量字段。
use_period	*flag*	
date_time_field	*field*	
input_interval	None 未知 Year Quarter Month Week Day Hour Hour_nonperiod Minute Minute_nonperiod Second Second_nonperiod	
period_field	*field*	
period_start_value	整数	
num_days_per_week	整数	

表 *148. ts* 属性 *(续)*

ts 属性	值(V)	属性说明
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	整数	
start_hour_of_day	整数	
timestamp_increments	整数	
cyclic_increments	整数	
cyclic_periods	列表	
output_interval	None Year Quarter Month Week Day Hour Minute Second	
is_same_interval	*flag*	
cross_hour	*flag*	
aggregate_and_distribute	列表	
aggregate_default	Mean Sum Mode Min Max	
distribute_default	Mean Sum	
group_default	Mean Sum Mode Min Max	
missing_imput	Linear_interp Series_mean K_mean K_median Linear_trend	
k_span_points	整数	
use_estimation_period	*flag*	
estimation_period	Observations Times	

表 *148. ts* 属性 *(续)*

ts 属性	值(V)	属性说明
date_estimation	列表	仅当使用 date_time_field 时可用
period_estimation	列表	仅当使用 use_period 时可用
observations_type	Latest Earliest	
observations_num	整数	
observations_exclude	整数	
method	ExpertModeler Exsmooth Arima	
expert_modeler_method	ExpertModeler Exsmooth Arima	
consider_seasonal	*flag*	
detect_outliers	*flag*	
expert_outlier_additive	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_innovational	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_transient	*flag*	
expert_outlier_seasonal_additive	*flag*	
expert_outlier_local_trend	*flag*	
expert_outlier_additive_patch	*flag*	
consider_newesmodels	*flag*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative DampedTrendAdditive DampedTrendMultiplicative MultiplicativeTrendAdditive MultiplicativeSeasonal MultiplicativeTrendMultiplicative MultiplicativeTrend	指定"指数平滑"方法。缺省值为 Simple。

表 *148. ts* 属性 *(续)*

ts 属性	值(V)	属性说明
futureValue_type_method	Compute specify	如果使用 Compute，那么系统会计算每个预测器的预测时间段的将来值。 针对每个预测器，可从函数列表（空、最近点的均值和最新值）进行选择，或使用 specify 以手动输入值。要指定单个字段和属性，请使用 extend_metric_values 属性。例如： set :ts.futureValue_type_method="specify" set :ts.extend_metric_values=[{'Market_1','US {'Market_2','MOST_RECENT_VALUE', ''},{'Market_3
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima.p	整数	
arima.d	整数	
arima.q	整数	
arima.sp	整数	
arima.sd	整数	
arima.sq	整数	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*flag*	
tf_arima.p. *fieldname*	整数	用于转换函数。
tf_arima.d. *fieldname*	整数	用于转换函数。
tf_arima.q. *fieldname*	整数	用于转换函数。
tf_arima.sp. *fieldname*	整数	用于转换函数。
tf_arima.sd. *fieldname*	整数	用于转换函数。
tf_arima.sq. *fieldname*	整数	用于转换函数。
tf_arima.delay. *fieldname*	整数	用于转换函数。
tf_arima.transformation_type. *fieldname*	None SquareRoot NaturalLog	用于转换函数。
arima_detect_outliers	*flag*	
arima_outlier_additive	*flag*	
arima_outlier_level_shift	*flag*	
arima_outlier_innovational	*flag*	
arima_outlier_transient	*flag*	
arima_outlier_seasonal_additive	*flag*	
arima_outlier_local_trend	*flag*	

表 *148. ts* 属性 *(续)*

ts 属性	值(V)	属性说明
arima_outlier_additive_patch	*flag*	
max_lags	整数	
cal_PI	*flag*	
conf_limit_pct	*real*	
events	字段	
continue	*flag*	
scoring_model_only	*flag*	用于包含大量（数万）时间序列的模型。
forecastperiods	整数	
extend_records_into_future	*flag*	
extend_metric_values	字段	允许您为预测变量提供未来值。
conf_limits	*flag*	
noise_res	*flag*	
max_models_output	整数	用于控制输出中显示的模型的数量。缺省值为 10。如果构建的模型总数超过此值，那么模型不会显示在输出。模型仍可用于评分。

timeseriesnode 属性（不推荐）

注：此原始"时间序列"节点在 SPSS Modeler V18 中已不推荐，并且替换为新"时间序列"节点，此节点设计为利用 IBM SPSS Analytic Server 能力来处理大数据。"时间序列"节点估计时间序列数据的指数平滑模型、单变量自回归整合移动平均值 (ARIMA) 模型和多变量 ARIMA（即转换函数）模型，并生成未来性能的预测数据。在"时间序列"节点之前必须有"时间区间"节点。

示例

```
node = stream.create("timeseries", "My node")
node.setPropertyValue("method", "Exsmooth")
node.setPropertyValue("exsmooth_model_type", "HoltsLinearTrend")
node.setPropertyValue("exsmooth_transformation_type", "None")
```

表 *149. timeseriesnode* 属性

timeseriesnode 属性	值	属性描述
targets	字段	"时间序列"节点可以预测一个或多个目标，可以选择使用一个或多个输入字段作为预测变量。不使用频率字段和权重字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
continue	*flag*	

表 *149. timeseriesnode* 属性 *(续)*

timeseriesnode 属性	**值**	**属性描述**
method	ExpertModeler Exsmooth Arima Reuse	
expert_modeler_method	*flag*	
consider_seasonal	*flag*	
detect_outliers	*flag*	
expert_outlier_additive	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_innovational	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_transient	*flag*	
expert_outlier_seasonal_additive	*flag*	
expert_outlier_local_trend	*flag*	
expert_outlier_additive_patch	*flag*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*integer*	
arima_d	*integer*	
arima_q	*integer*	
arima_sp	*integer*	
arima_sd	*integer*	
arima_sq	*integer*	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*flag*	
tf_arima_p. *fieldname*	*integer*	用于转换函数。
tf_arima_d. *fieldname*	*integer*	用于转换函数。
tf_arima_q. *fieldname*	*integer*	用于转换函数。
tf_arima_sp. *fieldname*	*integer*	用于转换函数。
tf_arima_sd. *fieldname*	*integer*	用于转换函数。
tf_arima_sq. *fieldname*	*integer*	用于转换函数。
tf_arima_delay. *fieldname*	*integer*	用于转换函数。

表 *149. timeseriesnode* 属性 *(续)*

timeseriesnode 属性	**值**	**属性描述**
tf_arima_transformation_type. *fieldname*	None SquareRoot NaturalLog	用于转换函数。
arima_detect_outlier_mode	None Automatic	
arima_outlier_additive	*flag*	
arima_outlier_level_shift	*flag*	
arima_outlier_innovational	*flag*	
arima_outlier_transient	*flag*	
arima_outlier_seasonal_additive	*flag*	
arima_outlier_local_trend	*flag*	
arima_outlier_additive_patch	*flag*	
conf_limit_pct	*real*	
max_lags	*integer*	
events	字段	
scoring_model_only	*flag*	用于包含大量（数万）时间序列的模型。

treeas 属性

"树-AS"节点类似于现有 CHAID 节点；但是，"树-AS"节点旨在处理大数据以创建单个树，并在 SPSS Modeler V17 中添加的输出查看器中显示生成的模型。此节点通过使用卡方统计 (CHAID) 来识别最优拆分，从而生成决策树。 对 CHAID 的这一使用可生成非二元树，意味着某些拆分将具有两个以上的分支。目标和输入字段可以是数字范围（连续）或分类。Exhaustive CHAID 是 CHAID 的修正版，它对所有分割进行更彻底的检查，但计算时间比较长。

表 *150. treeas* 属性

treeas 属性	**值**	**属性描述**
target	字段	在 Tree-AS 节点中，CHAID 模型需要单个目标和一个或多个输入字段。 另外，还可以指定频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
method	chaid exhaustive_chaid	
max_depth	*integer*	最大树深度（从 0 到 20）。 默认值为 5。
num_bins	*integer*	仅当数据由连续输入组成时才使用。 设置要用于输入的等频箱数；选项为：2、4、5、10、20、25、50 或 100。
record_threshold	*integer*	记录数，在达到此数量的情况下在构建树时模型将从使用 p 值切换为效应大小。 缺省值为 1,000,000；请按增量 10,000 将此增大或减小。

表 *150. treeas* 属性 *(续)*

treeas 属性	**值**	**属性描述**
split_alpha	*number*	用于执行分割的显著性水平。该值必须介于 0.01 和 0.99 之间。
merge_alpha	*number*	用于执行合并的显著性水平。该值必须介于 0.01 和 0.99 之间。
bonferroni_adjustment	*flag*	使用 Bonferroni 法调整显著性值。
effect_size_threshold_cont	*number*	设置在使用连续目标拆分节点和合并类别时的效应大小阈值。 该值必须介于 0.01 和 0.99 之间。
effect_size_threshold_cat	*number*	设置使用分类目标拆分节点和合并类别时的效应大小阈值。 该值必须介于 0.01 和 0.99 之间。
split_merged_categories	*flag*	允许对合并的类别进行再分割。
grouping_sig_level	*number*	用于确定如何形成节点组或如何识别异常节点。
chi_square	pearson likelihood_ratio	这是用于计算卡方统计的方法：Pearson 或似然比
minimum_record_use	use_percentage use_absolute	
min_parent_records_pc	*number*	缺省值为 2。最小值为 1，最大值为 100，增量为 1。父分支值必须高于子分支。
min_child_records_pc	*number*	缺省值为 1。最小值为 1，最大值为 100，增量为 1。
min_parent_records_abs	*number*	缺省值为 100。最小值为 1，最大值为 100，增量为 1。父分支值必须高于子分支。
min_child_records_abs	*number*	缺省值为 50。 最小值为 1，最大值为 100，增量为 1。
epsilon	*number*	期望单元格频率的最小变化值。
max_iterations	*number*	收敛的最大迭代次数。
use_costs	*flag*	
costs	结构化	结构化属性。格式是由 3 个值组成的列表：实际值、预测值和成本（如果预测错误）。 例如： tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])
default_cost_increase	none linear square 定制	**注：**仅对有序目标启用。 设置成本矩阵中的缺省值。
calculate_conf	*flag*	
display_rule_id	*flag*	在评分输出中添加一个字段，表示每个记录分配到的终端节点的标识。

twostepnode 属性

"二阶"节点使用两步聚类方法。第一步完成简单数据处理，以便将原始输入数据压缩为可管理的子聚类集合。第二步使用层级聚类方法将子聚类一步一步合并为更大的聚类。"二阶"具有一个优点，就是能够为训练数据自动估计最佳聚类数。它可以高效处理混合的字段类型和大型的数据集。

示例

```
node = stream.create("twostep", "My node")
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Age", "K", "Na", "BP"])
node.setPropertyValue("partition", "Test")
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "TwoStep_Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("exclude_outliers", True)
node.setPropertyValue("cluster_label", "String")
node.setPropertyValue("label_prefix", "TwoStep_")
node.setPropertyValue("cluster_num_auto", False)
node.setPropertyValue("max_num_clusters", 9)
node.setPropertyValue("min_num_clusters", 3)
node.setPropertyValue("num_clusters", 7)
```

表 *151. twostepnode* 属性

twostepnode 属性	**值**	**属性描述**
inputs	[*field1 ... fieldN*]	"二阶"模型使用输入字段列表，但不使用目标字段。不识别权重字段和频率字段。有关更多信息，请参阅第 167 页的『公共建模节点属性』主题。
standardize	*flag*	
exclude_outliers	*flag*	
percentage	*number*	
cluster_num_auto	*flag*	
min_num_clusters	*number*	
max_num_clusters	*number*	
num_clusters	*number*	
cluster_label	字符串 Number	
label_prefix	*string*	
distance_measure	Euclidean Loglikelihood	
clustering_criterion	AIC BIC	

twostepAS 属性

"二阶聚类"是一个探索工具，用于揭示数据集中原本不明显的自然分组（即聚类）。此过程使用的算法有多个不错的特征使其有别于传统聚类技术，例如处理分类和连续变量、聚类数目的自动选择以及可扩展性。

表 152. *twostepAS* 属性

twostepAS 属性	**值**	**属性描述**
inputs	[*f1 ... fN*]	TwoStepAS 模型使用一列输入字段，但不使用目标字段。不识别权重字段和频率字段。
use_predefined_roles	布尔值	缺省值为 True
use_custom_field_assignments	布尔值	缺省值为 False
cluster_num_auto	布尔值	缺省值为 True
min_num_clusters	integer	缺省值为 2
max_num_clusters	integer	缺省值为 15
num_clusters	integer	缺省值为 5
clustering_criterion	AIC BIC	
automatic_clustering_method	use_clustering_criterion_setting Distance_jump 最小值 最大值	
feature_importance_method	use_clustering_criterion_setting effect_size	
use_random_seed	布尔值	
random_seed	integer	
distance_measure	Euclidean Loglikelihood	
include_outlier_clusters	布尔值	缺省值为 True
num_cases_in_feature_tree_leaf_is _less_than	integer	缺省值为 10
top_perc_outliers	integer	缺省值为 5
initial_dist_change_threshold	integer	缺省值为 0
leaf_node_maximum_branches	integer	缺省值为 8
non_leaf_node_maximum_branches	integer	缺省值为 8
max_tree_depth	integer	缺省值为 3
adjustment_weight_on_measurement _level	integer	缺省值为 6
memory_allocation_mb	数字	缺省值为 512
delayed_split	布尔值	缺省值为 True
fields_to_standardize	[*f1 ... fN*]	
adaptive_feature_selection	布尔值	缺省值为 True
featureMisPercent	integer	缺省值为 70

表 *152. twostepAS* 属性 *(续)*

twostepAS 属性	**值**	**属性描述**
coefRange	数字	缺省值为 0.05
percCasesSingleCategory	integer	缺省值为 95
numCases	integer	缺省值为 24
include_model_specifications	布尔值	缺省值为 True
include_record_summary	布尔值	缺省值为 True
include_field_transformations	布尔值	缺省值为 True
excluded_inputs	布尔值	缺省值为 True
evaluate_model_quality	布尔值	缺省值为 True
show_feature_importance bar chart	布尔值	缺省值为 True
show_feature_importance_ word_cloud	布尔值	缺省值为 True
show_outlier_clusters interactive_table_and_chart	布尔值	缺省值为 True
show_outlier_clusters_pivot_table	布尔值	缺省值为 True
across_cluster_feature_importance	布尔值	缺省值为 True
across_cluster_profiles_pivot_table	布尔值	缺省值为 True
withinprofiles	布尔值	缺省值为 True
cluster_distances	布尔值	缺省值为 True
cluster_label	字符串 Number	
label_prefix	字符串	

第 14 章 模型块节点属性

模型块节点具有与其他节点相同的公共属性。有关更多信息，请参阅第 63 页的『公共节点属性』主题。

applyanomalydetectionnode 属性

您可以使用"异常检测"建模节点来生成"异常检测"模型块。该模型块的脚本名称为 *applyanomalydetectionnode*。有关编写建模节点自身脚本的详细信息，请参阅第 167 页的『anomalydetectionnode 属性』。

表 *153. applyanomalydetectionnode* 属性.

属性	值	属性描述
anomaly_score_method	FlagAndScore FlagOnly ScoreOnly	确定创建哪些输出用于评分。
num_fields	*integer*	要报告的字段。
discard_records	标志	指示是否从输出中丢弃记录。
discard_anomalous_records	标志	指示是丢弃异常记录还是丢弃 非异常记录。缺省状态为 off，表示丢弃非异常记录。否则，如果状态为 on，那么丢弃异常记录。仅当启用 discard_records 属性时，才会启用此属性。

applyapriorinode 属性

您可以使用 Apriori 建模节点来生成 Apriori 模型块。该模型块的脚本名称为 *applyapriorinode*。有关编写建模节点自身脚本的详细信息，请参阅第 168 页的『apriorinode 属性』。

表 *154. applyapriorinode* 属性.

applyapriorinode 属性	值	属性描述
max_predictions	数字（整数）	
ignore_unmatached	标志	
allow_repeats	标志	
check_basket	NoPredictions Predictions NoCheck	
criterion	Confidence Support RuleSupport Lift Deployability	

applyassociationrulesnode 属性

关联规则建模节点可用于生成关联规则模型块。此模型块的脚本编制脚本名称为 *applyassociationrulesnode*。有关对建模节点自身进行脚本编制的更多信息，请参阅第 170 页的『associationrulesnode 属性』。

表 *155. applyassociationrulesnode* 属性

applyassociationrulesnode 属性	数据类型	属性描述
max_predictions	*integer*	可以对每个输入应用以进行评分的规则的最大数。
criterion	Confidence Rulesupport Lift Conditionsupport Deployability	选择用于确定规则强度的度量。
allow_repeats	布尔值	确定是否在分数中包含具有相同预测的规则。
check_input	NoPredictions Predictions NoCheck	

applyautoclassifiernode 属性

您可以使用"自动分类器"建模节点来生成"自动分类器"模型块。此模型块的脚本名称为 *applyautoclassifiernode*。有关编写建模节点自身脚本的详细信息，请参阅第 171 页的『autoclassifiernode 属性』。

表 *156. applyautoclassifiernode* 属性.

applyautoclassifiernode 属性	值	属性描述
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting HighestConfidence AverageRawPropensity	指定用于确定整体评分的方法。仅当选定的目标是标志字段时，才会应用此设置。
flag_voting_tie_selection	Random HighestConfidence RawPropensity	如果已选定投票方法，那么指定解决结的方法。仅当选定的目标是标志字段时，才会应用此设置。
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	指定用于确定整体评分的方法。仅当选定的目标是集合字段时，才会应用此设置。
set_voting_tie_selection	Random HighestConfidence	如果已选定投票方法，那么指定解决结的方法。仅当选定的目标是名义字段时，才会应用此设置。

applyautoclusternode 属性

您可以使用"自动聚类"建模节点来生成"自动聚类"模型块。该模型块的脚本名称为 *applyautoclusternode*。此模型块不具有任何其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 174 页的『autoclusternode 属性』。

applyautonumericnode 属性

您可以使用"自动数字"建模节点来生成"自动数字"模型块。此模型块的脚本名称为 *applyautonumericnode*。有关编写建模节点自身脚本的详细信息，请参阅第 175 页的『autonumericnode 属性』。

表 *157. applyautonumericnode* 属性.

applyautonumericnode 属性	值	属性描述
calculate_standard_error	标志	

applybayesnetnode 属性

您可以使用"贝叶斯网络"建模节点来生成"贝叶斯网络"模型块。该模型块的脚本名称为 *applybayesnetnode*。有关编写建模节点自身脚本的详细信息，请参阅第 176 页的『bayesnetnode 属性』。

表 *158. applybayesnetnode* 属性.

applybayesnetnode 属性	值	属性描述
all_probabilities	标志	
raw_propensity	标志	
adjusted_propensity	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

applyc50node 属性

您可以使用 C5.0 建模节点来生成 C5.0 模型块。该模型块的脚本名称为 *applyc50node*。有关编写建模节点自身脚本的详细信息，请参阅第 178 页的『c50node 属性』。

表 *159. applyc50node* 属性.

applyc50node 属性	值	属性描述
sql_generate	Never NoMissingValues	用于设置规则集执行期间使用的 SQL 生成选项。
calculate_conf	标志	启用 SQL 生成时可用；此属性将置信度的计算包括在生成的树中。
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

applycarmanode 属性

您可以使用 CARMA 建模节点来生成 CARMA 模型块。该模型块的脚本名称为 *applycarmanode*。此模型块不具有任何其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 179 页的『carmanode 属性』。

applycartnode 属性

您可以使用"C&R 树"建模节点来生成"C&R 树"模型块。该模型块的脚本名称为 *applycartnode*。有关编写建模节点自身脚本的详细信息，请参阅第 180 页的『cartnode 属性』。

表 *160. applycartnode* 属性.

applycartnode 属性	值	属性描述
enable_sql_generation	Never MissingValues NoMissingValues	用于设置规则集执行期间使用的 SQL 生成选项。
calculate_conf	标志	启用 SQL 生成时可用；此属性将置信度的计算包括在生成的树中。
display_rule_id	标志	在评分输出中添加一个字段，表示每个记录分配到的终端节点的标识。
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

applychaidnode 属性

您可以使用 CHAID 建模节点来生成 CHAID 模型块。该模型块的脚本名称为 *applychaidnode*。有关编写建模节点自身脚本的详细信息，请参阅第 182 页的『chaidnode 属性』。

表 *161. applychaidnode* 属性.

applychaidnode 属性	值	属性描述
enable_sql_generation	Never MissingValues	用于设置规则集执行期间使用的 SQL 生成选项。
calculate_conf	标志	
display_rule_id	标志	在评分输出中添加一个字段，表示每个记录分配到的终端节点的标识。
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

applycoxregnode 属性

您可以使用 Cox 建模节点来生成 Cox 模型块。该模型块的脚本名称为 *applycoxregnode*。有关编写建模节点自身脚本的详细信息，请参阅第 184 页的『coxregnode 属性』。

表 *162. applycoxregnode* 属性.

applycoxregnode 属性	值	属性描述
future_time_as	Intervals Fields	
time_interval	*number*	
num_future_times	*integer*	
time_field	字段	
past_survival_time	字段	
all_probabilities	标志	

表 *162. applycoxregnode* 属性 *(续).*

applycoxregnode 属性	值	属性描述
cumulative_hazard	标志	

applydecisionlistnode 属性

您可以使用"决策列表"建模节点来生成"决策列表"模型块。该模型块的脚本名称为 *applydecisionlistnode*。有关编写建模节点自身脚本的详细信息，请参阅第 186 页的『decisionlistnode 属性』。

表 *163. applydecisionlistnode* 属性.

applydecisionlistnode 属性	值	属性描述
enable_sql_generation	标志	值为 true 时，IBM SPSS Modeler 会尝试将决策列表模型回推到 SQL。
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

applydiscriminantnode 属性

您可以使用"判别"建模节点来生成"判别"模型块。该模型块的脚本名称为 *applydiscriminantnode*。有关编写建模节点自身脚本的详细信息，请参阅第 187 页的『discriminantnode 属性』。

表 *164. applydiscriminantnode* 属性.

applydiscriminantnode 属性	值	属性描述
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

applyextension 属性

扩展模型节点可用于生成扩展模型块。此模型块的脚本编制名称为 *applyextension*。有关对建模节点自身进行脚本编制的更多信息，请参阅第 188 页的『extensionmodelnode 属性』。

Python for Spark 示例

```
#### script example for Python for Spark
applyModel = stream.findByType("extension_apply", None)

score_script = """
import json
import spss.pyspark.runtime
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import DenseVector
from pyspark.mllib.tree import DecisionTreeModel
from pyspark.sql.types import StringType, StructField

cxt = spss.pyspark.runtime.getContext()
```

```
if cxt.isComputeDataModelOnly():
    _schema = cxt.getSparkInputSchema()
    _schema.fields.append(StructField("Prediction", StringType(), nullable=True))
    cxt.setSparkOutputSchema(_schema)
else:
    df = cxt.getSparkInputData()

    _modelPath = cxt.getModelContentToPath("TreeModel")
    metadata = json.loads(cxt.getModelContentToString("model.dm"))

    schema = df.dtypes[:]
    target = "Drug"
    predictors = ["Age","BP","Sex","Cholesterol","Na","K"]

    lookup = {}
    for i in range(0,len(schema)):
        lookup[schema[i][0]] = i

    def row2LabeledPoint(dm,lookup,target,predictors,row):
        target_index = lookup[target]
        tval = dm[target_index].index(row[target_index])
        pvals = []
        for predictor in predictors:
            predictor_index = lookup[predictor]
            if isinstance(dm[predictor_index],list):
                pval = row[predictor_index] in dm[predictor_index] and dm[predictor_index].index(row[predictor_index]) or -1
            else:
                pval = row[predictor_index]
            pvals.append(pval)
        return LabeledPoint(tval, DenseVector(pvals))

    # convert dataframe to an RDD containing LabeledPoint
    lps = df.rdd.map(lambda row: row2LabeledPoint(metadata,lookup,target,predictors,row))
    treeModel = DecisionTreeModel.load(cxt.getSparkContext(), _modelPath);
    # score the model, produces an RDD containing just double values
    predictions = treeModel.predict(lps.map(lambda lp: lp.features))

    def addPrediction(x,dm,lookup,target):
        result = []
        for _idx in range(0, len(x[0])):
            result.append(x[0][_idx])
        result.append(dm[lookup[target]][int(x[1])])
        return result

    _schema = cxt.getSparkInputSchema()
    _schema.fields.append(StructField("Prediction", StringType(), nullable=True))
    rdd2 = df.rdd.zip(predictions).map(lambda x:addPrediction(x, metadata, lookup, target))
    outDF = cxt.getSparkSQLContext().createDataFrame(rdd2, _schema)

    cxt.setSparkOutputData(outDF)
"""
applyModel.setPropertyValue("python_syntax", score_script)
```

R 示例

```
#### script example for R
applyModel.setPropertyValue("r_syntax", """
result<-predict(modelerModel,newdata=modelerData)
modelerData<-cbind(modelerData,result)
var1<-c(fieldName="NaPrediction",fieldLabel="",fieldStorage="real",fieldMeasure="",
fieldFormat="",fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

表 *165. applyextension* 属性

applyextension 属性	**值**	**属性描述**
r_syntax	*string*	这是用于进行模型评分的 R 脚本语法。
python_syntax	*string*	这是用于进行模型评分的 Python 脚本语法。
use_batch_size	*flag*	启用使用批处理。
batch_size	*integer*	指定要包含在每个批处理中的数据记录数。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_missing	*flag*	此选项用于将缺失值转换为 R NA 值。
convert_datetime	*flag*	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。

applyfactornode 属性

您可以使用"PCA/因子"建模节点来生成"PCA/因子"模型块。该模型块的脚本名称为 *applyfactornode*。此模型块不具有任何其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 191 页的『factornode 属性』。

applyfeatureselectionnode 属性

您可以使用"特征选择"建模节点来生成"特征选择"模型块。该模型块的脚本名称为 *applyfeatureselectionnode*。有关编写建模节点自身脚本的详细信息，请参阅第 192 页的『featureselectionnode 属性』。

表 *166. applyfeatureselectionnode* 属性.

applyfeatureselectionnode 属性	**值**	**属性描述**
selected_ranked_fields		指定要在模型浏览器中检查哪些已排秩的字段。
selected_screened_fields		指定要在模型浏览器中检查哪些已筛选的字段。

applygeneralizedlinearnode 属性

您可以使用"广义线性 (genlin)"建模节点来生成"广义线性"模型块。该模型块的脚本名称为 *applygeneralizedlinearnode*。有关编写建模节点自身脚本的详细信息，请参阅第 194 页的『genlinnode 属性』。

表 *167. applygeneralizedlinearnode* 属性.

applygeneralizedlinearnode 属性	**值**	**属性描述**
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

applyglmmnode 属性

您可以使用 GLMM 建模节点来生成 GLMM 模型块。该模型块的脚本名称为 *applyglmmnode*。有关编写建模节点自身脚本的详细信息，请参阅第 197 页的『glmmnode 属性』。

表 *168. applyglmmnode* 属性.

applyglmmnode 属性	值	属性描述
confidence	onProbability onIncrease	计算评分置信度值的基础：最高预测概率或者最高与次高预测概率之差。
score_category_probabilities	标志	如果设置为 true，那么为分类目标生成预测概率。为每个类别创建一个字段。缺省值为 False。
max_categories	*integer*	要预测概率的最大类别数目。仅当 score_category_probabilities 为 True 时才使用。
score_propensity	标志	如果设置为 True，那么为具有标志目标的模型生成原始倾向评分（"True"结果的可能性）。如果分区处于有效，那么模型还会根据测试分区产生调整后的倾向评分。缺省值为 False。
enable_sql_generation	udf native	用于在流执行期间设置 SQL 生成选项。 选项包括推到数据库并使用 SPSS® Modeler Server 评分适配器（如果连接到安装了评分适配器的数据库）进行评分，或者在 SPSS Modeler 内评分。 缺省值为 udf。

applygle 属性

GLE 建模节点可用于生成 GLE 模型块。此模型块的脚本编制名称为 *applygle*。有关对建模节点自身进行脚本编制的更多信息，请参阅第 200 页的『gle 属性』。

表 *169. applygle* 属性

applygle 属性	值	属性描述
enable_sql_generation	udf native	用于在流执行期间设置 SQL 生成选项。 选择后推到数据库并使用 SPSS Modeler Server 评分适配器（如果连接到安装了评分适配器的数据库）进行评分，或者在 SPSS Modeler 内评分。

applykmeansnode 属性

您可以使用 K-Means 建模节点来生成 K-Means 模型块。该模型块的脚本名称为 *applykmeansnode*。此模型块不具有任何其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 205 页的『kmeansnode 属性』。

applyknnnode 属性

您可以使用 KNN 建模节点来生成 KNN 模型块。此模型块的脚本名称是 *applyknnnode*。有关编写建模节点自身脚本的详细信息，请参阅第 206 页的『knnnode 属性』。

表 *170. applyknnnode* 属性.

applyknnnode 属性	**值**	**属性描述**
all_probabilities	标志	
save_distances	标志	

applykohonennode 属性

您可以使用 Kohonen 建模节点来生成 Kohonen 模型块。该模型块的脚本名称为 *applykohonennode*。此模型块不具有任何其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 178 页的『c50node 属性』。

applylinearnode 属性

您可以使用"线性"建模节点来生成"线性"模型块。该模型块的脚本名称为 *applylinearnode*。有关编写建模节点自身脚本的详细信息，请参阅第 208 页的『linearnode 属性』。

表 *171. applylinearnode* 属性.

linear 属性	**值**	**属性描述**
use_custom_name	标志	
custom_name	字符串	
enable_sql_generation	udf native puresql	用于在流执行期间设置 SQL 生成选项。 选项包括推到数据库并使用 SPSS® Modeler Server 评分适配器（如果连接到安装了评分适配器的数据库）进行评分、在 SPSS Modeler 内评分，或者推到数据库并使用 SQL 评分。 缺省值为 udf。

applylinearasnode 属性

Linear-AS 建模节点可用于生成 Linear-AS 模型块。 此模型块的脚本编制名称为 *applylinearasnode*。 有关编写建模节点自身脚本的详细信息，请参阅第 210 页的『linearasnode 属性』。

表 *172. applylinearasnode* 属性

applylinearasnode 属性	**值**	**属性描述**
enable_sql_generation	udf native	缺省值为 udf。

applylogregnode 属性

您可以使用"Logistic 回归模型"建模节点来生成"Logistic 回归模型"模型块。该模型块的脚本名称为 *applylogregnode*。有关编写建模节点自身脚本的详细信息，请参阅第 211 页的『logregnode 属性』。

表 *173. applylogregnode* 属性.

applylogregnode 属性	值	属性描述
calculate_raw_propensities	标志	
calculate_conf	*flag*	
enable_sql_generation	*flag*	

applylsvmnode 属性

LSVM 建模节点可用于生成 LSVM 模型块。此模型块的脚本编制名称为 *applylsvmnode*。有关对建模节点自身进行脚本编制的更多信息，请参阅第 215 页的『lsvmnode 属性』。

表 *174. applylsvmnode* 属性

applylsvmnode 属性	值	属性描述
calculate_raw_propensities	*flag*	指定是否计算原始倾向得分。
enable_sql_generation	udf native	指定是使用"评分适配器"（如果已安装）或者在流程中评分，还是在数据库外部评分。

applyneuralnetnode 属性

您可以使用"神经网络"建模节点来生成"神经网络"模型块。该模型块的脚本名称为 *applyneuralnetnode*。有关编写建模节点自身脚本的详细信息，请参阅第 215 页的『neuralnetnode 属性』。

注意：在此发行版中提供了具有增强功能的新版本的神经网络模型块，并将在下一节 (*applyneuralnetwork*) 中进行介绍。尽管先前版本仍然可用，但我们建议您更新脚本以使用新的版本。此处保留了先前版本的详细信息以供参考，但会在将来的发行版中不再支持。

表 *175. applyneuralnetnode* 属性.

applyneuralnetnode 属性	值	属性描述
calculate_conf	标志	启用 SQL 生成时可用；此属性将置信度的计算包括在生成的树中。
enable_sql_generation	标志	
nn_score_method	Difference SoftMax	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

applyneuralnetworknode 属性

您可以使用"神经网络"建模节点来生成"神经网络"模型块。该模型块的脚本名称为 *applyneuralnetworknode*。有关编写建模节点自身脚本的详细信息，请参阅 neuralnetworknode 属性。

表 *176. applyneuralnetworknode* 属性

applyneuralnetworknode 属性	**值**	**属性描述**
use_custom_name	*flag*	
custom_name	*string*	
confidence	onProbability onIncrease	
score_category_probabilities	*flag*	
max_categories	*number*	
score_propensity	*flag*	
enable_sql_generation	udf native puresql	用于在流执行期间设置 SQL 生成选项。 选项包括推到数据库并使用 SPSS® Modeler Server 评分适配器（如果连接到安装了评分适配器的数据库）进行评分、在 SPSS Modeler 内评分，或者推到数据库并使用 SQL 评分。 缺省值为 udf。

applyocsvmnode 属性

单类 SVM 节点可用于生成单类 SVM 模型块。此模型块的脚本编制名称为 *applyocsvmnode*。该模型块不存在其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 319 页的『ocsvmnode 属性』。

applyquestnode 属性

您可以使用 QUEST 建模节点来生成 QUEST 模型块。该模型块的脚本名称为 *applyquestnode*。有关编写建模节点自身脚本的详细信息，请参阅第 219 页的『questnode 属性』。

表 *177. applyquestnode* 属性.

applyquestnode 属性	**值**	**属性描述**
enable_sql_generation	Never MissingValues NoMissingValues	用于设置规则集执行期间使用的 SQL 生成选项。
calculate_conf	标志	
display_rule_id	标志	在评分输出中添加一个字段，表示每个记录分配到的终端节点的标识。
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

applyr 属性

您可以使用"R 构建"节点来生成 R 模型块。此模型块的脚本编制名称为 *applyr*。有关编写建模节点自身脚本的详细信息，请参阅第 177 页的『buildr 属性』。

表 *178. applyr* 属性

applyr 属性	值	属性描述
score_syntax	*string*	这是用于进行模型评分的 R 脚本语法。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_datetime	*flag*	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。
convert_missing	*flag*	此选项用于将缺失值转换为 R NA 值。
use_batch_size	*flag*	支持使用批处理
batch_size	*integer*	指定要包含在每个批处理中的数据记录数

applyrandomtrees 属性

"随机树"建模节点可用于生成"随机树"模型块。此模型块的脚本编制名称为 *applyrandomtrees*。有关对建模节点自身进行脚本编制的更多信息，请参阅第 221 页的『randomtrees 属性』。

表 *179. applyrandomtrees* 属性

applyrandomtrees 属性	值	属性描述
calculate_conf	*flag*	此属性将置信度计算包含在生成的树中。
enable_sql_generation	udf native	用于在流执行期间设置 SQL 生成选项。 选择后推到数据库并使用 SPSS Modeler Server 评分适配器（如果连接到安装了评分适配器的数据库）进行评分，或者在 SPSS Modeler 内评分。

asapplyregressionnode 属性

您可以使用"线性回归"建模节点来生成"线性回归"模型块。该模型块的脚本名称为 *applyregressionnode*。此模型块不具有任何其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 222 页的『regressionnode 属性』。

applyselflearningnode 属性

您可以使用"自学响应模型 (SLRM)"建模节点来生成 SLRM 模型块。该模型块的脚本名称为 *applyselflearningnode*。有关编写建模节点自身脚本的详细信息，请参阅第 226 页的『slrmnode 属性』。

表 *180. applyselflearningnode* 属性.

applyselflearningnode 属性	值	属性描述
max_predictions	*number*	
randomization	*number*	

表 *180. applyselflearningnode* 属性 *(续).*

applyselflearningnode 属性	**值**	**属性描述**
scoring_random_seed	*number*	
sort	ascending descending	指定是先显示评分最高还是最低的报价。
model_reliability	标志	将"设置"选项卡中的模型可靠性选项考虑在内。

applysequencenode 属性

您可以使用"序列"建模节点来生成"序列"模型块。该模型块的脚本名称为 *applysequencenode*。此模型块不具有任何其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 224 页的『sequencenode 属性』。

applysvmnode 属性

您可以使用 SVM 建模节点来生成 SVM 模型块。该模型块的脚本名称为 *applysvmnode*。有关编写建模节点自身脚本的详细信息，请参阅第 230 页的『svmnode 属性』。

表 *181. applysvmnode* 属性.

applysvmnode 属性	**值**	**属性描述**
all_probabilities	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

applystpnode 属性

STP 建模节点可用于生成关联的模型块，此模型块在输出查看器中显示模型输出。此模型块的脚本编制名称为 *applystpnode*。有关对建模节点自身进行脚本编制的更多信息，请参阅第 227 页的『stpnode 属性』。

表 *182. applystpnode* 属性

applystpnode 属性	**数据类型**	**属性描述**
uncertainty_factor	布尔值	最小值为 0，最大值为 100。

applytcmnode 属性

时间因果建模 (TCM) 建模节点可用于生成 TCM 模型块。 此模型块的脚本编制名称为 *applytcmnode*。 有关对建模节点自身进行脚本编制的更多信息，请参阅第 231 页的『tcmnode 属性』。

表 *183. applytcmnode* 属性

applytcmnode 属性	**值**	**属性描述**
ext_future	布尔值	
ext_future_num	*integer*	
noise_res	布尔值	
conf_limits	布尔值	
target_fields	列表	
target_series	列表	

applyts 属性

您可以使用"时间序列"建模节点来生成"时间序列"模型块。此模型块的脚本编制名称为 *applyts*。有关对建模节点自身进行脚本编制的更多信息，请参阅第 235 页的『ts 属性』。

表 *184. applyts* 属性

applyts 属性	值	属性描述
extend_records_into_future	布尔值	
ext_future_num	*integer*	
compute_future_values_input	布尔值	
forecastperiods	*integer*	
noise_res	布尔值	
conf_limits	布尔值	
target_fields	列表	
target_series	列表	
includeTargets	字段	

applytimeseriesnode 属性（不推荐）

您可以使用"时间序列"建模节点来生成"时间序列"模型块。该模型块的脚本名称为 *applytimeseriesnode*。有关编写建模节点自身脚本的详细信息，请参阅第 239 页的『timeseriesnode 属性（不推荐）』。

表 *185. applytimeseriesnode* 属性.

applytimeseriesnode 属性	值	属性描述
calculate_conf	标志	
calculate_residuals	标志	

applytreeas 属性

Tree-AS 建模节点可用于生成 Tree-AS 模型块。 此模型块的脚本编制名称为 *applytreenas*。有关对建模节点自身进行脚本编制的更多信息，请参阅第 241 页的『treeas 属性』。

表 *186. applytreeas* 属性

applytreeas 属性	值	属性描述
calculate_conf	*flag*	此属性将置信度计算包含在生成的树中。
display_rule_id	*flag*	在评分输出中添加一个字段，表示每个记录分配到的终端节点的标识。
enable_sql_generation	udf native	用于在流执行期间设置 SQL 生成选项。 选择后推到数据库并使用 SPSS Modeler Server 评分适配器（如果连接到安装了评分适配器的数据库）进行评分，或者在 SPSS Modeler 内评分。

applytwostepnode 属性

您可以使用"二阶"建模节点来生成"二阶"模型块。该模型块的脚本名称为 *applytwostepnode*。此模型块不具有任何其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 243 页的『twostepnode 属性』。

applytwostepAS 属性

二阶 AS 建模节点可用于生成二阶 AS 模型块。此模型块的脚本编制名称为 *applytwostepAS*。有关编写建模节点自身脚本的详细信息，请参阅第 244 页的『twostepAS 属性』。

表 *187. applytwostepAS* 属性

applytwostepAS 属性	值	属性描述
enable_sql_generation	udf native	用于在流执行期间设置 SQL 生成选项。 选项包括推到数据库并使用 SPSS® Modeler Server 评分适配器（如果连接到安装了评分适配器的数据库）进行评分，或者在 SPSS Modeler 内评分。 缺省值为 udf。

applyxgboosttreenode 属性

XGBoost Tree 节点可用于生成 XGBoost Tree 模型块。此模型块的脚本编制名称为 *applyxgboosttreenode*。该模型块不存在其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 325 页的『xgboosttreenode 属性』。

applyxgboostlinearnode 属性

XGBoost Linear 节点可用于生成 XGBoost Linear 模型块。此模型块的脚本编制名称为 *applyxgboostlinearnode*。该模型块不存在其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 324 页的『xgboostlinearnode 属性』。

第 15 章 数据库建模节点属性

IBM SPSS Modeler 支持与多家数据库供应商的数据挖掘和建模工具集成，这包括 Microsoft SQL Server Analysis Services、Oracle Data Mining、和 IBM Netezza® Analytics。您可以使用 IBM SPSS Modeler 应用程序自有的数据库算法来构建模型并对模型进行评分。还可以使用本节介绍的属性通过编写脚本来构建和操纵数据库模型。

例如，以下脚本摘录说明了如何使用 IBM SPSS Modeler 脚本编制界面创建 Microsoft Decision Trees 模型：

```
stream = modeler.script.stream()
msbuilder = stream.createAt("mstreenode", "MSBuilder", 200, 200)

msbuilder.setPropertyValue("analysis_server_name", 'localhost')
msbuilder.setPropertyValue("analysis_database_name", 'TESTDB')
msbuilder.setPropertyValue("mode", 'Expert')
msbuilder.setPropertyValue("datasource", 'LocalServer')
msbuilder.setPropertyValue("target", 'Drug')
msbuilder.setPropertyValue("inputs", ['Age', 'Sex'])
msbuilder.setPropertyValue("unique_field", 'IDX')
msbuilder.setPropertyValue("custom_fields", True)
msbuilder.setPropertyValue("model_name", 'MSDRUG')

typenode = stream.findByType("type", None)
stream.link(typenode, msbuilder)
results = []
msbuilder.run(results)
msapplier = stream.createModelApplierAt(results[0], "Drug", 200, 300)
tablenode = stream.createAt("table", "Results", 300, 300)
stream.linkBetween(msapplier, typenode, tablenode)
msapplier.setPropertyValue("sql_generate", True)
tablenode.run([])
```

Microsoft 建模的节点属性

Microsoft 建模节点属性

公共属性

Microsoft 数据库建模节点的公共属性如下所示。

表 *188.* 公共 *Microsoft* 节点属性

公共 Microsoft 节点属性	值	属性描述
analysis_database_name	*string*	Analysis Services 数据库的名称。
analysis_server_name	*string*	Analysis Services 主机的名称。
use_transactional_data	*flag*	指定输入数据是采用表格式还是事务格式。
inputs	列表	表数据的输入字段。
target	字段	预测字段（不适用于"MS 聚类"或"序列聚类"节点）。
unique_field	字段	键字段。
msas_parameters	结构化	算法参数。有关更多信息，请参阅第 265 页的『算法参数』主题。

表 188. 公共 *Microsoft* 节点属性 *(续)*

公共 Microsoft 节点属性	值	属性描述
with_drillthrough	*flag*	"进行深入钻取"选项。

MS 决策树

没有为 mstreenode 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

MS 聚类

没有为 msclusternode 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

MS 关联规则

以下特定属性可用于类型为 msassocnode 的节点：

表 189. *msassocnode* 属性

msassocnode 属性	值	属性描述
id_field	字段	标识数据中的每项事务。
trans_inputs	列表	事务数据的输入字段。
transactional_target	字段	预测字段（事务数据）。

MS 朴素贝叶斯

没有为 msbayesnode 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

MS 线性回归

没有为 msregressionnode 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

MS 神经网络

没有为 msneuralnetworknode 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

MS Logistic 回归

没有为 mslogisticnode 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

MS 时间序列

没有为 mstimeseriesnode 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

MS 序列聚类

以下特定属性可用于类型为 mssequenceclusternode 的节点：

表 190. *mssequenceclusternode* 属性

mssequenceclusternode 属性	值	属性描述
id_field	字段	标识数据中的每项事务。
input_fields	列表	事务数据的输入字段。
sequence_field	字段	序列标识。

表 *190. mssequenceclusternode* 属性 *(续)*

mssequenceclusternode 属性	**值**	**属性描述**
target_field	字段	预测字段（表数据）。

算法参数

每种 Microsoft 数据库模型类型均有可使用 msas_parameters 属性来设置的特定参数，例如：

```
stream = modeler.script.stream()
msregressionnode = stream.findByType("msregression", None)
msregressionnode.setPropertyValue("msas_parameters", [["MAXIMUM_INPUT_ATTRIBUTES", 255],
["MAXIMUM_OUTPUT_ATTRIBUTES", 255]])
```

这些参数源自 SQL Server。要查看每个节点的相关参数，请执行如下操作：

1. 将数据库源节点放入画布中。
2. 打开该数据库源节点。
3. 从**数据源**下拉列表中选择一个有效源。
4. 从**表名称**列表中选择一个有效表。
5. 单击**确定**以关闭该数据库源节点。
6. 附加要列出其属性的 Microsoft 数据库建模节点。
7. 打开该数据库建模节点。
8. 选择**专家**选项卡。

此时会显示该节点的可用 msas_parameters 属性。

Microsoft 模型块属性

使用 Microsoft 数据库建模节点创建的模型块具有下列属性。

MS 决策树

表 *191. MS* 决策树属性

applymstreenode 属性	**值**	**描述**
analysis_database_name	*string*	可以直接在流中对此节点进行评分。 此属性用于标识 Analysis Services 数据库的名称。
analysis_server_name	*string*	Analysis 服务器主机的名称。
datasource	*string*	SQL Server ODBC 数据源名称 (DSN) 的名称。
sql_generate	*flag*	启用 SQL 生成。

MS 线性回归

表 *192. MS* 线性回归属性

applymsregressionnode 属性	**值**	**描述**
analysis_database_name	*string*	可以直接在流中对此节点进行评分。 此属性用于标识 Analysis Services 数据库的名称。
analysis_server_name	*string*	Analysis 服务器主机的名称。

MS 神经网络

表 *193. MS* 神经网络属性

applymsneuralnetworknode 属性	**值**	**描述**
analysis_database_name	*string*	可以直接在流中对此节点进行评分。 此属性用于标识 Analysis Services 数据库的名称。
analysis_server_name	*string*	Analysis 服务器主机的名称。

MS Logistic 回归

表 *194. MS Logistic* 回归属性

applymslogisticnode 属性	**值**	**描述**
analysis_database_name	*string*	可以直接在流中对此节点进行评分。 此属性用于标识 Analysis Services 数据库的名称。
analysis_server_name	*string*	Analysis 服务器主机的名称。

MS 时间序列

表 *195. MS* 时间序列属性

applymstimeseriesnode 属性	**值**	**描述**
analysis_database_name	*string*	可以直接在流中对此节点进行评分。 此属性用于标识 Analysis Services 数据库的名称。
analysis_server_name	*string*	Analysis 服务器主机的名称。
start_from	new_prediction historical_ prediction	指定是进行未来预测还是历史预测。
new_step	*number*	定义未来预测的开始时间段。
historical_step	*number*	定义历史预测的开始时间段。
end_step	*number*	定义预测结束时间段。

MS 序列聚类

表 *196. MS* 序列聚类属性

applymssequenceclusternode 属性	**值**	**描述**
analysis_database_name	*string*	可以直接在流中对此节点进行评分。 此属性用于标识 Analysis Services 数据库的名称。
analysis_server_name	*string*	Analysis 服务器主机的名称。

Oracle 建模的节点属性

Oracle 建模节点属性

Oracle 数据库建模节点的公共属性如下所示。

表 *197.* 公共 *Oracle* 节点属性.

公共 Oracle 节点属性	值(V)	属性说明
target	字段	
inputs	字段的列表	
partition	字段	此域用于将数据分区为不同的样本，以用于模型构建的训练、检验和验证阶段。
datasource		
username		
password		
epassword		
use_model_name	*flag*	
model_name	*string*	新模型的定制名称。
use_partitioned_data	*flag*	如果定义了分区字段，则此选项可确保仅训练分区的数据用于构建模型。
unique_field	字段	
auto_data_prep	*flag*	启用或禁用 Oracle 自动数据准备功能（仅适用于 11g 数据库）。
costs	结构化	格式如下的结构化属性： [[drugA drugB 1.5] [drugA drugC 2.1]]，其中 [] 中的自变量是实际预测成本。
mode	Simple Expert	如在各个节点属性中注释的那样，如果设置为 Simple，会导致忽略某些属性。
use_prediction_probability	*flag*	
prediction_probability	*string*	
use_prediction_set	*flag*	

Oracle 朴素贝叶斯

类型为 oranbnode 的节点的可用属性如下所示：

表 *198. oranbnode* 属性

oranbnode 属性	值(V)	属性说明
singleton_threshold	数字	0.0–1.0.*
pairwise_threshold	数字	0.0–1.0.*
priors	Data Equal Custom	
custom_priors	结构化	格式如下的结构化属性： set :oranbnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* 如果 mode 设置为 Simple，则忽略属性。

Oracle Adaptive Bayes

类型为 oraabnnode 的节点的可用属性如下所示：

表 *199. oraabnnode* 属性

oraabnnode 属性	**值(V)**	**属性说明**
model_type	SingleFeature MultiFeature NaiveBayes	
use_execution_time_limit	*flag*	*
execution_time_limit	整数	值必须大于 0。*
max_naive_bayes_predictors	整数	值必须大于 0。*
max_predictors	整数	值必须大于 0。*
priors	Data Equal Custom	
custom_priors	结构化	格式如下的结构化属性： set :oraabnnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* 如果 mode 设置为 Simple，则忽略属性。

Oracle 支持向量机

类型为 orasvmnode 的节点的可用属性如下所示：

表 *200. orasvmnode* 属性

orasvmnode 属性	**值(V)**	**属性说明**
active_learning	Enable Disable	
kernel_function	Linear Gaussian System	
normalization_method	zscore minmax none	
kernel_cache_size	整数	仅适用于高斯内核。值必须大于 0。*
convergence_tolerance	数字	值必须大于 0。*
use_standard_deviation	*flag*	仅适用与高斯内核。*
standard_deviation	数字	值必须大于 0。*
use_epsilon	*flag*	仅适用于回归模型。*
epsilon	数字	值必须大于 0。*
use_complexity_factor	*flag*	*
complexity_factor	数字	*
use_outlier_rate	*flag*	仅适用于单类变体。*

表 *200. orasvmnode* 属性 *(续)*

orasvmnode 属性	**值(V)**	**属性说明**
outlier_rate	数字	仅适用于单类变体。0.0–1.0*
权重	Data Equal Custom	
custom_weights	结构化	格式如下的结构化属性： set :orasvmnode.custom_weights = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* 如果 mode 设置为 Simple，则忽略属性。

Oracle 广义线性模型

类型为 oraglmnode 的节点的可用属性如下所示：

表 *201. oraglmnode* 属性

oraglmnode 属性	**值(V)**	**属性说明**
normalization_method	zscore minmax none	
missing_value_handling	ReplaceWithMean UseCompleteRecords	
use_row_weights	*flag*	*
row_weights_field	*field*	*
save_row_diagnostics	*flag*	*
row_diagnostics_table	字符串	*
coefficient_confidence	数字	*
use_reference_category	*flag*	*
reference_category	字符串	*
ridge_regression	Auto Off On	*
parameter_value	数字	*
vif_for_ridge	*flag*	*

* 如果 mode 设置为 Simple，则忽略属性。

Oracle 决策树

类型为 oradecisiontreenode 的节点的可用属性如下所示：

表 *202. oradecisiontreenode* 属性

oradecisiontreenode 属性	**值(V)**	**属性说明**
use_costs	*flag*	

表 202. *oradecisiontreenode* 属性 (续)

oradecisiontreenode 属性	**值(V)**	**属性说明**
impurity_metric	熵 (Entropy) Gini	
term_max_depth	整数	2–20.*
term_minpct_node	数字	0.0–10.0.*
term_minpct_split	数字	0.0–20.0.*
term_minrec_node	整数	值必须大于 0。*
term_minrec_split	整数	值必须大于 0。*
display_rule_ids	*flag*	*

* 如果 mode 设置为 Simple，则忽略属性。

Oracle O-Cluster

类型为 oraoclusternode 的节点的可用属性如下所示：

表 203. *oraoclusternode* 属性

oraoclusternode 属性	**值(V)**	**属性说明**
max_num_clusters	整数	值必须大于 0。
max_buffer	整数	值必须大于 0。*
sensitivity	数字	0.0–1.0.*

* 如果 mode 设置为 Simple，则忽略属性。

Oracle KMeans

类型为 orakmeansnode 的节点的可用属性如下所示：

表 204. *orakmeansnode* 属性

orakmeansnode 属性	**值(V)**	**属性说明**
num_clusters	整数	值必须大于 0。
normalization_method	zscore minmax none	
distance_function	Euclidean Cosine	
迭代	整数	0–20.*
conv_tolerance	数字	0.0–0.5.*
split_criterion	Variance Size	缺省值为 Variance。*
num_bins	整数	值必须大于 0。*
block_growth	整数	1–5.*
min_pct_attr_support	数字	0.0–1.0.*

* 如果 mode 设置为 Simple，则忽略属性。

Oracle NMF

类型为 oranmfnode 的节点的可用属性如下所示：

表 *205. oranmfnode* 属性

oranmfnode 属性	**值(V)**	**属性说明**
normalization_method	minmax none	
use_num_features	*flag*	*
num_features	整数	0-1。缺省值由算法根据数据估计得出。*
random_seed	数字	*
num_iterations	整数	0–500.*
conv_tolerance	数字	0.0–0.5.*
display_all_features	*flag*	*

* 如果 mode 设置为 Simple，则忽略属性。

Oracle Apriori

类型为 oraapriorinode 的节点的可用属性如下所示：

表 *206. oraapriorinode* 属性

oraapriorinode 属性	**值(V)**	**属性说明**
content_field	*field*	
id_field	*field*	
max_rule_length	整数	2–20。
min_confidence	数字	0.0–1.0。
min_support	数字	0.0–1.0。
use_transactional_data	*flag*	

Oracle 最小描述长度 (MDL)

没有为类型为 oramdlnode 的节点定义具体属性。请参阅本章节开头部分的通用 Oracle 属性。

Oracle 属性重要性 (AI)

类型为 oraainode 的节点的可用属性如下所示：

表 *207. oraainode* 属性

oraainode 属性	**值(V)**	**属性说明**
custom_fields	*flag*	如果为 true，则允许您为当前节点指定目标、输入和其他字段。如果为 false，则使用来自上游类型节点的当前设置。
selection_mode	ImportanceLevel ImportanceValue TopN	

表 207. *oraainode* 属性 (续)

oraainode 属性	值(V)	属性说明
select_important	*flag*	在 selection_mode 设置为 ImportanceLevel 时，指定是否选择"重要"字段。
important_label	字符串	指定"重要"排秩的标签。
select_marginal	*flag*	在 selection_mode 设置为 ImportanceLevel 时，指定是否选择"边际"字段。
marginal_label	字符串	指定"边际"排秩的标签。
important_above	数字	0.0–1.0。
select_unimportant	*flag*	在 selection_mode 设置为 ImportanceLevel 时，指定是否选择"不重要"字段。
unimportant_label	字符串	指定"不重要"排秩的标签。
unimportant_below	数字	0.0–1.0。
importance_value	数字	在 selection_mode 设置为 ImportanceValue 时，指定要使用的分界值。接受从 0 到 100 的值。
top_n	数字	在 selection_mode 设置为 TopN 时，指定要使用的分界值。接受从 0 到 1000 的值。

Oracle 模型块属性

使用 Oracle 模型创建的模型块具有下列属性。

Oracle 朴素贝叶斯

没有为 applyoranbnode 类型的节点定义具体属性。

Oracle Adaptive Bayes

没有为 applyoraabnnode 类型的节点定义具体属性。

Oracle 支持向量机

没有为 applyorasvmnode 类型的节点定义具体属性。

Oracle 决策树

类型为 applyoradecisiontreenode 的节点的可用属性如下所示：

表 208. *applyoradecisiontreenode* 属性

applyoradecisiontreenode 属性	值	属性描述
use_costs	*flag*	
display_rule_ids	*flag*	

Oracle O-Cluster

没有为 applyoraoclusternode 类型的节点定义具体属性。

Oracle KMeans

没有为 applyorakmeansnode 类型的节点定义具体属性。

Oracle NMF

下列属性用于 applyoranmfnode 类型的节点：

表 209. *applyoranmfnode* 属性

applyoranmfnode 属性	值	属性描述
display_all_features	*flag*	

Oracle Apriori

该模型块不能应用于脚本。

Oracle MDL

该模型块不能应用于脚本。

IBM Netezza Analytics 建模节点属性

Netezza 建模节点属性

IBM Netezza 数据库建模节点的公共属性如下所示。

表 210. 公共 *Netezza* 节点属性.

公共 Netezza 节点属性	值(V)	属性说明
custom_fields	*flag*	如果为 true，则允许您为当前节点指定目标、输入和其他字段。如果为 false，则使用来自上游类型节点的当前设置。
inputs	*[field1 ... fieldN]*	模型所使用的输入或预测变量字段。
target	字段	目标字段（连续或分类）。
record_id	字段	要用作唯一记录标识的字段。
use_upstream_connection	*flag*	如果为 true（缺省），那么连接详细信息在上游节点中指定。在指定了 move_data_to_connection 时不使用。
move_data_connection	*flag*	如果为 true，则将数据移动到由 connection 指定的数据库。在指定了 use_upstream_connection 时不使用。
connection	结构化	这是用于存储模型的 Netezza 数据库的连接字符串。格式如下的结构化属性： ['odbc' '<dsn>' '<username>' '<psw>' '<catname>' '<conn_attribs>' [true|false]] 其中： <dsn> 是数据源名称 <username> 和 <psw> 是数据库的用户名和密码 <catname> 是目录名称 <conn_attribs> 是连接属性 true | false 指示是否需要密码。
table_name	*string*	这是用于存储模型的数据库表的名称。

表 *210.* 公共 *Netezza* 节点属性 *(续).*

公共 Netezza 节点属性	值(V)	属性说明
use_model_name	*flag*	如果为 true，使用由 model_name 指定的名称作为模型名称，否则采用系统创建的模型名称。
model_name	*string*	新模型的定制名称。
include_input_fields	*flag*	如果为 true，向下游传递所有输入字段，否则仅传递模型产生的 record_id 和字段。

Netezza 决策树

类型为 netezzadectreenode 的节点的可用属性如下所示：

表 *211. netezzadectreenode* 属性

netezzadectreenode 属性	值(V)	属性说明
impurity_measure	熵 (Entropy) Gini	对杂质的测量，用于评估树的最佳拆分位置。
max_tree_depth	整数	树可以增长到的最大级别数。缺省值为 62（最大可能值）。
min_improvement_splits	数字	进行分割前必须满足的最低杂质改进。缺省值为 0.01。
min_instances_split	整数	可以进行分割前余下的最小未分割记录数。缺省值为 2（最小可能值）。
权重	结构化	各个类的相对权重。格式如下的结构化属性： set :netezza_dectree.weights = [[drugA 0.3][drugB 0.6]] 缺省情况是所有类的权重均为 1。
pruning_measure	Acc wAcc	缺省值为 Acc（准确性）。如果要在应用修剪时将类权重考虑在内，可使用 wAcc（加权精确度）替代。
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	缺省情况下，使用 allTrainingData 来估计模型精确度。使用 partitionTrainingData 来指定要使用训练数据的百分比，或 useOtherTable 来使用源自指定数据库表的训练数据集。
perc_training_data	数字	如果 prune_tree_options 设置为 partitionTrainingData，则指定用于训练的数据所占的百分比。
prune_seed	整数	在 prune_tree_options 设置为 partitionTrainingData 时，用于重复分析结果的随机种子，缺省值是 1。
pruning_table	字符串	这是用于估计模型精确度的单独修剪数据集的表名称。
compute_probabilities	*flag*	如果为 true，那么将生成置信度级别（概率）字段以及预测字段。

Netezza K-Means

类型为 netezzakmeansnode 的节点的可用属性如下所示：

表 *212. netezzakmeansnode* 属性

netezzakmeansnode 属性	**值(V)**	**属性说明**
distance_measure	Euclidean Manhattan 堪培拉距离 最大值	这是用于对数据点之间的距离进行测量的方法。
num_clusters	整数	要创建的聚类数；缺省值为 3。
max_iterations	整数	算法迭代次数，模型训练在此之后停止；缺省值为 5。
rand_seed	整数	这是用于复制分析结果的随机种子；缺省值为 12345。

Netezza Bayes 网络

类型为 netezzabayesnode 的节点的可用属性如下所示：

表 *213. netezzabayesnode* 属性

netezzabayesnode 属性	**值(V)**	**属性说明**
base_index	整数	对第一个输入字段指定的数字标识，用于进行内部管理；缺省值为 777。
sample_size	整数	属性数目非常大时的采样大小；缺省值为 10,000。
display_additional_information	*flag*	如果为 true，则在消息对话框中显示额外的进度信息。
type_of_prediction	best neighbors nn-neighbors	要使用的预测算法类型：best（最相关的相邻值）、neighbors（相邻值的加权预测）或 nn-neighbors（非空相邻值）。

Netezza 朴素贝叶斯

类型为 netezzanaivebayesnode 的节点的可用属性如下所示：

表 *214. netezzanaivebayesnode* 属性

netezzanaivebayesnode 属性	**值(V)**	**属性说明**
compute_probabilities	*flag*	如果为 true，那么将生成置信度级别（概率）字段以及预测字段。
use_m_estimation	*flag*	如果为 true，则使用 m-estimation 技术以避免估算期间的零概率。

Netezza KNN

类型为 netezzaknnnode 的节点的可用属性如下所示：

表 *215. netezzaknnnode* 属性

netezzaknnnode 属性	**值(V)**	**属性说明**
权重	结构化	这是用于对各个类指定权重的结构化属性。示例： set :netezzaknnnode.weights = [[drugA 0.3][drugB 0.6]]

表 215. *netezzaknnnode* 属性 (续)

netezzaknnnode 属性	**值(V)**	**属性说明**
distance_measure	Euclidean Manhattan 堪培拉距离 最大值	这是用于对数据点之间的距离进行测量的方法。
num_nearest_neighbors	整数	特定个案的最近相邻元素数；缺省值为 3。
standardize_measurements	*flag*	如果为 true，那么在计算距离值之前，对连续输入字段的测量值进行标准化。
use_coresets	*flag*	如果为 true，则对大型数据集使用核心集采样以提高计算速度。

Netezza 分裂式聚类

类型为 netezzadivclusternode 的节点的可用属性如下所示：

表 216. *netezzadivclusternode* 属性

netezzadivclusternode 属性	**值(V)**	**属性说明**
distance_measure	Euclidean Manhattan 堪培拉距离 最大值	这是用于对数据点之间的距离进行测量的方法。
max_iterations	整数	在模型训练停止前执行的最大算法迭代次数；缺省值为 5。
max_tree_depth	整数	可以将数据集拆分为的最大级别数；缺省值为 3。
rand_seed	整数	随机种子，用于复制分析；缺省值为 12345。
min_instances_split	整数	可以拆分的最小记录数，缺省值为 5。
level	整数	要将记录评分到的层次结构级别；缺省值为 -1。

Netezza PCA

类型为 netezzapcanode 的节点的可用属性如下所示：

表 217. *netezzapcanode* 属性

netezzapcanode 属性	**值(V)**	**属性说明**
center_data	*flag*	如果为 true（缺省值），那么先执行数据集中（也称为"平均值消去法"），然后再执行分析。
perform_data_scaling	*flag*	如果为 true，那么在分析前执行数据换算。这样做可以减低以不同单位测量不同变量时的分析任意性。
force_eigensolve	*flag*	如果为 true，则使用不太准确但较快的方法来查找主成份。
pc_number	整数	要将数据集精简到的主成份数；缺省值为 1。

Netezza 回归树

类型为 netezzaregtreenode 的节点的可用属性如下所示：

表 *218. netezzaregtreenode* 属性

netezzaregtreenode 属性	**值(V)**	**属性说明**
max_tree_depth	整数	树在根节点下可以增长到的最大级别数；缺省值为 10。
split_evaluation_measure	Variance	类杂质测量，用于评估分割树的最佳位置，缺省值（当前唯一选项）是 Variance。
min_improvement_splits	数字	在树中进行新拆分前要将杂质减少到的最小数量。
min_instances_split	整数	可以拆分的最小记录数。
pruning_measure	mse r2 pearson spearman	要使用的修剪方法
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	缺省情况下，使用 allTrainingData 来估计模型精确度。使用 partitionTrainingData 来指定要使用训练数据的百分比，或 useOtherTable 来使用源自指定数据库表的训练数据集。
perc_training_data	数字	如果 prune_tree_options 设置为 PercTrainingData，则指定用于训练的数据所占的百分比。
prune_seed	整数	在 prune_tree_options 设置为 PercTrainingData 时，用于重复分析结果的随机种子，缺省值是 1。
pruning_table	字符串	这是用于估计模型精确度的单独修剪数据集的表名称。
compute_probabilities	*flag*	如果为 true，则指定应该包括在输出中的指定类的方差。

Netezza 线性回归

类型为 netezzalineregressionnode 的节点的可用属性如下所示：

表 *219. netezzalineregressionnode* 属性

netezzalineregressionnode 属性	**值(V)**	**属性说明**
use_svd	*flag*	如果为 true，则使用"奇异值分解"矩阵代替原始矩阵，以便提高速度和数字准确性。
include_intercept	*flag*	如果为 true（缺省值），那么提高解的整体准确性。
calculate_model_diagnostics	*flag*	如果为 true，则对模型计算诊断信息。

Netezza 时间序列

类型为 netezzatimeseriesnode 的节点的可用属性如下所示：

表 220. *netezzatimeseriesnode* 属性

netezzatimeseriesnode 属性	**值(V)**	**属性说明**
time_points	*field*	此输入字段包含时间序列的日期值或时间值。
time_series_ids	*field*	此输入字段包含时间序列标识；在输入包含多个时间序列时使用。
model_table	*field*	这是用于存储 Netezza 时间序列模型的数据库表。
description_table	*field*	这是包含时间序列名称和描述的输入表的名称。
seasonal_adjustment_table	*field*	这是一个输出表的名称，该表用于存储指数平滑或季节性趋势分解算法所计算的按季度调整值。
algorithm_name	SpectralAnalysis 或 spectral ExponentialSmoothing 或 esmoothing ARIMA SeasonalTrendDecomposition 或 std	这是用于时间序列模型的算法。
trend_name	N A DA M DM	指数平滑的趋势类型： N - 无 A - 加性 DA - 衰减加性 M - 乘性 DM - 衰减乘性
seasonality_type	N A M	指数平滑的季节性类型： N - 无 A - 加性 M - 乘性
interpolation_method	linear cubicspline exponentialspline	要使用的插值方法。
timerange_setting	SD SP	用于设置要使用的时间范围： SD - 由系统确定（使用时间序列数据的完整范围） SP - 用户通过 earliest_time 和 latest_time 指定

表 220. *netezzatimeseriesnode* 属性 (续)

netezzatimeseriesnode 属性	**值(V)**	**属性说明**
earliest_time	整数 日期 时间 *timestamp*	开始值和结束值（如果 timerange_setting 为 SP）。 格式应遵循 time_points 值。 例如，如果 time_points 字段包含日期，那么此值也应该是日期。 示例： set NZ_DT1.timerange_setting = 'SP' set NZ_DT1.earliest_time = '1921-01-01' set NZ_DT1.latest_time = '2121-01-01'
latest_time		
arima_setting	SD SP	用于设置 ARIMA 算法（仅当 algorithm_name 设置为 ARIMA 时才使用）： SD - 由系统确定 SP - 由用户指定 如果 arima_setting = SP，请使用下列参数来设置季节性值和非季节性值。示例（仅非季节性）： set NZ_DT1.algorithm_name = 'arima' set NZ_DT1.arima_setting = 'SP' set NZ_DT1.p_symbol = 'lesseq' set NZ_DT1.p = '4' set NZ_DT1.d_symbol = 'lesseq' set NZ_DT1.d = '2' set NZ_DT1.q_symbol = 'lesseq' set NZ_DT1.q = '4'
p_symbol	less eq lesseq	ARIMA - 参数 p、d、q、sp、sd 和 sq 的运算符： less - 小于 eq - 等于 lesseq - 小于或等于
d_symbol		
q_symbol		
sp_symbol		
sd_symbol		
sq_symbol		
p	整数	ARIMA - 自动关联的非季节性程度。
q	整数	ARIMA - 非季节性派生值。
d	整数	ARIMA - 模型中的移动平均值移动平均值阶的非季节性数目。
sp	整数	ARIMA - 自动关联的季节性程度。
sq	整数	ARIMA - 季节性派生值。
sd	整数	ARIMA - 模型中的移动平均值移动平均值阶的季节性数目。

表 220. *netezzatimeseriesnode* 属性 (续)

netezzatimeseriesnode 属性	**值(V)**	**属性说明**
advanced_setting	SD SP	确定如何处理高级设置: SD - 由系统确定 SP - 由用户通过 period、units_period 和 forecast_setting 指定。 示例: set NZ_DT1.advanced_setting = 'SP' set NZ_DT1.period = 5 set NZ_DT1.units_period = 'd'
句号	整数	季节周期的长度，与 units_period 一起指定。不适用于谱分析。
units_period	ms s min h d wk q y	period 的表示单位: ms - 毫秒 s - 秒 min - 分钟 h - 小时 d - 日 wk - 星期 q - 季度 y - 年 例如，对于每周时间序列，请对 period 使用 1，并对 units_period 使用 wk。
forecast_setting	forecasthorizon forecasttimes	指定如何进行预测。
forecast_horizon	整数 日期 时间 *timestamp*	如果 forecast_setting = forecasthorizon，那么指定预测结束点值。 格式应遵循 time_points 值。 例如，如果 time_points 字段包含日期，那么此值也应该是日期。
forecast_times	整数 日期 时间 *timestamp*	如果 forecast_setting = forecasttimes，那么指定用于进行预测的值。 格式应遵循 time_points 值。 例如，如果 time_points 字段包含日期，那么此值也应该是日期。
include_history	*flag*	指示是否将历史值包括在输出中。
include_interpolated_values	*flag*	指示是否将内插值包括在输出中。如果 include_history 为 false，则不适用。

Netezza 广义线性

类型为 netezzaglmnode 的节点的可用属性如下所示：

表 221. *netezza GLM* 属性

netezzaglmnode 属性	**值(V)**	**属性说明**
dist_family	bernoulli gaussian poisson negativebinomial wald gamma	分布类型；缺省值为 bernoulli。
dist_params	数字	要使用的分布参数值。仅当 distribution 为 Negativebinomial 时才适用。
trials	整数	仅当 distribution 为 Binomial 时才适用。当目标响应为发生在一组试验中的事件数时，target 字段包含事件数，trials 字段包含试验数。
model_table	*field*	这是用于存储 Netezza 广义线性模型的数据库表。
maxit	整数	算法应执行的最大迭代次数；缺省值为 20。
eps	数字	指定最大误差值（以科学记数法表示），达到此值后，算法应停止查找最佳匹配模型。缺省值为 -3，这表示 1E-3，即 0.001。
tol	数字	设置数值（用科学表示法），低于此值的所有误差均被视为 0 值。缺省值为 -7，表示误差值若低于 1E-7 （或 0.0000001），则被视为不显著。
link_func	identity inverse invnegative invsquare sqrt power oddspower log clog loglog cloglog logit probit gaussit cauchit canbinom cangeom cannegbinom	要使用的联接函数；缺省值为 logit。

表 221. *netezza GLM* 属性 *(续)*

netezzaglmnode 属性	**值(V)**	**属性说明**
link_params	数字	要使用的关联函数参数值。仅当 link_function 为 power 或 oddspower 时才适用。
interaction	[[[*colnames1*],[*levels1*]], [[*colnames2*],[*levels2*]], ...,[[*colnamesN*],[*levelsN*]],]	指定字段之间的交互。*colnames* 是输入字段的列表，而 *level* 对于每个字段始终为 0。 示例： [[["K","BP","Sex","K"],[0,0,0,0]], [["Age","Na"],[0,0]]]
intercept	*flag*	如果为 true，则在模型中包括截距。

Netezza 模型块属性

Netezza 数据库模型块的公共属性如下所示。

表 222. 公共 *Netezza* 模型块属性

公共 Netezza 模型块属性	**值**	**属性描述**
connection	*string*	这是用于存储模型的 Netezza 数据库的连接字符串。
table_name	*string*	这是用于存储模型的数据库表的名称。

其他模型块的属性与相应建模节点的属性相同。

模型块的脚本名称如下所示。

表 223. *Netezza* 模型块的脚本名称

模型块	**脚本名称**
决策树	applynetezzadectreenode
K-Means	applynetezzakmeansnode
贝叶斯网络	applynetezzabayesnode
朴素贝叶斯	applynetezzanaivebayesnode
KNN	applynetezzaknnnode
分裂式聚类	applynetezzadivclusternode
PCA	applynetezzapcanode
回归树	applynetezzaregtreenode
线性回归	applynetezzalineregressionnode
时间序列	applynetezzatimeseriesnode
广义线性	applynetezzaglmnode

第 16 章 输出节点属性

输出节点的属性与其他"类型"节点的属性略有不同。输出节点属性不是指特定的节点选项，而是存储对输出对象的引用。这在从表中提取值并将其设置为流参数时非常有用。

本节说明输出节点的可用的脚本编制属性。

analysisnode 属性

"分析"节点评估预测模型生成准确预测的能力。"分析"节点执行一个或多个模型块的预测值和实际值之间的各种比较。"分析"节点也可以对比各 个预测模型。

示例

```
node = stream.create("analysis", "My node")
# "Analysis" tabnode.setPropertyValue("coincidence", True)
node.setPropertyValue("performance", True)
node.setPropertyValue("confidence", True)
node.setPropertyValue("threshold", 75)
node.setPropertyValue("improve_accuracy", 3)
node.setPropertyValue("inc_user_measure", True)
# "Define User Measure..."node.setPropertyValue("user_if", "@TARGET = @PREDICTED")
node.setPropertyValue("user_then", "101")
node.setPropertyValue("user_else", "1")
node.setPropertyValue("user_compute", ["Mean", "Sum"])
node.setPropertyValue("by_fields", ["Drug"])
# "Output" tabnode.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/analysis_out.html")
```

表 *224. analysisnode* 属性.

analysisnode 属性	**数据类型**	**属性描述**
output_mode	Screen File	用于指定输出节点中生成的输出的目标位置。
use_output_name	标志	指定是否使用定制的输出名。
output_name	字符串	如果 use_output_name 为 true ，那么指定使用的名称。
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	用于指定输出类型。
by_fields	列表	
full_filename	*string*	如果是磁盘、数据或 HTML 输出，那么此属性指输出文件的名称。
coincidence	标志	
performance	标志	
evaluation_binary	*flag*	

表 224. *analysisnode* 属性 *(续)*.

analysisnode 属性	**数据类型**	**属性描述**
confidence	标志	
threshold	*number*	
improve_accuracy	*number*	
inc_user_measure	标志	
user_if	表达式	
user_then	表达式	
user_else	表达式	
user_compute	[Mean Sum Min Max SDev]	

dataauditnode 属性

"数据审核"节点提供有关数据的全面概览，包括每个字段的汇总统计、直方图和分布以及有关离群值、缺失值和极值的信息。结果显示在易于读取的矩阵 中，该矩阵可以排序并且可以用于生成完整大小的图表和数据准备节点。

示例

```
filenode = stream.createAt("variablefile", "File", 100, 100)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node = stream.createAt("dataaudit", "My node", 196, 100)
stream.link(filenode, node)
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("fields", ["Age", "Na", "K"])
node.setPropertyValue("display_graphs", True)
node.setPropertyValue("basic_stats", True)
node.setPropertyValue("advanced_stats", True)
node.setPropertyValue("median_stats", False)
node.setPropertyValue("calculate", ["Count", "Breakdown"])
node.setPropertyValue("outlier_detection_method", "std")
node.setPropertyValue("outlier_detection_std_outlier", 1.0)
node.setPropertyValue("outlier_detection_std_extreme", 3.0)
node.setPropertyValue("output_mode", "Screen")
```

表 225. *dataauditnode* 属性.

dataauditnode 属性	**数据类型**	**属性描述**
custom_fields	标志	
fields	*[field1 ... fieldN]*	
overlay	字段	
display_graphs	标志	用于打开或关闭输出矩阵中图形的显示。
basic_stats	标志	
advanced_stats	标志	
median_stats	标志	
calculate	Count Breakdown	用于计算缺失值。选择两种计算方法中的一种、两种，或均不选择。

表 225. *dataauditnode* 属性 *(续)*.

dataauditnode 属性	**数据类型**	**属性描述**
outlier_detection_method	std iqr	用于指定离群值和极值的检测方法。
outlier_detection_std_outlier	*number*	如果 outlier_detection_method 是 std，那么指定用于定义离群值的数值。
outlier_detection_std_extreme	*number*	如果 outlier_detection_method 是 std，那么指定用于定义极值的数值。
outlier_detection_iqr_outlier	*number*	如果 outlier_detection_method 是 iqr，那么指定用于定义离群值的数值。
outlier_detection_iqr_extreme	*number*	如果 outlier_detection_method 是 iqr，那么指定用于定义极值的数值。
use_output_name	标志	指定是否使用定制的输出名。
output_name	字符串	如果 use_output_name 为 true ，那么指定使用的名称。
output_mode	Screen File	用于指定输出节点中生成的输出的目标位置。
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	用于指定输出类型。
paginate_output	*flag*	当 output_format 是 HTML 时，使输出分页。
lines_per_page	*number*	与 paginate_output 一起使用时，指定每个输出页中的行数。
full_filename	*string*	

extensionoutputnode 属性

通过使用您自己的定制 R 或 Python for Spark 脚本，可以使用"扩展输出"节点来分析模型评分的数据和结果。分析的输出可以是文本，也可以是图形。输出将添加到管理器窗格的**输出**选项卡中；另外，可以将输出重定向到文件。

Python for Spark 示例

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_output", "extension_output")
node.setPropertyValue("syntax_type", "Python")
```

```
python_script = """
import json
import spss.pyspark.runtime

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
schema = df.dtypes[:]
print df
"""

node.setPropertyValue("python_syntax", python_script)
```

R 示例

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", "print(modelerData$Age)")
```

表 *226. extensionoutputnode* 属性

extensionoutputnode 属性	**数据类型**	**属性描述**
syntax_type	*R* *Python*	指定要运行的脚本 - R 或 Python（R 为缺省值）
r_syntax	*string*	这是用于进行模型评分的 R 脚本语法。
python_syntax	*string*	这是用于进行模型评分的 Python 脚本语法。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_missing	*flag*	此选项用于将缺失值转换为 R NA 值。
convert_datetime	*flag*	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。
output_to	Screen File	指定输出类型（Screen 或 File）。
output_type	Graph Text	指定生成图形输出还是文本输出。
full_filename	*string*	用于生成输出的文件名。
graph_file_type	HTML COU	输出文件的文件类型（.html 或 .cou）。
text_file_type	HTML TEXT COU	指定文本输出的文件类型（.html、.txt 或 .cou）。

matrixnode 属性

"矩阵"节点创建一个表，用于显示字段之间的关系。此节点最常用于显示两个符号字段之间的关系，但也可用于显示标志字段或数字字段之间的关系。

示例

```
node = stream.create("matrix", "My node")
# "Settings" tabnode.setPropertyValue("fields", "Numerics")
node.setPropertyValue("row", "K")
node.setPropertyValue("column", "Na")
node.setPropertyValue("cell_contents", "Function")
node.setPropertyValue("function_field", "Age")
node.setPropertyValue("function", "Sum")
# "Appearance" tabnode.setPropertyValue("sort_mode", "Ascending")
node.setPropertyValue("highlight_top", 1)
node.setPropertyValue("highlight_bottom", 5)
node.setPropertyValue("display", ["Counts", "Expected", "Residuals"])
node.setPropertyValue("include_totals", True)
# "Output" tabnode.setPropertyValue("full_filename", "C:/output/matrix_output.html")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("paginate_output", True)
node.setPropertyValue("lines_per_page", 50)
```

表 *227. matrixnode* 属性.

matrixnode 属性	**数据类型**	**属性描述**
fields	Selected Flags Numerics	
row	字段	
column	字段	
include_missing_values	标志	指定在行和列输出中是否包含用户缺失值（空白）和系统缺失值（空值）。
cell_contents	CrossTabs Function	
function_field	字符串	
function	Sum Mean Min Max SDev	
sort_mode	Unsorted Ascending 降序 (Descending)	
highlight_top	*number*	如果非零，那么为 true。
highlight_bottom	*number*	如果非零，那么为 true。
display	[Counts 期望值(E) 残差 (Residuals) RowPct ColumnPct TotalPct]	
include_totals	标志	
use_output_name	标志	指定是否使用定制的输出名。

表 227. *matrixnode* 属性 *(续)*.

matrixnode 属性	**数据类型**	**属性描述**
output_name	字符串	如果 use_output_name 为 true ，那么指定使用的名称。
output_mode	Screen File	用于指定输出节点中生成的输出的目标位置。
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	用于指定输出类型。Formatted 和 Delimited 格式都可使用修饰符 transposed，此符号可转置表中的行和列。
paginate_output	*flag*	当 output_format 是 HTML 时，使输出分页。
lines_per_page	*number*	与 paginate_output 一起使用时，指定每个输出页中的行数。
full_filename	*string*	

meansnode 属性

"均值"节点在独立组之间或者相关字段的配对之间进行均值比较，以检验是否存在显著差别。例如，您可以比较开展促销 前后的平均收入，或者将来自未接受促销客户的收入与接受促销客户的收入进行比较。

示例

```
node = stream.create("means", "My node")
node.setPropertyValue("means_mode", "BetweenFields")
node.setPropertyValue("paired_fields", [["OPEN_BAL", "CURR_BAL"]])
node.setPropertyValue("label_correlations", True)
node.setPropertyValue("output_view", "Advanced")
node.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/means_output.html")
```

表 228. *meansnode* 属性.

meansnode 属性	**数据类型**	**属性描述**
means_mode	BetweenGroups BetweenFields	指定要在数据上执行的均值统计的类型。
test_fields	[field1 ... fieldn]	指定当 means_mode 设置为 BetweenGroups时的检验字段。
grouping_field	字段	指定分组字段。
paired_fields	[[field1 field2] [field3 field4] ...]	指定当 means_mode 设置为 BetweenFields时要使用的字段对。
label_correlations	标志	指定在输出中是否显示相关标签。仅在当 means_mode 设置为 BetweenFields 时才应用此设置。

表 228. *meansnode* 属性 (续).

meansnode 属性	数据类型	属性描述
correlation_mode	Probability Absolute	指定用概率还是绝对值标注相关。
weak_label	字符串	
medium_label	字符串	
strong_label	字符串	
weak_below_probability	*number*	当 correlation_mode 设置为 Probability 时，指定弱相关的分界值。这必须是 0 到 1 之间的一个值，例如 0.90。
strong_above_probability	*number*	强相关的分界值。
weak_below_absolute	*number*	当 correlation_mode 设置为 Absolue 时，指定弱相关的分界值。这必须是 0 到 1 之间的一个值，例如 0.90。
strong_above_absolute	*number*	强相关的分界值。
unimportant_label	字符串	
marginal_label	字符串	
important_label	字符串	
unimportant_below	*number*	低字段重要性的分界值。这必须是 0 到 1 之间的一个值，例如 0.90。
important_above	*number*	
use_output_name	标志	指定是否使用定制的输出名。
output_name	字符串	使用的名称。
output_mode	Screen File	指定从输出节点中生成的输出的目标位置。
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	指定输出类型。
full_filename	*string*	
output_view	Simple Advanced	指定在输出中显示简单视图还是高级视图。

reportnode 属性

"报告"节点可创建格式化报告，其中包含固定文本、数据及得自数据的其他表达式。您可以使用文本模板指定报告的格式，以定义固定文本和数据输出构造。通过在模板中使用 HTML 标记以及在"输出"选项卡上设置选项，可以提供定制文本格式。通过使用模板中的 CLEM 表达式，可以包括数据值和其他条件输出。

示例

```
node = stream.create("report", "My node")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/report_output.html")
node.setPropertyValue("lines_per_page", 50)
node.setPropertyValue("title", "Report node created by a script")
node.setPropertyValue("highlights", False)
```

表 *229. reportnode* 属性.

reportnode 属性	**数据类型**	**属性描述**
output_mode	Screen File	用于指定输出节点中生成的输出的目标位置。
output_format	HTML (*.html*) Text (*.txt*) Output (*.cou*)	用于指定文件输出的类型。
format	Auto Custom	用于选择是自动格式化输出还是使用模板中包含的 HTML 进行格式化。要在模板中使用 HTML 格式，请指定 Custom。
use_output_name	标志	指定是否使用定制的输出名。
output_name	字符串	如果 use_output_name 为 true ，那么指定使用的名称。
text	字符串	
full_filename	字符串	
highlights	标志	
title	字符串	
lines_per_page	*number*	

routputnode 属性

通过使用您自己的定制 R 脚本，可以使用"R 输出"节点来分析模型评分的数据和结果。分析的输出可以是文本，也可以是图形。输出将添加到管理器窗格的**输出**选项卡中；另外，可以将输出重定向到文件。

表 *230. routputnode* 属性

routputnode 属性	**数据类型**	**属性描述**
syntax	*string*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*flag*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*flag*	
output_name	Auto Custom	
custom_name	*string*	

表 230. *routputnode* 属性 (续)

routputnode 属性	数据类型	属性描述
output_to	Screen File	
output_type	Graph Text	
full_filename	*string*	
graph_file_type	HTML COU	
text_file_type	HTML TEXT COU	

setglobalsnode 属性

"设置全局值"节点扫描数据并计算可在 CLEM 表达式中使用的汇总值。例如，可以用该节点为一个名为年龄的字段计算统计并通过插入函数 @GLOBAL_MEAN(age) 在 CLEM 表达式中使用年龄的总均值。

示例

```
node = stream.create("setglobals", "My node")
node.setKeyedPropertyValue("globals", "Na", ["Max", "Sum", "Mean"])
node.setKeyedPropertyValue("globals", "K", ["Max", "Sum", "Mean"])
node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
node.setPropertyValue("clear_first", False)
node.setPropertyValue("show_preview", True)
```

表 231. *setglobalsnode* 属性.

setglobalsnode 属性	数据类型	属性描述
globals	[Sum Mean Min Max SDev]	结构属性，在其中，必须使用下面的语法引用要设置的字段： node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
clear_first	标志	
show_preview	标志	

simevalnode 属性

"模拟评估"节点对指定的预测目标字段进行评估，并显示有关该目标字段的分布和相关性信息。

表 232. *simevalnode* 属性.

simevalnode 属性	数据类型	属性描述
target	字段	
iteration	字段	
presorted_by_iteration	布尔值	
max_iterations	*number*	
tornado_fields	*[field1...fieldN]*	
plot_pdf	布尔值	
plot_cdf	布尔值	
show_ref_mean	布尔值	
show_ref_median	布尔值	
show_ref_sigma	布尔值	
num_ref_sigma	*number*	
show_ref_pct	布尔值	
ref_pct_bottom	*number*	
ref_pct_top	*number*	
show_ref_custom	布尔值	
ref_custom_values	*[number1...numberN]*	
category_values	Category Probabilities Both	
category_groups	Categories Iterations	
create_pct_table	布尔值	
pct_table	Quartiles Intervals Custom	
pct_intervals_num	*number*	
pct_custom_values	*[number1...numberN]*	

simfitnode 属性

"模拟拟合"节点检查每个字段中的数据的统计分布，并生成（或更新）"模拟生成"节点，同时将最佳拟合分布分配给每个字段。然后，可以使用"模拟生成"节点来生成模拟数据。

表 233. *simfitnode* 属性.

simfitnode 属性	数据类型	属性描述
build	Node XMLExport Both	
use_source_node_name	布尔值	

表 233. *simfitnode* 属性 *(续)*.

simfitnode 属性	**数据类型**	**属性描述**
source_node_name	*string*	正在生成或更新的源节点的定制名称。
use_cases	All LimitFirstN	
use_case_limit	*integer*	
fit_criterion	AndersonDarling KolmogorovSmirnov	
num_bins	*integer*	
parameter_xml_filename	*string*	
generate_parameter_import	布尔值	

statisticsnode 属性

"统计"节点提供有关数字字段的基本汇总信息。它计算各个字段的汇总统计以及字段间的相关性。

示例

```
node = stream.create("statistics", "My node")
# "Settings" tabnode.setPropertyValue("examine", ["Age", "BP", "Drug"])
node.setPropertyValue("statistics", ["mean", "sum", "sdev"])
node.setPropertyValue("correlate", ["BP", "Drug"])
# "Correlation Labels..." section
node.setPropertyValue("label_correlations", True)
node.setPropertyValue("weak_below_absolute", 0.25)
node.setPropertyValue("weak_label", "lower quartile")
node.setPropertyValue("strong_above_absolute", 0.75)
node.setPropertyValue("medium_label", "middle quartiles")
node.setPropertyValue("strong_label", "upper quartile")
# "Output" tabnode.setPropertyValue("full_filename", "c:/output/statistics_output.html")
node.setPropertyValue("output_format", "HTML")
```

表 234. *statisticsnode* 属性.

statisticsnode 属性	**数据类型**	**属性描述**
use_output_name	标志	指定是否使用定制的输出名。
output_name	字符串	如果 use_output_name 为 true ，那么指定使用的名称。
output_mode	Screen File	用于指定输出节点中生成的输出的目标位置。
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	用于指定输出类型。
full_filename	*string*	
examine	列表	

表 234. *statisticsnode* 属性 (续).

statisticsnode 属性	数据类型	属性描述
correlate	列表	
statistics	[count mean sum min max range variance sdev semean median mode]	
correlation_mode	Probability Absolute	指定用概率还是绝对值标注相关。
label_correlations	标志	
weak_label	字符串	
medium_label	字符串	
strong_label	字符串	
weak_below_probability	*number*	当 correlation_mode 设置为 Probability 时，指定弱相关的分界值。这必须是 0 到 1 之间的一个值，例如 0.90。
strong_above_probability	*number*	强相关的分界值。
weak_below_absolute	*number*	当 correlation_mode 设置为 Absolue 时，指定弱相关的分界值。这必须是 0 到 1 之间的一个值，例如 0.90。
strong_above_absolute	*number*	强相关的分界值。

statisticsoutputnode 属性

"Statistics 输出"节点用于调用 IBM SPSS Statistics 过程，以分析 IBM SPSS Modeler 数据。可以访问许多不同的 IBM SPSS Statistics 分析过程。此节点需要 IBM SPSS Statistics 的许可副本。

有关此节点属性的信息，请参阅第 316 页的『statisticsoutputnode 属性』。

tablenode 属性

"表"节点以表格式显示数据，这些数据还可以写入到文件中。每当您需要检查数据值或者采用可轻松阅读的格式导出这些数据值时，此节点非常有用。

示例

```
node = stream.create("table", "My node")
node.setPropertyValue("highlight_expr", "Age > 30")
node.setPropertyValue("output_format", "HTML")
```

```
node.setPropertyValue("transpose_data", True)
node.setPropertyValue("full_filename", "C:/output/table_output.htm")
node.setPropertyValue("paginate_output", True)
node.setPropertyValue("lines_per_page", 50)
```

表 *235. tablenode* 属性.

tablenode 属性	**数据类型**	**属性描述**
full_filename	*string*	如果是磁盘、数据或 HTML 输出，那么此属性指输出文件的名称。
use_output_name	标志	指定是否使用定制的输出名。
output_name	字符串	如果 use_output_name 为 true ，那么指定使用的名称。
output_mode	Screen File	用于指定输出节点中生成的输出的目标位置。
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	用于指定输出类型。
transpose_data	标志	导出前转置数据，使行表示字段，列表示记录。
paginate_output	*flag*	当 output_format 是 HTML 时，使输出分页。
lines_per_page	*number*	与 paginate_output 一起使用时，指定每个输出页中的行数。
highlight_expr	字符串	
output	字符串	只读属性，可保留对由节点构建的最后一个表的引用。
value_labels	*[[Value LabelString]* *[Value LabelString] ...]*	用于为值对指定标签。
display_places	*integer*	为字段设置显示的小数位数（仅用于以 REAL 存储的字段）。值为 -1 时将使用流缺省值。
export_places	*integer*	为字段设置导出的小数位数（仅用于以 REAL 存储的字段）。值为 -1 时将使用流缺省值。
decimal_separator	DEFAULT PERIOD COMMA	为字段设置十进制分隔符（仅用于以 REAL 存储的字段）。

表 *235. tablenode* 属性 *(续)*.

tablenode 属性	**数据类型**	**属性描述**
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	为字段设置日期格式（仅用于以 DATE 或 TIMESTAMP 存储的字段）。
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	为字段设置时间格式（仅用于以 TIME 或 TIMESTAMP 存储的字段）。
column_width	*integer*	为字段设置列宽度。值为 -1 表示将列宽度设置为 Auto。
justify	AUTO CENTER LEFT RIGHT	为字段设置列对齐格式。

transformnode 属性

"转换"节点允许您先选择并以可视方式预览转换结果，然后再将这些转换应用于选择的字段。

示例

```
node = stream.create("transform", "My node")
node.setPropertyValue("fields", ["AGE", "INCOME"])
node.setPropertyValue("formula", "Select")
node.setPropertyValue("formula_log_n", True)
node.setPropertyValue("formula_log_n_offset", 1)
```

表 *236. transformnode* 属性.

transformnode 属性	**数据类型**	**属性描述**
fields	[*field1... fieldn*]	要在转换中使用的字段。
formula	All Select	表示应计算所有转换还是选定的转换。
formula_inverse	标志	表示是否应使用逆转换。
formula_inverse_offset	数字	表示公式中要使用的数据偏移量。除非用户指定，否则缺省情况下设置为 0。
formula_log_n	标志	表示是否应使用 $\log_n$ 转换。
formula_log_n_offset	数字	
formula_log_10	标志	表示是否应使用 $\log_{10}$ 转换。
formula_log_10_offset	数字	
formula_exponential	标志	表示是否应使用指数（e^x）转换。
formula_square_root	标志	表示是否应使用平方根转换。
use_output_name	标志	指定是否使用定制的输出名。
output_name	字符串	如果 use_output_name 为 true，那么指定使用的名称。
output_mode	Screen File	用于指定输出节点中生成的输出的目标位置。
output_format	HTML (*.html*) Output (*.cou*)	用于指定输出类型。
paginate_output	*flag*	当 output_format 是 HTML 时，使输出分页。
lines_per_page	*number*	与 paginate_output 一起使用时，指定每个输出页中的行数。
full_filename	*string*	表示要在文件输出中使用的文件名。

第 17 章 导出节点属性

公共导出节点属性

以下属性通用于所有导出节点。

表 *237.* 公共导出节点属性

属性	值	属性描述
publish_path	*string*	输入要用于所发布的图像和参数文件的 rootname 名称。
publish_metadata	*flag*	指定是否生成元数据文件（用于描述图像的输入和输出以及它们的数据模型）。
publish_use_parameters	*flag*	指定是否在 *.par 文件中包含流参数。
publish_parameters	*string list*	指定要包括的参数。
execute_mode	export_data 发布	指定是执行节点而不发布流，还是在执行节点时自动发布流。

asexport 属性

您可以使用 Analytic Server 导出在 Hadoop 分布式文件系统 (HDFS) 上运行流。

示例

```
node.setPropertyValue("use_default_as", False)
node.setPropertyValue("connection",
["false","9.119.141.141","9080","analyticserver","ibm","admin","admin","false","","","","",""])
```

表 *238. asexport* 属性.

asexport 属性	数据类型	属性说明
data_source	*string*	数据源名称。
export_mode	字符串	指定是要将导出的数据**附加**到现有数据源，还是要**覆写**现有数据源。
use_default_as	*boolean*	如果设置为 True，请使用服务器 options.cfg 文件中配置的缺省 Analytic Server 连接。如果设置为 False，请使用此节点的连接。

表 238. *asexport* 属性 *(续)*.

asexport 属性	数据类型	属性说明
connection	["string","string","string", "string","string","string","string", "string","string","string", "string" ,"string"]	这是包含 Analytic Server 连接详细信息的列表属性。格式为：["is_secure_connect", "server_url", "server_port", "context_root", "consumer", "user_name", "password", "use-kerberos-auth", "kerberos-krb5-config-file-path", "kerberos-jaas-config-file-path", "kerberos-krb5-service-principal-name", "enable-kerberos-debug"]，其中 is_secure_connect: 指示是否使用安全连接，其值为 true 或 false。use-kerberos-auth: 指示是否使用 Kerberos 认证，其值为 true 或 false。enable-kerberos-debug: 指示是否使用 Kerberos 认证的调试模式，其值为 true 或 false。

cognosexportnode 属性

IBM Cognos 导出节点以 Cognos 数据库可以读取的格式导出数据。

对于此节点，必须定义 Cognos 连接和 ODBC 连接。

Cognos 连接

Cognos 连接的属性如下。

表 239. *cognosexportnode* 属性

cognosexportnode 属性	**数据类型**	**属性描述**
cognos_connection	*["string","flag","string","string","string"]*	这是包含 Cognos 服务器连接详细信息的列表属性。格式为：["Cognos_server_URL", login_mode, "namespace", "username", "password"] 其中： Cognos_server_URL 是包含源的 Cognos 服务器的 URL。 login_mode 指示是否使用匿名登录，其值为 true 或 false；如果设置为 true，那么应将下列字段设置为 ""。 namespace 指定用于登录服务器的安全认证提供程序。 username 和 password 为用于登录 Cognos 服务器的用户名和密码。 作为替代 login_mode 的选项，还可以使用以下方式： • anonymousMode。例如：['Cognos_server_url', 'anonymousMode', "namespace", "username", "password"] • credentialMode。例如：['Cognos_server_url', 'credentialMode', "namespace", "username", "password"] • storedCredentialMode。例如：['Cognos_server_url', 'storedCredentialMode', "stored_credential_name"] 其中 stored_credential_name 是存储库中 Cognos 凭证的名称。
cognos_package_name	*string*	您要将数据导出到的 Cognos 数据包的路径和名称，例如： /Public Folders/MyPackage
cognos_datasource	*string*	
cognos_export_mode	发布 ExportFile	
cognos_filename	*string*	

ODBC 连接

ODBC 连接的属性和为下一节中的 databaseexportnode 列出的相同，例外是 datasource 属性无效。

databaseexportnode 属性

"数据库导出"节点将数据写入符合 ODBC 标准的关系数据源。要写入 ODBC 数据源，该数据源必须存在，且您对该数据源必须具有写许可权。

示例

```
'''
Assumes a datasource named "MyDatasource" has been configured'''
stream = modeler.script.stream()
db_exportnode = stream.createAt("databaseexport", "DB Export", 200, 200)
applynn = stream.findByType("applyneuralnetwork", None)
stream.link(applynn, db_exportnode)

# Export tabdb_exportnode.setPropertyValue("username", "user")
db_exportnode.setPropertyValue("datasource", "MyDatasource")
db_exportnode.setPropertyValue("password", "password")
db_exportnode.setPropertyValue("table_name", "predictions")
db_exportnode.setPropertyValue("write_mode", "Create")
db_exportnode.setPropertyValue("generate_import", True)
db_exportnode.setPropertyValue("drop_existing_table", True)
db_exportnode.setPropertyValue("delete_existing_rows", True)
db_exportnode.setPropertyValue("default_string_size", 32)

# Schema dialogdb_exportnode.setKeyedPropertyValue("type", "region", "VARCHAR(10)")
db_exportnode.setKeyedPropertyValue("export_db_primarykey", "id", True)
db_exportnode.setPropertyValue("use_custom_create_table_command", True)
db_exportnode.setPropertyValue("custom_create_table_command", "My SQL Code")

# Indexes dialogdb_exportnode.setPropertyValue("use_custom_create_index_command", True)
db_exportnode.setPropertyValue("custom_create_index_command", "CREATE BITMAP INDEX <index-name>
 ON <table-name> <(index-columns)>")
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields", ["id", "region"]])
```

表 *240. databaseexportnode* 属性.

databaseexportnode 属性	**数据类型**	**属性描述**
datasource	字符串	
username	字符串	
password	字符串	
epassword	字符串	在执行期间，此槽为只读。要生成经过编码的密码，请使用"工具"菜单中的"密码工具"。有关更多信息，请参阅第 45 页的『生成加密密码』主题。
table_name	字符串	
write_mode	Create Append Merge	

表 240. *databaseexportnode* 属性 (续).

databaseexportnode 属性	**数据类型**	**属性描述**
map	字符串	用于将流字段名称映射到数据库列名称（仅在 write_mode 为 Merge 的情况下才有效）。 对于合并，所有字段必须经过映射才能导出。数据库中不存在的字段名称将添加为新列。
key_fields	列表	指定用作键的流字段；map 属性指示了此字段在数据库中的对应项。
join	数据库 Add	
drop_existing_table	标志	
delete_existing_rows	标志	
default_string_size	*integer*	
type		用于设置模式类型的结构属性。
generate_import	标志	
use_custom_create_table_command	标志	使用 *custom_create_table* 通道修改标准 CREATE TABLE SQL 命令。
custom_create_table_command	字符串	指定字符串命令代替标准 CREATE TABLE SQL 命令使用。
use_batch	标志	下列属性是用于数据库批量装入的高级选项。Use_batch 为 true 时将关闭向数据库逐行提交的功能。
batch_size	*number*	指定在提交到内存前发送到数据库的记录数。
bulk_loading	Off ODBC External	指定批量装入类型。下面列出了 ODBC 和 External 的其他选项。
not_logged	*flag*	
odbc_binding	Row Column	指定通过 ODBC 批量装入时使用逐行绑定或逐列绑定。
loader_delimit_mode	Tab Space Other	对于通过外部程序的批量装入，指定定界符的类型。选择 Other 连同 loader_other_delimiter 属性以指定定界符，例如逗号 (,)。
loader_other_delimiter	*string*	
specify_data_file	*flag*	标志为 True 时可激活下面的 data_file 属性，在该属性中可以指定批量装入到数据库时写入的文件名和路径。
data_file	*string*	
specify_loader_program	*flag*	标志为 True 时可激活下面的 loader_program 属性，在该属性中可以指定外部装入程序脚本或程序的名称和位置。

表 240. *databaseexportnode* 属性 (续).

databaseexportnode 属性	数据类型	属性描述
loader_program	*string*	
gen_logfile	*flag*	标志为 True 时可激活下面的 logfile_name，在该属性中可以指定服务器上的文件的名称以生成错误日志。
logfile_name	*string*	
check_table_size	*flag*	标志为 True 时允许进行表检查以确保数据库表大小的增加与从 IBM SPSS Modeler 导出的行数相符。
loader_options	*string*	指定装入程序的其他参数，例如 -comment 和 -specialdir。
export_db_primarykey	标志	指定给定字段是否为主键。
use_custom_create_index_command	标志	如果标志为 true，那么为所有索引启用定制 SQL。
custom_create_index_command	字符串	指定启用定制 SQL 后用于创建索引的 SQL 命令。（创建特定索引时，该值可被覆盖，如下所示。）
indexes.INDEXNAME.fields		必要时创建指定的索引，并列出要包括在该索引中的字段名。
INDEXNAME "use_custom_create_index_command"	标志	用于启用或禁用特定索引的定制 SQL。请参阅下表之后的示例。
INDEXNAME "custom_create_index_ command"	*string*	指定用于指定索引的定制 SQL。请参阅下表之后的示例。
indexes.INDEXNAME.remove	标志	如果为 True，那么将从索引集中除去指定的索引。
table_space	字符串	指定将创建表空间。
use_partition	标志	指定将使用分布散列字段。
partition_field	字符串	指定分布散列字段的内容。

注：对于某些数据库，您可以指定以导出时进行压缩的方式来创建数据库表（例如，SQL 中的等效语句 CREATE TABLE MYTABLE (...) COMPRESS YES;）。为了支持此功能，提供了属性 use_compression 和 compression_mode，如下所示。

表 241. 使用了压缩功能的 *databaseexportnode* 属性.

databaseexportnode 属性	数据类型	属性描述
use_compression	布尔值	如果设置为 True，那么将以导出时进行压缩的方式创建表。

表 241. 使用了压缩功能的 *databaseexportnode* 属性 *(续)*.

databaseexportnode 属性	数据类型	属性描述
compression_mode	Row Page	设置 SQL Server 数据库的压缩级别。
	Default Direct_Load_Operations All_Operations Basic OLTP Query_High Query_Low Archive_High Archive_Low	设置 Oracle 数据库的压缩级别。请注意，值 OLTP、Query_High、Query_LowArchive_High 和 Archive_Low 至少需要 Oracle 11gR2。

显示如何针对特定索引更改 CREATE INDEX 命令的示例：

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["use_custom_create_index_command",
 True])db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["custom_create_index_command",
 "CREATE BITMAP INDEX <index-name> ON <table-name> <(index-columns)>"])
```

或者，也可以通过散列表完成此操作：

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields":["id", "region"],
 "use_custom_create_index_command":True, "custom_create_index_command":"CREATE INDEX <index-name> ON
 <table-name> <(index-columns)>"])
```

datacollectionexportnode 属性

Data Collection 导出节点以 Data Collection 市场调查软件使用的格式输出数据。必须安装 Data Collection 数据库才可使用此节点。

示例

```
stream = modeler.script.stream()
datacollectionexportnode = stream.createAt("datacollectionexport", "Data Collection", 200, 200)
datacollectionexportnode.setPropertyValue("metadata_file", "c:\\museums.mdd")
datacollectionexportnode.setPropertyValue("merge_metadata", "Overwrite")
datacollectionexportnode.setPropertyValue("casedata_file", "c:\\museumdata.sav")
datacollectionexportnode.setPropertyValue("generate_import", True)
datacollectionexportnode.setPropertyValue("enable_system_variables", True)
```

表 242. *datacollectionexportnode* 属性

datacollectionexportnode 属性	数据类型	属性描述
metadata_file	*string*	要导出的元数据文件的名称。
merge_metadata	Overwrite MergeCurrent	
enable_system_variables	*flag*	指定导出的 *.mdd* 文件是否应包括 Data Collection 系统变量。
casedata_file	*string*	要导出观测数据的 *.sav* 文件的名称。

表 242. *datacollectionexportnode* 属性（续）

datacollectionexportnode 属性	**数据类型**	**属性描述**
generate_import	*flag*	

excelexportnode 属性

Excel 导出节点以 Microsoft Excel .xlsx 文件格式输出数据。您还可选择在执行完此节点后自动启动 Excel 并打开导出的文件。

示例

```
stream = modeler.script.stream()
excelexportnode = stream.createAt("excelexport", "Excel", 200, 200)
excelexportnode.setPropertyValue("full_filename", "C:/output/myexport.xlsx")
excelexportnode.setPropertyValue("excel_file_type", "Excel2007")
excelexportnode.setPropertyValue("inc_field_names", True)
excelexportnode.setPropertyValue("inc_labels_as_cell_notes", False)
excelexportnode.setPropertyValue("launch_application", True)
excelexportnode.setPropertyValue("generate_import", True)
```

表 243. *excelexportnode* 属性

excelexportnode 属性	**数据类型**	**属性描述**
full_filename	*string*	
excel_file_type	Excel2007	
export_mode	Create Append	
inc_field_names	*flag*	指定字段名是否可以包含在工作表的第一行中。
start_cell	*string*	指定导出的开始单元格。
worksheet_name	*string*	要写入的工作表的名称。
launch_application	*flag*	指定是否应该对生成的文件调用 Excel。请注意，必须在"帮助应用程序"对话框（"工具"菜单 >"帮助应用程序"）中指定用于启动 Excel 的路径。
generate_import	*flag*	指定是否应生成用于读取所导出数据文件的"Excel 导入"节点。

extensionexportnode 属性

通过扩展导出节点，您可以运行 R 或 Python for Spark 脚本来导出数据。

Python for Spark 示例

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_export", "extension_export")
node.setPropertyValue("syntax_type", "Python")

python_script = """import spss.pyspark.runtime
from pyspark.sql import SQLContext
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
print df.dtypes[:]
_newDF = df.select("Age","Drug")
print _newDF.dtypes[:]

df.select("Age", "Drug").write.save("c:/data/ageAndDrug.json", format="json")
"""

node.setPropertyValue("python_syntax", python_script)
```

R 示例

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", """write.csv(modelerData, "C:/export.csv")""")
```

表 *244. extensionexportnode* 属性

extensionexportnode 属性	**数据类型**	**属性描述**
syntax_type	*R* *Python*	指定要运行的脚本 - R 或 Python（R 为缺省值）
r_syntax	*string*	要运行的 R 脚本编制语法。
python_syntax	*string*	要运行的 Python 脚本编制语法。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_missing	*flag*	此选项用于将缺失值转换为 R NA 值。
convert_datetime	*flag*	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。

outputfilenode 属性

"平面文件导出"节点将数据输出到定界文本文件中。这对于导出可以由其他分析或电子表格软件读取的数据非常有用。

示例

```
stream = modeler.script.stream()
outputfile = stream.createAt("outputfile", "File Output", 200, 200)
outputfile.setPropertyValue("full_filename", "c:/output/flatfile_output.txt")
outputfile.setPropertyValue("write_mode", "Append")
outputfile.setPropertyValue("inc_field_names", False)
outputfile.setPropertyValue("use_newline_after_records", False)
outputfile.setPropertyValue("delimit_mode", "Tab")
outputfile.setPropertyValue("other_delimiter", ",")
outputfile.setPropertyValue("quote_mode", "Double")
outputfile.setPropertyValue("other_quote", "*")
outputfile.setPropertyValue("decimal_symbol", "Period")
outputfile.setPropertyValue("generate_import", True)
```

表 *245. outputfilenode* 属性

outputfilenode 属性	**数据类型**	**属性描述**
full_filename	*string*	输出文件的名称。
write_mode	Overwrite Append	
inc_field_names	*flag*	
use_newline_after_records	*flag*	
delimit_mode	Comma Tab Space Other	
other_delimiter	*char*	
quote_mode	None Single Double Other	
other_quote	*flag*	
generate_import	*flag*	
encoding	StreamDefault SystemDefault "UTF-8"	

sasexportnode 属性

"SAS 导出"节点以 SAS 格式输出数据，以便将该数据读入 SAS 或者与 SAS 兼容的软件包。共有三种可用的 SAS 文件格式：SAS for Windows/OS2、SAS for UNIX 和 SAS V7/V8。

示例

```
stream = modeler.script.stream()
sasexportnode = stream.createAt("sasexport", "SAS Export", 200, 200)
sasexportnode.setPropertyValue("full_filename", "c:/output/SAS_output.sas7bdat")
sasexportnode.setPropertyValue("format", "SAS8")
sasexportnode.setPropertyValue("export_names", "NamesAndLabels")
sasexportnode.setPropertyValue("generate_import", True)
```

表 *246. sasexportnode* 属性

sasexportnode 属性	**数据类型**	**属性描述**
format	Windows UNIX SAS7 SAS8	可变属性标签字段。
full_filename	*string*	
export_names	NamesAndLabels NamesAsLabels	用于将字段名从 IBM SPSS Modeler 的导出中映射到 IBM SPSS Statistics 或 SAS 的变量名中。
generate_import	*flag*	

statisticsexportnode 属性

Statistics 导出节点以 IBM SPSS Statistics *.sav* 或 *.zsav* 格式输出数据。IBM SPSS Statistics Base 和其他产品可以读取 *.sav* 或 *.zsav* 文件。这种格式也用于 IBM SPSS Modeler 中的某些缓存文件。

有关此节点属性的信息，请参阅第 317 页的『statisticsexportnode 属性』。

tm1odataexport 节点属性

IBM Cognos TM1 导出节点以 Cognos TM1 数据库可以读取的格式导出数据。

表 *247. tm1odataexport* 节点属性

tm1odataexport 节点属性	**数据类型**	**属性描述**
admin_host	*string*	REST API 的主机名的 URL。
server_name	*string*	从 admin_host 中选择的 TM1 服务器的名称。
credential_type	inputCredential 或 storedCredential	用于指示凭证类型。
input_credential	列表	当 credential_type 为 inputCredential 时；指定域、用户名和密码。
stored_credential_name	*string*	当 credential_type 为 storedCredential 时；指定 C&DS 服务器上的凭证名称。
selected_cube	字段	您要向其中导出数据的多维数据集的名称。例如：TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan")
spss_field_to_tm1_element_mapping	列表	要映射到的 tm1 元素必须是所选多维数据集视图的列维度的组成部分。格式为：[[[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...], [[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]] 有 2 个列表可用于描述映射信息。将叶元素映射到维度对应于以下示例 2： 示例 1：第一个列表：([[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...]) 用于 TM1 维度映射信息。 每个含 3 个值的列表均指示维度映射信息。第三个布尔值用于指示是否选择维度的元素。例如："[Field_1, Dimension_1, False]" 表示 Field_1 映射到 Dimension_1；"[Element_1, Dimension_2, True]" 表示针对 Dimension_2 选中 Element_1。 示例 2：第二个列表：([[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]) 用于 TM1 度量维度元素映射信息。 每个含 3 个值的列表均指示度量元素映射信息。第三个布尔值用于指示是否需要创建新元素。"[Field_2, ExistMeasureElement, False]" 表示 Field_2 映射到 ExistMeasureElement；"[Field_3, NewMeasureElement, True]" 表示在 selected_measure 中需要选中 NewMeasureElement 度量维度，并将 Field_3 映射到此度量维度。
selected_measure	*string*	指定度量维度。 示例：setPropertyValue("selected_measure", "Measures")

tm1export 节点属性（不推荐）

IBM Cognos TM1 导出节点以 Cognos TM1 数据库可以读取的格式导出数据。

注：此节点在 Modeler 18.0 中不推荐。替换节点脚本名称为 *tm1odataexport*。

表 *248. tm1export* 节点属性.

tm1export 节点属性	**数据类型**	**属性描述**
pm_host	*string*	**注：**仅限 V16.0 和 17.0 主机名。例如：TM1_export.setPropertyValue ("pm_host", 'http://9.191.86.82:9510/pmhub/pm')
tm1_connection	*["field","field", ... ,"field"]*	**注：**仅限 V16.0 和 17.0 包含 TM1 服务器的连接详细信息的列表属性。格式为：["TM1_Server_Name", "tm1_ username", "tm1_password"] 例如：TM1_export.setPropertyValue ("tm1_connection", ['Planning Sample', "admin" "apple"])
selected_cube	字段	您要向其中导出数据的多维数据集的名称。例如：TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan")

表 248. *tm1export* 节点属性 *(续)*.

tm1export 节点属性	数据类型	属性描述
spssfield_tm1element_mapping	列表	要映射到的 tm1 元素必须是所选多维数据集视图的列维度的组成部分。格式为：[[[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...], [[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]] 有 2 个列表可用于描述映射信息。将叶元素映射到维度对应于以下示例 2： 示例 1：第一个列表：([[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...]) 用于 TM1 维度映射信息。 每个含 3 个值的列表均指示维度映射信息。第三个布尔值用于指示是否选择维度的元素。例如："[Field_1, Dimension_1, False]" 表示 Field_1 映射到 Dimension_1；"[Element_1, Dimension_2, True]" 表示针对 Dimension_2 选中 Element_1。 示例 2：第二个列表：([[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]) 用于 TM1 度量维度元素映射信息。 每个含 3 个值的列表均指示度量元素映射信息。第三个布尔值用于指示是否需要创建新元素。"[Field_2, ExistMeasureElement, False]" 表示 Field_2 映射到 ExistMeasureElement；"[Field_3, NewMeasureElement, True]" 表示在 selected_measure 中需要选中 NewMeasureElement 度量维度，并将 Field_3 映射到此度量维度。
selected_measure	*string*	指定度量维度。 示例：setPropertyValue("selected_measure", "Measures")

xmlexportnode 属性

"XML 导出"节点将数据以 XML 格式输出到文件。您可以选择性地创建"XML 源"节点，以便将导出的数据重新读取到流中。

示例

```
stream = modeler.script.stream()
xmlexportnode = stream.createAt("xmlexport", "XML Export", 200, 200)
xmlexportnode.setPropertyValue("full_filename", "c:/export/data.xml")
xmlexportnode.setPropertyValue("map", [["/catalog/book/genre", "genre"],
["/catalog/book/title", "title"]])
```

表 *249. xmlexportnode* 属性

xmlexportnode 属性	**数据类型**	**属性描述**
full_filename	*string*	（必需）XML 导出文件的完整路径和文件名。
use_xml_schema	*flag*	指定是否使用 XML 模式（XSD 或 DTD 文件）控制导出数据的结构。
full_schema_filename	*string*	要使用的 XSD 或 DTD 文件的完整路径和文件名。如果 use_xml_schema 设为 true，那么为必需。
generate_import	*flag*	生成用于将导出的数据文件读回到流中的"XML 源"节点。
records	*string*	表示记录边界的 XPath 表达式。
map	*string*	将字段名映射到 XML 结构。

第 18 章 IBM SPSS Statistics 节点属性

statisticsimportnode 属性

Statistics 文件节点从 IBM SPSS Statistics 使用的 *.sav* 或 *.zsav* 文件格式以及保存在 IBM SPSS Modeler 中的高速缓存文件（也使用同一格式）读取数据。

示例

```
stream = modeler.script.stream()
statisticsimportnode = stream.createAt("statisticsimport", "SAV Import", 200, 200)
statisticsimportnode.setPropertyValue("full_filename", "C:/data/drug1n.sav")
statisticsimportnode.setPropertyValue("import_names", True)
statisticsimportnode.setPropertyValue("import_data", True)
```

表 *250. statisticsimportnode* 属性.

statisticsimportnode 属性	**数据类型**	**属性描述**
full_filename	字符串	完整文件名（包括路径）。
password	字符串	密码。必须在 file_encrypted 参数之前设置 password 参数。
file_encrypted	标志	此文件是否受密码保护。
import_names	NamesAndLabels LabelsAsNames	处理变量名和标签的方法。
import_data	DataAndLabels LabelsAsData	处理值和标签的方法。
use_field_format_for_storage	布尔值	指定导入时是否使用 IBM SPSS Statistics 字段格式信息。

statisticstransformnode 属性

"Statistics 转换"节点针对 IBM SPSS Modeler 中的数据源运行选择的 IBM SPSS Statistics 语法命令。此节点需要 IBM SPSS Statistics 的许可副本。

示例

```
stream = modeler.script.stream()
statisticstransformnode = stream.createAt("statisticstransform", "Transform", 200, 200)
statisticstransformnode.setPropertyValue("syntax", "COMPUTE NewVar = Na + K.")
statisticstransformnode.setKeyedPropertyValue("new_name", "NewVar", "Mixed Drugs")
statisticstransformnode.setPropertyValue("check_before_saving", True)
```

表 *251. statisticstransformnode* 属性

statisticstransformnode 属性	**数据类型**	**属性描述**
syntax	*string*	
check_before_saving	*flag*	保存输入项之前验证已输入的语法。如果语法无效，那么会显示一条错误消息。
default_include	*flag*	有关更多信息，请参阅第 129 页的『filternode 属性』主题。
include	*flag*	有关更多信息，请参阅第 129 页的『filternode 属性』主题。
new_name	*string*	有关更多信息，请参阅第 129 页的『filternode 属性』主题。

statisticsmodelnode 属性

"Statistics 模型"节点使您能够通过运行将会生成 PMML 的 IBM SPSS Statistics 过程来分析和处理数据。此节点需要 IBM SPSS Statistics 的许可副本。

示例

```
stream = modeler.script.stream()
statisticsmodelnode = stream.createAt("statisticsmodel", "Model", 200, 200)
statisticsmodelnode.setPropertyValue("syntax", "COMPUTE NewVar = Na + K.")
statisticsmodelnode.setKeyedPropertyValue("new_name", "NewVar", "Mixed Drugs")
```

statisticsmodelnode 属性	**数据类型**	**属性描述**
syntax	*string*	
default_include	*flag*	有关更多信息，请参阅第 129 页的『filternode 属性』主题。
include	*flag*	有关更多信息，请参阅第 129 页的『filternode 属性』主题。
new_name	*string*	有关更多信息，请参阅第 129 页的『filternode 属性』主题。

statisticsoutputnode 属性

"Statistics 输出"节点用于调用 IBM SPSS Statistics 过程，以分析 IBM SPSS Modeler 数据。可以访问许多不同的 IBM SPSS Statistics 分析过程。此节点需要 IBM SPSS Statistics 的许可副本。

示例

```
stream = modeler.script.stream()
statisticsoutputnode = stream.createAt("statisticsoutput", "Output", 200, 200)
statisticsoutputnode.setPropertyValue("syntax", "SORT CASES BY Age(A) Sex(A) BP(A) Cholesterol(A)")
statisticsoutputnode.setPropertyValue("use_output_name", False)
statisticsoutputnode.setPropertyValue("output_mode", "File")
statisticsoutputnode.setPropertyValue("full_filename", "Cases by Age, Sex and Medical History")
statisticsoutputnode.setPropertyValue("file_type", "HTML")
```

表 252. *statisticsoutputnode* 属性

statisticsoutputnode 属性	**数据类型**	**属性描述**
mode	Dialog Syntax	选择"IBM SPSS Statistics 对话框"选项或语法编辑器
syntax	*string*	
use_output_name	*flag*	
output_name	*string*	
output_mode	Screen File	
full_filename	*string*	
file_type	HTML SPV SPW	

statisticsexportnode 属性

Statistics 导出节点以 IBM SPSS Statistics *.sav* 或 *.zsav* 格式输出数据。IBM SPSS Statistics Base 和其他产品可以读取 *.sav* 或 *.zsav* 文件。这种格式也用于 IBM SPSS Modeler 中的某些缓存文件。

示例

```
stream = modeler.script.stream()
statisticsexportnode = stream.createAt("statisticsexport", "Export", 200, 200)
statisticsexportnode.setPropertyValue("full_filename", "c:/output/SPSS_Statistics_out.sav")
statisticsexportnode.setPropertyValue("field_names", "Names")
statisticsexportnode.setPropertyValue("launch_application", True)
statisticsexportnode.setPropertyValue("generate_import", True)
```

表 253. *statisticsexportnode* 属性.

statisticsexportnode 属性	**数据类型**	**属性描述**
full_filename	*string*	
file_type	sav zsav	以 *sav* 或 *zsav* 格式保存文件。例如: statisticsexportnode.setPropertyValue("file_type","sav")
encrypt_file	*flag*	此文件是否受密码保护。
password	*string*	密码。
launch_application	*flag*	
export_names	NamesAndLabels NamesAsLabels	用于将字段名从 IBM SPSS Modeler 的导出中映射到 IBM SPSS Statistics 或 SAS 的变量名中。

表 253. *statisticsexportnode* 属性 *(续)*.

statisticsexportnode 属性	**数据类型**	**属性描述**
generate_import	*flag*	

第 19 章 Python 节点属性

ocsvmnode 属性

单类 SVM 节点使用无监督学习算法。此节点可用于新内容检测。它将检测指定样本集的软边界，以便按是否属于该集合对新点进行分类。此单类 SVM 建模节点在 SPSS Modeler 中使用 Python 进行实现并且需要 scikit-learn© Python 库。

表 *254. ocsvmnode* 属性

ocsvmnode 属性	**数据类型**	**属性说明**
role_use	字符串	指定 predefined 以使用预定义角色，或者指定 custom 以使用定制字段分配。缺省为 predefined。
输入	*field*	这是输入的字段名称。
分割	*field*	这是用于分隔的字段名称的列表。
use_partition	布尔值	请指定 true 或 false。缺省值为 true。如果设置为 true，那么在构建模型时，将仅使用训练数据。
mode_type	字符串	这是模式。可能的值为简单或专家。如果指定了简单，那么"专家"选项卡上的所有参数将处于禁用状态。
stopping_criteria	字符串	这是科学记数法字符串。可能的值为 1.0E-1、1.0E-2、1.0E-3、1.0E-4、1.0E-5 或 1.0E-6。缺省值为 1.0E-3。
precision	浮点数	回归精度 (nu)。训练错误和支持向量的尾数边界。请指定大于 0 并小于或等于 1.0 的数字。缺省值为 0.1。
kernel	字符串	要用在算法中的内核类型。可能的值为线性、多项式、rbf、sigmoid 或预先计算。缺省值为 rbf。
enable_gamma	布尔值	用于启用伽玛参数。请指定 true 或 false。缺省值为 true。
gamma	浮点数	仅对内核 rbf、多项式和 sigmoid 启用此参数。如果 enable_gamma 参数设置为 false，那么此参数将设置为自动。如果设置为 true，那么缺省值为 0.1。
coef0	浮点数	这是内核函数中的独立项。仅对多项式内核和 sigmoid 内核启用此参数。缺省值为 0.0。
degree	整数	多项式内核函数的次数。仅对多项式内核启用此参数。请指定任何整数。缺省值为 3。
shrinking	布尔值	用于指定是否要使用缩小启发式选项。请指定 true 或 false。缺省值为 false。

表 254. *ocsvmnode* 属性（续）

ocsvmnode 属性	数据类型	属性说明
enable_cache_size	布尔值	用于启用 cache_size 参数。请指定 true 或 false。缺省值为 false。
cache_size	浮点数	这是内核高速缓存的大小 (MB)。缺省值为 200。
enable_random_seed	布尔值	用于启用 random_seed 参数。请指定 true 或 false。缺省值为 false。
random_seed	整数	这是对数据进行排列来估算可能性时要使用的随机数种子。请指定任何整数。
pc_type	字符串	这是并行坐标图形的类型。可能的选项为独立或一般。
lines_amount	整数	这是要包含在图形上的最大行数。请指定介于 1 和 1000 之间的整数。
lines_fields_custom	布尔值	用于启用 lines_fields 参数，您可以通过此参数指定要显示在图形输出中的定制字段。如果设置为 false，那么将显示所有字段。如果设置为 true，那么仅显示使用 lines_fields 参数指定的字段。为了提高性能，最多将显示 20 个字段。
lines_fields	*field*	这是要作为纵轴包含在图形上的字段名称的列表。
enable_graphic	布尔值	请指定 true 或 false。启用图形输出（如果希望节省时间和减小流文件大小，请禁用此选项）。
enable_hpo	布尔值	指定 true 或 false 以启用或禁用 HPO 选项。如果设置为 true，那么将应用 Rbfopt 以自动查找"最佳"一类 SVM 模型，这达到了用户使用以下 target_objval 参数定义的目标值。
target_objval	浮点数	要达到的目标函数值（样本上模型的错误率），例如，未知最优项的值。如果最优值为未知，请将此参数设置为相应值（例如，0.01）。
max_iterations	整数	尝试模型的最大迭代数。缺省值为 1000。
max_evaluations	整数	尝试模型的最大函数求值次数，其中焦点为基于速度的精度。缺省值为 300。

rfnode 属性

随机林节点使用将树模型用作基本模型的梯度提升算法的高级实现。此随机林建模节点在 SPSS Modeler 中使用 Python 进行实现并且需要 scikit-learn© Python 库。

表 255. *rfnode* 属性

rfnode 属性	数据类型	属性描述
role_use	字符串	指定 predefined 以使用预定义角色，或者指定 custom 以使用定制字段分配。缺省为 predefined。

表 255. *rfnode* 属性 (续)

rfnode 属性	**数据类型**	**属性描述**
输入	*field*	这是输入的字段名称。
分割	*field*	这是用于分隔的字段名称的列表。
n_estimators	整数	要构建的树的数量。缺省值为 10。
specify_max_depth	布尔值	指定定制最大长度。如果为 false，那么会扩展节点，直到所有叶为纯叶，或者所有叶包含的样本少于 min_samples_split。缺省值为 false。
max_depth	整数	树的最大深度。缺省值为 10。
min_samples_leaf	整数	最小叶节点大小。缺省值为 1。
max_features	字符串	查找最佳分割时要考虑的功能部件数： • 如果为 auto，那么针对分类器为 max_features=sqrt(n_features)，针对回归为 max_features=sqrt(n_features)。 • 如果为 sqrt，那么 max_features=sqrt(n_features)。 • 如果为 log2，那么 max_features=log2(n_features)。 缺省值为自动。
bootstrap	布尔值	构建树时使用 bootstrap 样本。缺省值为 true。
oob_score	布尔值	使用 out-of-bag 样本估算泛化关系准确度。缺省值为 false。
extreme	布尔值	使用极度随机化树。缺省值为 false。
use_random_seed	布尔值	指定此项以获取复制的结果。缺省值为 false。
random_seed	整数	构建树时要使用的随机数种子。请指定任何整数。
cache_size	浮点数	这是内核高速缓存的大小 (MB)。缺省值为 200。
enable_random_seed	布尔值	用于启用 random_seed 参数。指定 true 或 false。缺省值为 false。
enable_hpo	布尔值	指定 true 或 false 以启用或禁用 HPO 选项。如果设置为 true，那么将应用 Rbfopt 以自动确定"最佳"随机林模型，这达到了用户使用以下 target_objval 参数定义的目标值。
target_objval	浮点数	要达到的目标函数值（样本上模型的错误率），例如，未知最优项的值。如果最优值为未知，请将此参数设置为相应值（例如，0.01）。
max_iterations	整数	尝试模型的最大迭代数。缺省值为 1000。
max_evaluations	整数	尝试模型的最大函数求值次数，其中焦点为基于速度的精度。缺省值为 300。

tsnenode 属性

t-分布随机邻近嵌入 (t-SNE) (t-SNE) 是一款用于可视化高维数据的工具。此工具可将数据点的亲缘关系转换为概率。此 t-SNE 节点在 SPSS Modeler 中使用 Python 进行实现并且需要 scikit-learn© Python 库。

表 256. *tsnenode* 属性

tsnenode 属性	数据类型	属性描述
mode_type	字符串	指定 simple 或 expert 方式。
n_components	字符串	嵌入空间的维度（2D 或 3D）。指定 2 或 3。缺省值为 2。
method	字符串	指定 barnes_hut 或 exact。缺省值为 barnes_hut。
init	字符串	初始化嵌入。指定 random 或 pca。缺省值为 random。
target_field	字符串	目标字段名称。它可为输出图形上的颜色映射图。如果未指定目标字段，那么此图形将使用一种颜色。
perplexity	浮点数	perplexity 与其他流形学习算法中使用的最近相邻元素数相关。通常，数据集越大，所需的 perplexity 也越大。考虑选择 5 到 50 之间的值。缺省值为 30。
early_exaggeration	浮点数	控制原始空间中自然集群在嵌入空间中紧密程度以及集群之间的空间。缺省值为 12.0。
learning_rate	浮点数	缺省值为 200。
n_iter	整数	优化的最大迭代次数。设置为至少 250。缺省值为 1000。
angle	浮点数	从某个点度量的距离节点的角度大小。指定范围为 0-1 的值。缺省值为 0.5。
enable_random_seed	布尔值	设置为 true 以启用 random_seed 参数。缺省值为 false。
random_seed	整数	要使用的随机数种子。缺省值为 None。
n_iter_without_progress	整数	没有进度的最大迭代次数。缺省值为 300。

表 256. *tsnenode* 属性 (续)

tsnenode 属性	数据类型	属性描述
min_grad_norm	字符串	如果梯度标准值低于此阈值，那么优化将停止。缺省值为 1.0E-7。可能的值为： • 1.0E-1 • 1.0E-2 • 1.0E-3 • 1.0E-4 • 1.0E-5 • 1.0E-6 • 1.0E-7 • 1.0E-8
isGridSearch	布尔值	设置为 true 以执行具有多个不同复杂度的 t-SNE。缺省值为 false。
output_Rename	布尔值	如果要提供定制名称，请指定 true，或者如果要自动对输出命名，请指定 false。缺省值为 false。
output_to	字符串	指定 Screen 或 Output。缺省值为 Screen。
full_filename	字符串	指定输出文件名。
output_file_type	字符串	输出文件格式。指定 HTML 或 Output object。缺省值为 HTML。

smotenode 属性

合成少数类过采样技术 (Synthetic Minority Over-sampling Technique, SMOTE) 节点提供了用于处理不平衡数据集的过采样算法。它提供了用于均衡数据的高级方法。SMOTE 过程节点在 SPSS Modeler 中使用 Python 进行实现并且需要 imbalanced-learn© Python 库。

表 257. *smotenode* 属性

smotenode 属性	数据类型	属性说明
target_field	*field*	目标字段。
sample_ratio	字符串	用于启用定制比率值。两个选项分别为"自动"(sample_ratio_auto) 和"设置比率"(sample_ratio_manual)。
sample_ratio_value	浮点数	此比率是少数类中的样本数与多数类中的样本数之比。它必须大于 0 并小于或等于 1。缺省值为自动。
enable_random_seed	布尔值	如果设置为 true，那么会启用 random_seed 属性。
random_seed	整数	这是由随机数字生成器使用的种子。
k_neighbours	整数	这是要用于构建合成样本的最近邻居的数量。缺省值为 5。

表 257. *smotenode* 属性 (续)

smotenode 属性	数据类型	属性说明
m_neighbours	整数	这是要用于确定是否少数样本处于危险状态的最近邻居的数量。只有在 SMOTE 算法类型为 borderline1 和 borderline2 时，才启用此选项。缺省值为 10。
algorithm_kind	字符串	SMOTE 算法的类型：regular、borderline1 或 borderline2。
usepartition	布尔值	如果设置为 true，那么仅将训练数据用于模型构建。缺省值为 true。

xgboostlinearnode 属性

XGBoost Linear© 是将线性模型用作基本模型的梯度提升算法的高级实现。提升算法以迭代方式学习弱分类器，然后将它们添加到最终的强分类器中。SPSS Modeler 中的 XGBoost Linear 节点使用 Python 进行实现。

表 258. *xgboostlinearnode* 属性

xgboostlinearnode 属性	数据类型	属性说明
TargetField	*field*	
InputFields	*field*	
alpha	*Double*	这是 alpha 线性提升系数参数。请指定任何数字，0 或更大值。缺省值为 0。
lambda	*Double*	这是 lambda 线性提升系数参数。请指定任何数字，0 或更大值。缺省值为 1。
lambdaBias	*Double*	这是 lambda 偏差线性提升系数参数。请指定任何数。缺省值为 0。
numBoostRound	整数	模型构建的提升舍入次数值。请指定介于 1 和 1000 之间的值。缺省值为 10。
objectiveType	字符串	学习任务的目标类型。可能的值为 reg:linear、reg:logistic、reg:gamma、reg:tweedie、count:poisson、rank:pairwise、binary:logistic 或 multi。请注意，对于标志目标，只能使用 binary:logistic 或 multi。如果使用了 multi，那么评分结果将显示 multi:softmax 和 multi:softprob XGBoost 目标类型。
random_seed	整数	随机数种子。任何介于 0 和 9999999 之间的数字。缺省值为 0。
useHPO	布尔值	指定 true 或 false 以启用或禁用 HPO 选项。如果设置为 true，那么将应用 Rbfopt 以自动查找"最佳"一类 SVM 模型，这将达到用户使用 target_objval 参数定义的目标值。

xgboosttreenode 属性

XGBoost Tree© 是将树模型用作基本模型的梯度提升算法的高级实现。提升算法以迭代方式学习弱分类器，然后将它们添加到最终的强分类器中。XGBoost Tree 具有很高的灵活性，并提供了很多对于大多数用户来说过于复杂的参数，因此 SPSS Modeler 中的 XGBoost Tree 节点仅显示了核心功能和常用参数。此节点使用 Python 进行实现。

表 *259. xgboosttreenode* 属性

xgboosttreenode 属性	**数据类型**	**属性说明**
TargetField	*field*	目标字段。
InputFields	*field*	输入字段。
treeMethod	字符串	模型构建的树方法。可能的值为 auto、exact 或 approx。缺省值为 auto。
numBoostRound	整数	模型构建的提升舍入次数值。请指定介于 1 和 1000 之间的值。缺省值为 10。
maxDepth	整数	树增长的最大深度。请指定值 1 或更高值。缺省值为 6。
minChildWeight	*Double*	树增长的最小子代权重。请指定值 0 或更高值。缺省值为 1。
maxDeltaStep	*Double*	树增长的最大变化量步骤。请指定值 0 或更高值。缺省值为 0。
objectiveType	字符串	学习任务的目标类型。可能的值为 reg:linear、reg:logistic、reg:gamma、reg:tweedie、count:poisson、rank:pairwise、binary:logistic 或 multi。请注意，对于标志目标，只能使用 binary:logistic 或 multi。如果使用了 multi，那么评分结果将显示 multi:softmax 和 multi:softprob XGBoost 目标类型。
random_seed	整数	随机数种子。任何介于 0 和 9999999 之间的数字。缺省值为 0。
sampleSize	*Double*	用于控制过度拟合的子样本。请指定介于 0.1 和 1.0 之间的值。缺省值为 0.1。
eta	*Double*	用于控制过度拟合的 eta。请指定介于 0 和 1 之间的值。缺省值为 0.3。
gamma	*Double*	用于控制过度拟合的伽玛。请指定任何数字，0 或更大值。缺省值为 6。
colsSampleRatio	*Double*	用于控制过度拟合的列样本（按树列出）。请指定介于 0.01 和 1 之间的值。缺省值为 1。
colsSampleLevel	*Double*	用于控制过度拟合的列样本（按级别列出）。请指定介于 0.01 和 1 之间的值。缺省值为 1。
lambda	*Double*	用于控制过度拟合的 lambda。请指定任何数字，0 或更大值。缺省值为 1。
alpha	*Double*	用于控制过度拟合的 alpha。请指定任何数字，0 或更大值。缺省值为 0。

表 *259. xgboosttreenode* 属性 *(续)*

xgboosttreenode 属性	**数据类型**	**属性说明**
scalePosWeight	*Double*	用于处理不平衡数据集的刻度位置权重。缺省值为 1。

第 20 章 Spark 节点属性

isotonicasnode 属性

保序回归属于回归算法系列。SPSS Modeler 中的 Isotonic-AS 节点使用 Spark 进行实现。有关保序回归算法的详细信息，请参阅 https://spark.apache.org/docs/2.2.0/mllib-isotonic-regression.html。

表 *260. isotonicasnode* 属性

isotonicasnode 属性	**数据类型**	**属性描述**
label	字符串	该属性是要为其计算保序回顾的因变量。
features	字符串	该属性是自变量。
weightCol	字符串	权重表示多个测量值。缺省值为 1。
isotonic	布尔值	该属性指示类型是 isotonic 还是 antitonic。
featureIndex	整数	如果 featuresCol 是向量列，该属性表示特征索引。缺省值为 0。

xgboostasnode 属性

XGBoost 是梯度提升算法的高级实现。提升算法以迭代方式学习弱分类器，然后将它们添加到最终的强分类器中。XGBoost 具有很高的灵活性，并提供了很多对于大多数用户来说过于复杂的参数，因此 SPSS Modeler 中的 XGBoost-AS 节点仅显示了核心功能和常用参数。XGBoost-AS 节点使用 Spark 进行实现。

表 *261. xgboostasnode* 属性

xgboostasnode 属性	**数据类型**	**属性描述**
target_field	*field*	这是用于目标的字段名称的列表。
input_fields	*field*	这是用于输入的字段名称的列表。
nWorkers	整数	这是用于训练 XGBoost 模型的工作程序数目。缺省值为 1。
numThreadPerTask	整数	每个工作程序使用的线程数。缺省值为 1。
useExternalMemory	布尔值	是否使用外部内存作为缓存。缺省值为 **false**。
boosterType	字符串	要使用的提升系数类型。可用选项包括 gbtree、gblinear 或 dart。缺省值为 gbtree。
numBoostRound	整数	提升的次数。请指定值 0 或更高值。缺省值为 10。
scalePosWeight	*Double*	控制正权重和负权重的平衡。缺省值为 1。

表 *261. xgboostasnode* 属性 *(续)*

xgboostasnode 属性	**数据类型**	**属性描述**
randomseed	整数	这是由随机数字生成器使用的种子。缺省值为 **0**。
objectiveType	字符串	学习目标。可能的值为 reg:linear、reg:logistic、reg:gamma、reg:tweedie、count:poisson、rank:pairwise、binary:logistic 或 multi。请注意，对于标志目标，只能使用 binary:logistic 或 multi。如果使用了 multi，那么评分结果将显示 multi:softmax 和 multi:softprob XGBoost 目标类型。缺省值为 reg:linear。
evalMetric	字符串	验证数据的评估度量。将根据目标分配缺省度量。可能的值为 rmse、mae、logloss、error、merror、mlogloss、auc、ndcg、map 或 gamma-deviance。缺省值为 rmse。
lambda	*Double*	关于权重的 L2 规则化术语。增大此值将使模型更保守。请指定任何数字，0 或更大值。缺省值为 1。
alpha	*Double*	关于权重的 L1 规则化术语。增大此值将使模型更保守。请指定任何数字，0 或更大值。缺省值为 0。
lambdaBias	*Double*	关于偏差的 L2 规则化术语。如果使用 gblinear 提升系数类型，那么可以使用该 lambda 偏差线性提升系数参数。请指定任何数字，0 或更大值。缺省值为 0。
treeMethod	字符串	如果使用了 gbtree 或 dart 提升系数类型，用于树生长的该树方法参数（以及后续其他树参数）可用。它指定要使用的 XGBoost 树构造算法。可用选项包括 auto、exact 或 approx。缺省值为 auto。
maxDepth	整数	树的最大深度。请指定值 2 或更高值。缺省值为 6。
minChildWeight	*Double*	子代中需要的实例权重 (hessian) 的最小和。请指定值 0 或更高值。缺省值为 1。
maxDeltaStep	*Double*	允许用于每个树权重估计的最大增量步骤。请指定值 0 或更高值。缺省值为 0。
sampleSize	*Double*	子样本用于训练实例比率。请指定介于 0.1 和 1.0 之间的值。缺省值为 1.0。
eta	*Double*	更新步骤中用于放置过度拟合的步长收缩。请指定介于 0 和 1 之间的值。缺省值为 0.3。
gamma	*Double*	这是对树的某个叶节点进行进一步分区所需的最小损失减小。请指定任何数字，0 或更大值。缺省值为 6。
colsSampleRatio	*Double*	构造每个树时列的子样本比率。请指定介于 0.01 和 1 之间的值。缺省值为 1。
colsSampleLevel	*Double*	在每个级别中，针对每个分隔的列的子样本比率。请指定介于 0.01 和 1 之间的值。缺省值为 1。

表 *261. xgboostasnode* 属性 *(续)*

xgboostasnode 属性	**数据类型**	**属性描述**
normalizeType	字符串	如果使用 dart 提升系数类型，该 dart 参数和后续三个 dart 参数可用。该参数设置标准化算法。请指定 tree 或 forest。缺省值为 tree。
sampleType	字符串	采样算法类型。指定 uniform 或 weighted。缺省值为 uniform。
rateDrop	*Double*	丢弃率 dart 提升系数参数。请指定介于 0.0 和 1.0 之间的值。缺省值为 0.0。
skipDrop	*Double*	用于跳过丢弃的概率的 dart 提升系数参数。请指定介于 0.0 和 1.0 之间的值。缺省值为 0.0。

第 21 章 超节点属性

下列各表描述特定于超节点的属性。注意公共节点属性也可应用于超节点。

表 *262.* 终端超节点属性

属性名称	属性类型/值列表	属性描述
execute_method	Script Normal	
script	*string*	

超节点参数

可使用通用格式在脚本中创建或设置超节点参数：

```
mySuperNode.setParameterValue("minvalue", 30)
```

您可以使用以下内容检索参数值：

```
value mySuperNode.getParameterValue("minvalue")
```

查找现有超节点

您可以使用 findByType() 函数在流中查找超节点：

```
source_supernode = modeler.script.stream().findByType("source_super", None)
process_supernode = modeler.script.stream().findByType("process_super", None)
terminal_supernode = modeler.script.stream().findByType("terminal_super", None)
```

设置已封装节点的属性

可以通过访问某个超节点中的子图来设置封装在该超节点中特定节点的属性。例如，假设有一个源超节点，其中封装有变量文件节点以读取数据。可以通过访问子图并查找相关节点来传递要读取的文件的名称（使用 full_filename 属性指定），如下所示：

```
childDiagram = source_supernode.getChildDiagram()
varfilenode = childDiagram.findByType("variablefile", None)
varfilenode.setPropertyValue("full_filename", "c:/mydata.txt")
```

创建超节点

如果您要从头开始创建超节点及其内容，那么可以通过创建超节点、访问子图并创建所需节点来以类似方式完成此操作。还必须确保超节点图中的节点也链接到输入 - 和/或输出连接器节点。例如，如果您要创建过程超节点：

```
process_supernode = modeler.script.stream().createAt("process_super", "My SuperNode", 200, 200)
childDiagram = process_supernode.getChildDiagram()
filternode = childDiagram.createAt("filter", "My Filter", 100, 100)
childDiagram.linkFromInputConnector(filternode)
childDiagram.linkToOutputConnector(filternode)
```

附录 A. 节点名引用

此部分提供 IBM SPSS Modeler 中节点的脚本编制名称引用。

模型块名称

模型块（也称为生成的模型）可以按类型进行引用，就好像节点和输出对象一样。下表列出模型对象的引用名称。

请注意，这些名称专用于引用模型选用板（位于 IBM SPSS Modeler 窗口的右上角）中的模型块。要引用已经添加到流中进行评分的模型节点，那么使用另外一套以 `apply...` 为前缀的名称。有关更多信息，请参阅模型块节点属性主题。

注：在通常情况下，建议按名称和类型来引用模型，以避免引起混淆。

表 *263.* 模型块名称（建模选用板）.

模型名称	模型
anomalydetection	异常
apriori	Apriori
autoclassifier	自动分类器
autocluster	自动聚类
autonumeric	自动数字
bayesnet	贝叶斯网络
c50	C5.0
carma	Carma
cart	C&R 树
chaid	CHAID
coxreg	Cox 回归
decisionlist	决策列表
discriminant	判别
factor	PCA/因子
featureselection	特征选择
genlin	广义线性回归
glmm	GLMM
kmeans	K-Means
knn	*k*-最近相邻元素
kohonen	Kohonen
linear	线性
logreg	Logistic 回归
neuralnetwork	类神经网络
quest	QUEST
regression	线性回归

表 *263.* 模型块名称（建模选用板） *(续).*

模型名称	模型
sequence	序列
slrm	自学响应模型
statisticsmodel	IBM SPSS Statistics 模型
SVM	支持向量机
timeseries	时间序列
twostep	二阶

表 *264.* 模型块名称（数据库建模选用板）.

模型名称	模型
db2imcluster	IBM ISW 聚类
db2imlog	IBM ISW Logistic 回归
db2imnb	IBM ISW 朴素贝叶斯
db2imreg	IBM ISW 回归
db2imtree	IBM ISW 决策树
msassoc	MS 关联规则
msbayes	MS 朴素贝叶斯
mscluster	MS 聚类
mslogistic	MS Logistic 回归
msneuralnetwork	MS 神经网络
msregression	MS 线性回归
mssequencecluster	MS 序列聚类
mstimeseries	MS 时间序列
mstree	MS 决策树
netezzabayes	Netezza Bayes 网络
netezzadectree	Netezza 决策树
netezzadivcluster	Netezza 分裂式聚类
netezzaglm	Netezza 广义线性
netezzakmeans	Netezza K-Means
netezzaknn	Netezza KNN
netezzalineregression	Netezza 线性回归
netezzanaivebayes	Netezza 朴素贝叶斯
netezzapca	Netezza PCA
netezzaregtree	Netezza 回归树
netezzatimeseries	Netezza 时间序列
oraabn	Oracle Adaptive Bayes
oraai	Oracle AI
oradecisiontree	Oracle 决策树
oraglm	Oracle GLM
orakmeans	Oracle *k*-Means
oranb	Oracle 朴素贝叶斯

表 *264.* 模型块名称（数据库建模选用板） *(续).*

模型名称	模型
oranmf	Oracle NMF
oraocluster	Oracle O-Cluster
orasvm	Oracle SVM

避免重复的模型名称

使用脚本对生成的模型进行操作时，务必注意：允许重复的模型名称可能会导致歧义引用。为了避免这种情况的发生，最好在编制脚本时要求对生成的模型使用唯一的名称。

要为重复模型名称设置选项：

1. 从菜单中选择：

   工具 > **用户选项**
2. 单击**通知**选项卡。
3. 选择**替换原有模型**以限制生成的模型的重复命名。

存在不明确的模型引用时，脚本执行行为在 SPSS Modeler 与 IBM SPSS Collaboration and Deployment Services 之间可能有所不同。SPSS Modeler 客户机提供了"替换先前模型"选项，此选项将自动替换同名的模型（例如，脚本通过循环执行迭代，以便每次都生成不同的模型）。但是，在 IBM SPSS Collaboration and Deployment Services 中运行同一脚本时，此选项不可用。通过将每次迭代中生成的模型重命名以避免对模型进行不明确的引用，或者通过在循环结束前清除当前模型（例如，添加 `clear generated palette` 语句），可以避免这种情况。

输出类型名称

下表列出了所有的输出对象类型和创建它们的节点。有关每种输出类型可用的导出格式完整列表，请参阅创建该输出类型的节点的属性描述（图形节点公共属性和输出节点属性）。

表 *265.* 输出对象类型以及创建这些类型的节点.

输出对象类型	节点(N)
analysisoutput	分析
collectionoutput	集合
dataauditoutput	数据审核
distributionoutput	分布
evaluationoutput	评估
histogramoutput	直方图
matrixoutput	矩阵
meansoutput	平均值
multiplotoutput	多重散点图
plotoutput	散点图
qualityoutput	质量
reportdocumentoutput	此对象类型不属于节点；它是工程报告创建的输出
reportoutput	报告

表 *265.* 输出对象类型以及创建这些类型的节点 *(续).*

输出对象类型	节点(N)
statisticsprocedureoutput	Statistics 输出
statisticsoutput	统计
tableoutput	表
timeplotoutput	时间散点图
weboutput	Web

附录 B. 从旧脚本编制迁移到 Python 脚本编制

旧脚本迁移概述

本节提供 IBM SPSS Modeler 中 Python 脚本编制与旧脚本编制之间的差异摘要，并提供有关如何将旧脚本迁移为 Python 脚本的信息。在本节中，您将找到标准 SPSS Modeler 旧命令和等效的 Python 命令的列表。

一般差异

旧脚本编制的设计在很大程度上借鉴了操作系统命令脚本。尽管包含一些块结构（例如 `if...then...else...endif` 和 `for...endfor`），但旧脚本编制面向行，并且缩进通常没有意义。

在 Python 脚本编制中，缩进有意义，并且属于同一逻辑块的行必须在同一级别进行缩进。

注：复制和粘贴 Python 代码时，请务必小心操作。在编辑器中，使用 tab 缩进的行可能与使用空格缩进的行看起来一样。但是，Python 脚本将生成错误，这是因为未将这些行视作缩进相同的行。

脚本编制上下文

脚本编制上下文定义了将在其中执行脚本的环境，例如，用于执行脚本的流或超节点。例如，在旧脚本编制中，上下文是隐式的，这意味着假定流脚本中的所有节点引用都包含在执行该脚本的流中。

在 Python 脚本编制中，脚本编制上下文通过 `modeler.script` 模块以显式方式提供。例如，Python 流脚本可以使用以下代码访问执行该脚本的流：

```
s = modeler.script.stream()
```

然后，可以通过返回的对象来调用与流相关的函数。

命令与函数

旧脚本编制面向命令。这意味着脚本的每一行通常以后跟参数进行运行的命令开始，例如：

```
connect 'Type':typenode to :filternode
rename :derivenode as "Compute Total"
```

Python 使用通常通过定义函数的对象（模块、类或对象）所调用的函数，例如：

```
stream = modeler.script.stream()
typenode = stream.findByType("type", "Type)
filternode = stream.findByType("filter", None)
stream.link(typenode, filternode)
derive.setLabel("Compute Total")
```

文字和注释

IBM SPSS Modeler 中一些常用的文字和注释命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 *266.* 文字和注释的旧脚本编制到 *Python* 脚本编制的映射.

旧脚本编制	Python 脚本编制
整数，例如 `4`	相同
浮点数，例如 `0.003`	相同
加单引号的字符串，例如 `'Hello'`	相同 **注：** 包含非 ASCII 字符的字符串字面值必须以 `u` 作为前缀，以确保它们表示为 Unicode。
加双引号的字符串，例如 `"Hello again"`	相同 **注：** 包含非 ASCII 字符的字符串字面值必须以 `u` 作为前缀，以确保它们表示为 Unicode。
长字符串，例如 `"""This is a string that spans multiple lines"""`	相同
列表，例如 `[1 2 3]`	`[1, 2, 3]`
变量引用，例如 `set x = 3`	`x = 3`
行继续符 (\)，例如 `set x = [1 2 \ 3 4]`	`x = [ 1, 2,\ 3, 4]`
块注释，例如 `/* This is a long comment over a line.*/`	`""" This is a long comment over a line."""`
行注释，例如 `set x = 3 # make x 3`	`x = 3 # make x 3`
`undef`	无
`true`	`True`
`false`	`False`

运算符

IBM SPSS Modeler 中一些常用的运算符命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 *267.* 运算符的旧脚本编制到 *Python* 脚本编制的映射.

旧脚本编制	Python 脚本编制

`NUM1 + NUM2`
`LIST + ITEM`
`LIST1 + LIST2`
	`NUM1 + NUM2`
	`LIST.append(ITEM)`
	`LIST1.extend(LIST2)`

`NUM1 – NUM2` `LIST - ITEM`	`NUM1 – NUM2` `LIST.remove(ITEM)`
`NUM1 * NUM2`	`NUM1 * NUM2`
`NUM1 / NUM2`	`NUM1 / NUM2`
`=` `==`	`==`

表 *267.* 运算符的旧脚本编制到 *Python* 脚本编制的映射 *(续).*

旧脚本编制	Python 脚本编制
/= /==	!=
X ** Y	X ** Y
X < Y X <= Y X > Y X >= Y	X < Y X <= Y X > Y X >= Y
X div Y X rem Y X mod Y	X // Y X % Y X % Y
and or not(EXPR)	and or not EXPR

条件语句和循环

IBM SPSS Modeler 中一些常用的条件和循环命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 *268.* 条件语句和循环的旧脚本编制到 *Python* 脚本编制的映射.

旧脚本编制	Python 脚本编制

for VAR from INT1 to INT2
...
endfor
	for VAR in range(INT1, INT2):
	...
	或
	VAR = INT1
	while VAR <= INT2:
	...
	VAR += 1

for VAR in LIST ... endfor	for VAR in LIST: ...
for VAR in_fields_to NODE ... endfor	for VAR in NODE.getInputDataModel(): ...
for VAR in_fields_at NODE ... endfor	for VAR in NODE.getOutputDataModel(): ...

if...then
...
elseif...then
...
else
...
endif
	if ...:
	...
	elif ...:
	...
	else:
	...

with TYPE OBJECT ... endwith	无等效项
var VAR1	不需要变量声明

变量

在旧脚本编制中，引用变量之前对变量进行了声明，例如：

```
var mynodeset mynode = create typenode at 96 96
```

在 Python 脚本编制中，变量在首次引用时进行创建，例如：

```
mynode = stream.createAt("type", "Type", 96, 96)
```

在旧脚本编制中，必须使用 ^ 运算符显式除去对变量的引用，例如：

```
var mynodeset mynode = create typenode at 96 96
set ^mynode.direction."Age" = Input
```

与大对数脚本编制语言一样，在 Python 脚本编制中，这不是必需操作，例如：

```
mynode = stream.createAt("type", "Type", 96, 96)
mynode.setKeyedPropertyValue("direction","Age","Input")
```

节点、输出和模型类型

在旧脚本编制中，各种对象类型（节点、输出和模型）通常向对象类型追加了类型。例如，"派生"节点具有 derivenode 类型：

```
set feature_name_node = create derivenode at 96 96
```

Python 中的 IBM SPSS Modeler API 未包含 node 后缀，因此"派生"节点具有 derive 类型，例如：

```
feature_name_node = stream.createAt("derive", "Feature", 96, 96)
```

旧脚本编制与 Python 脚本编制的类型名称中的唯一差异在于缺少类型后缀。

属性名

在旧脚本编制和 Python 脚本编制中，属性名相同。例如，在这两种脚本编制环境中，变量文件节点中用于定义文件位置的属性为 full_filename。

节点引用

许多旧脚本使用隐式搜索来查找和访问要修改的节点。例如，下列命令用于在当前流中搜索带有"类型"标签的"类型"节点，然后将"年龄"字段的方向（或建模角色）设置为输入，并将"药品"字段设置为目标（也就是要预测的值）：

```
set 'Type':typenode.direction."Age" = Input
set 'Type':typenode.direction."Drug" = Target
```

在 Python 脚本编制中，必须先显式查找节点对象，然后再调用用于设置属性值的函数，例如：

```
typenode = stream.findByType("type", "Type")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
typenode.setKeyedPropertyValue("direction", "Drug", "Target")
```

注：在本例中，"Target" 必须包含在字符串引号中。

另外，Python 脚本可使用 modeler.api 软件包中的 ModelingRole 枚举。

虽然 Python 脚本编制版本可能更为繁琐，但它能够实现更佳的运行时性能，这是因为通常仅执行一次搜索节点。在旧脚本编制示例中，将针对各个命令搜索节点。

另外，还支持按标识查找节点（可以在节点对话框的"注释"选项卡中查看节点标识）。例如，在旧脚本编制中：

```
# id65EMPB9VL87 is the ID of a Type node
set @id65EMPB9VL87.direction."Age" = Input
```

以下脚本显示 Python 脚本编制中的同一示例：

```
typenode = stream.findByID("id65EMPB9VL87")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
```

获取并设置属性

旧脚本编制使用 set 命令来指定值。set 命令后跟的词汇可以是属性定义。以下脚本显示了两种可能的用于设置属性的脚本格式：

```
set <node reference>.<property> = <value>
set <node reference>.<keyed-property>.<key> = <value>
```

在 Python 脚本编制中，通过使用函数 setPropertyValue() 和 setKeyedPropertyValue()，可以实现同一结果，例如：

```
object.setPropertyValue(property, value)
object.setKeyedPropertyValue(keyed-property, key, value)
```

在旧脚本编制中，可以使用 get 命令来实现访问属性值，例如：

```
var n v
set n = get node :filternode
set v = ^n.name
```

在 Python 脚本编制中，通过使用函数 getPropertyValue()，可以实现同一结果，例如：

```
n = stream.findByType("filter", None)
v = n.getPropertyValue("name")
```

编辑流

在旧脚本编制中，create 命令用于创建新节点，例如：

```
var agg select
set agg = create aggregatenode at 96 96
set select = create selectnode at 164 96
```

在 Python 脚本编制中，流具有多种创建节点的方法，例如：

```
stream = modeler.script.stream()
agg = stream.createAt("aggregate", "Aggregate", 96, 96)
select = stream.createAt("select", "Select", 164, 96)
```

在旧脚本编制中，connect 命令用于创建节点之间的链接，例如：

```
connect ^agg to ^select
```

在 Python 脚本编制中，link 方法用于创建节点之间的链接，例如：

```
stream.link(agg, select)
```

在旧脚本编制中，disconnect 命令用于除去节点之间的链接，例如：

```
disconnect ^agg from ^select
```

在 Python 脚本编制中，unlink 方法用于除去节点之间的链接，例如：

```
stream.unlink(agg, select)
```

在旧脚本编制中，position 命令用于将节点放在流画布上或其他节点之间，例如：

```
position ^agg at 256 256
position ^agg between ^myselect and ^mydistinct
```

在 Python 脚本编制中，通过使用两种不同的方法（setXYPosition 和 setPositionBetween），可以实现同一结果。例如：

```
agg.setXYPosition(256, 256)
agg.setPositionBetween(myselect, mydistinct)
```

节点操作

IBM SPSS Modeler 中一些常用的节点操作命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 *269.* 节点操作的旧脚本编制到 *Python* 脚本编制的映射.

旧脚本编制	Python 脚本编制
create *nodespec* at x y	*stream*.create(*type*, *name*) *stream*.createAt(*type*, *name*, x, y) *stream*.createBetween(*type*, *name*, preNode, postNode) *stream*.createModelApplier(*model*, *name*)
connect *fromNode* to *toNode*	*stream*.link(*fromNode*, *toNode*)
delete *node*	*stream*.delete(*node*)
disable *node*	*stream*.setEnabled(*node*, False)
enable *node*	*stream*.setEnabled(*node*, True)
disconnect *fromNode* from *toNode*	*stream*.unlink(*fromNode*, *toNode*) *stream*.disconnect(*node*)
duplicate *node*	*node*.duplicate()
execute *node*	*stream*.runSelected(*nodes*, *results*) *stream*.runAll(*results*)
flush *node*	*node*.flushCache()
position *node* at x y	*node*.setXYPosition(x, y)
position *node* between *node1* and *node2*	*node*.setPositionBetween(*node1*, *node2*)
rename *node* as *name*	*node*.setLabel(*name*)

循环

在旧脚本编制中，主要支持下列两种循环选项：

- 计数循环，在此循环中，下标变量在两个整数范围之间进行移动。
- 序列循环，此循环对值序列进行遍历，以便将当前值绑定到循环变量。

以下脚本是旧脚本编制中的计数循环示例：

```
for i from 1 to 10
 println ^i
endfor
```

以下脚本是旧脚本编制中的序列循环示例：

```
var items
set items = [a b c d]

for i in items
 println ^i
endfor
```

另外，还可以使用其他类型的循环：

- 对模型选用板中的模型或输出选用板中的输出执行迭代。
- 对传入或传出节点的字段执行迭代。

Python 脚本编制还支持其他类型的循环。以下脚本是 Python 脚本编制中的计数循环示例：

```
i = 1
while i <= 10:
 print i
 i += 1
```

以下脚本是 Python 脚本编制中的序列循环示例：

```
items = ["a", "b", "c", "d"]
for i in items:
 print i
```

序列循环非常灵活，并且在与 IBM SPSS Modeler API 方法结合后，此循环可支持多个脚本编制用例。以下示例显示如何使用 Python 脚本编制中的序列循环对传出节点的字段执行迭代：

```
node = modeler.script.stream().findByType("filter", None)
for column in node.getOutputDataModel().columnIterator():
 print column.getColumnName()
```

执行流

执行流的过程中，生成的模型或输出对象将添加到其中一个对象管理器中。在旧脚本编制中，脚本必须在对象管理器中找到构建对象，或者从生成输出的节点访问最新生成的输出。

在 Python 中，执行流的过程有所不同：执行生成的任何模型或输出将以传递到执行函数的列表形式返回。这使得可以更加轻松地访问流执行结果。

旧脚本编制支持下列三种流执行命令：

- execute_all，用于执行流中的所有可执行终端节点。
- execute_script，用于执行流脚本（与脚本执行的设置无关）。
- execute *node*，用于执行指定的节点。

Python 脚本编制支持一组类似的函数：

- *stream*.runAll(*results-list*)，用于执行流中的所有可执行终端节点。
- *stream*.runScript(*results-list*)，用于执行流脚本（与脚本执行的设置无关）。
- *stream*.runSelected(*node-array*, *results-list*)，用于按节点的提供顺序执行指定的一组节点。
- *node*.run(*results-list*)，用于执行指定的节点。

在旧脚本中，可以使用带有可选整数代码的 exit 命令来终止流执行，例如：

```
exit 1
```

在 Python 脚本编制中，使用以下脚本可实现同一结果：

```
modeler.script.exit(1)
```

通过文件系统和存储库访问对象

在旧脚本编制中，您可以使用 open 命令打开现有流、模型或输出对象，例如：

```
var sset s = open stream "c:/my streams/modeling.str"
```

在 Python 脚本编制中，存在一个可从会话进行访问的 TaskRunner 类，并且这个类可用于执行类似的任务，例如：

```
taskrunner = modeler.script.session().getTaskRunner()
s = taskrunner.openStreamFromFile("c:/my streams/modeling.str", True)
```

在旧脚本编制中，要保存对象，您可以使用 save 命令，例如：

```
save stream s as "c:/my streams/new_modeling.str"
```

等效的 Python 脚本方法将使用 TaskRunner 类，例如：

```
taskrunner.saveStreamToFile(s, "c:/my streams/new_modeling.str")
```

基于 IBM SPSS Collaboration and Deployment Services Repository 的操作在旧脚本编制中通过 retrieve 和 store 命令受支持，例如：

```
var sset s = retrieve stream "/my repository folder/my_stream.str"
store stream ^s as "/my repository folder/my_stream_copy.str"
```

在 Python 脚本编制中，可以通过与会话关联的存储库对象来访问等效功能，例如：

```
session = modeler.script.session()
repo = session.getRepository()
s = repo.retrieveStream("/my repository folder/my_stream.str", None, None, True)
repo.storeStream(s, "/my repository folder/my_stream_copy.str", None)
```

注：存储库访问要求使用有效存储库连接对会话进行了配置。

流操作

IBM SPSS Modeler 中一些常用的流操作命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 *270.* 流操作的旧脚本编制到 *Python* 脚本编制的映射.

旧脚本编制	Python 脚本编制
create stream *DEFAULT_FILENAME*	*taskrunner*.createStream(*name*, *autoConnect*, *autoManage*)
close stream	*stream*.close()
clear stream	*stream*.clear()
get stream *stream*	无等效项
load stream *path*	无等效项
open stream *path*	*taskrunner*.openStreamFromFile(*path*, *autoManage*)
save *stream* as *path*	*taskrunner*.saveStreamToFile(*stream*, *path*)
retreive stream *path*	*repository*.retreiveStream(*path*, *version*, *label*, *autoManage*)
store *stream* as *path*	*repository*.storeStream(*stream*, *path*, *label*)

模型操作

IBM SPSS Modeler 中一些常用的模型操作命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 271. 模型操作的旧脚本编制到 *Python* 脚本编制的映射.

旧脚本编制	Python 脚本编制
`open model path`	*taskrunner*.openModelFromFile(*path*, *autoManage*)
`save model as path`	*taskrunner*.saveModelToFile(*model*, *path*)
`retrieve model path`	*repository*.retrieveModel(*path*, *version*, *label*, *autoManage*)
`store model as path`	*repository*.storeModel(*model*, *path*, *label*)

文档输出操作

IBM SPSS Modeler 中一些常用的文档输出操作命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 272. 文档输出操作的旧脚本编制到 *Python* 脚本编制的映射.

旧脚本编制	Python 脚本编制
`open output path`	*taskrunner*.openDocumentFromFile(*path*, *autoManage*)
`save output as path`	*taskrunner*.saveDocumentToFile(*output*, *path*)
`retrieve output path`	*repository*.retrieveDocument(*path*, *version*, *label*, *autoManage*)
`store output as path`	*repository*.storeDocument(*output*, *path*, *label*)

旧脚本编制与 Python 脚本编制之间的其他差异

旧脚本提供对处理 IBM SPSS Modeler 工程的支持。 Python 脚本编制当前不提供此支持。

旧脚本编制提供了某种装入状态对象（流和模型的组合）的支持。IBM SPSS Modeler 8.0 后的版本不推荐使用状态对象。 Python 脚本编制不支持状态对象。

Python 脚本编制提供了下列附加功能，旧脚本编制中未提供这些功能：

- 类和函数定义
- 错误处理
- 更复杂的输入/输出支持
- 外部模块和第三方模块

声明

本信息是为在美国提供的产品和服务编写的。IBM 可能以其他语言提供本材料。然而，您可能需要拥有产品或产品版本的该语言副本才能进行访问。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往：

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

有关双字节 (DBCS) 信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往：

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

International Business Machines Corporation"按现状"提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些管辖区域在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本出版物中描述的产品进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：(i) 使其能够在独立创建的程序和其它程序（包括本程序）之间进行信息交换，以及 (ii) 使其能够对已经交换的信息进行相互使用，请与下列地址联系：

IBM Director of Licensing
IBM Corporation

North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

所引用的性能数据和客户示例仅作说明用途。实际的性能结果可能会因特定的配置和运营条件而异。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的 资料中获取。 IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。 有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名字都是虚构的，若与实际个人或业务企业相似，纯属巧合。

商标

IBM、IBM 徽标和 ibm.com 是 International Business Machines Corp. 在全球许多行政管辖地区的商标或注册商标。其他产品和服务名称可能是 IBM 或其他公司的商标。IBM 商标的最新列表可从 Web 上的 "Copyright and trademark information"处获得，网址为：www.ibm.com/legal/copytrade.shtml。

Adobe、Adobe 徽标、PostScript 和 PostScript 徽标是 Adobe Systems Incorporated 在美国和/或其他国家或地区的注册商标或商标。

Intel、Intel 徽标、Intel Inside、Intel Inside 徽标、Intel Centrino、Intel Centrino 徽标、Celeron、Intel Xeon、Intel SpeedStep、Itanium 和 Pentium 是 Intel Corporation 或其子公司在美国和其他国家或地区的商标或注册商标。

Linux 是 Linus Torvalds 在美国和/或其他国家或地区的注册商标。

Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。

Java 和所有基于 Java 的商标和徽标是 Oracle 和/或其子公司的商标或注册商标。

产品文档的条款和条件

只有遵守以下条款和条件才会授予使用这些出版物的许可权。

适用性

除用于 IBM Web 站点的任何条款外，还需要遵守这些条款和条件。

个人使用

您可以为了个人使用而非商业性使用复制这些出版物，但前提是保留所有专有权声明。没有 IBM 的明确同意，您不能对这些出版物或者其中的任何部分进行分发、展示或从中派生内容。

商业使用

您仅可在贵公司内部复制、分发和显示这些出版物，但前提是保留所有专有权声明。未经 IBM 的明确许可，您不得制作这些出版物的演绎作品，也不得在贵公司外部复制、分发或显示这些出版物或其部分出版物。

权利

除非本许可权中明确授予，否则不得授予对这些出版物或其中包含的任何信息、数据、软件或其他知识产权的任何许可权、许可证或权利，无论明示的还是暗含的。

只要 IBM 认为这些出版物的使用会损害其利益或者 IBM 判定未正确遵守上述指示信息，IBM 将有权撤销此处授予的许可权。

您不得下载、出口或再出口这些信息，除非完全遵守所有适用的法律和法规（包括所有的美国出口法律和法规）。

IBM 对这些出版物的内容不作任何保证。 本出版物以"按现状"的基础提供，不附有任何形式的（无论是明示的，还是暗含的）保证，包括但不限于暗含的有关适销性、非侵权以及适用于某特定用途的保证。

索引

[K]

[L]

[M]

[N]

[P]

[Q]

[S]

[T]

[X]

[Y]

[Z]

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

[特别字符]

Printed in China