Manual de scripts y automatización Python de IBM SPSS Modeler 17

Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información del apartado "Avisos" en la página 321.

Información de producto

Esta edición se aplica a la versión 17, release 0, modificación 0 de IBM SPSS Modeler y a todos los releases y modificaciones subsiguientes hasta que se indique lo contrario en nuevas ediciones.

Contenido

Capítulo 1. Scripts y lenguaje de scripts

Conceptos básicos del uso de scripts

El procesamiento enIBM® SPSS Modeler es una herramienta potente para automatizar procesos en la interfaz de usuario. Los scripts pueden realizar los mismos tipos de acciones que se realizan con el ratón o el teclado y se utilizan para automatizar tareas que resultarían extremadamente repetitivas o llevarían mucho tiempo si se realizaran manualmente.

Puede utilizar los scripts para:

- Imponer un orden concreto para la ejecución de nodos en una ruta.
- Establecer propiedades de los nodos y realizar derivaciones usando un subconjunto de CLEM (Control Language for Expression Manipulation).
- Especificar una secuencia automática de acciones que normalmente implican la interacción del usuario (por ejemplo, puede generar un modelo y comprobarlo a continuación).
- Configurar procesos complejos que requieren una interacción sustancial del usuario, como los procedimientos de validación cruzada que requieren una repetitiva generación y comprobación de modelo.
- Configurar procesos que manipulen rutas; por ejemplo, puede tomar una ruta de entrenamiento de modelo, ejecutarla y producir la ruta de comprobación del modelo automáticamente.

Este capítulo proporciona descripciones de alto nivel y ejemplos de scripts de nivel de ruta, scripts autónomos y scripts en Supernodos en la interfaz de IBM SPSS Modeler. Para obtener más información sobre el lenguaje, la sintaxis y los comandos consulte los capítulos siguientes.

Note: no puede importar y ejecutar scripts creados en IBM SPSS Statistics de IBM SPSS Modeler.

Tipos de scripts

IBM SPSS Modeler utiliza tres tipos de scripts:

- Los **scripts de la ruta** se guardan como una propiedad de ruta y se guardan y se cargan con una ruta específica. Por ejemplo, puede escribir un script de ruta que automatice el proceso de entrenamiento y aplicación de un nugget de modelo. También puede especificar que cuando se ejecute una ruta particular, se ejecute el script, en lugar del contenido del lienzo de la ruta.
- Los **scripts autónomos** no están asociados a ninguna ruta en particular y se guardan en archivos de texto externos. Puede utilizar un script autónomo, por ejemplo, para manipular varias rutas a la vez.
- Los **scripts Supernodos** se guardan como una propiedad de ruta de supernodo. Los scripts Supernodos sólo están disponibles en supernodos terminales. Puede utilizar un script de supernodo para controlar la secuencia de ejecución del contenido del supernodo. En supernodos no terminales (origen o proceso), puede definir propiedades del supernodo o los nodos que contiene en su script de ruta directamente.

Scripts de ruta

Los scripts se pueden utilizar para personalizar operaciones dentro de una ruta particular y se guardan con esa ruta. Los scripts de la ruta se pueden utilizar para especificar un orden de ejecución particular para los nodos terminales de una ruta. El cuadro de diálogo del script de ruta se utiliza para editar el script que está guardado con la ruta actual.

Para acceder a la pestaña de scripts de ruta en el cuadro de diálogo Propiedades de ruta:

1. En el menú Herramientas, seleccione:

Propiedades de la ruta > **Ejecución**.

2. Pulse en la pestaña **Ejecución** para trabajar con scripts en la ruta actual.

Los iconos de la barra de herramientas de la parte superior del cuadro de diálogo del script de ruta le permiten realizar las siguientes operaciones:

- Importar el contenido de un script actónomo preexistente a la ventana.
- Guardar un script como archivo de texto.
- Imprimir un script.
- Añadir script predeterminado.
- Editar un script (deshacer, cortar, copiar, pegar y otras funciones de edición comunes).
- Ejecutar el script completo actual.
- Ejecutar líneas concretas de un script.
- Detener un script durante la ejecución. (Este icono sólo está habilitado cuando un script se está ejecutando).
- Comprobar la sintaxis del script y, si se detectan errores, abrirlos en el panel inferior del cuadro de diálogo para su revisión.

A partir de la versión 16.0, SPSS Modeler utiliza el lenguaje de scripts Python. Todas las versiones anteriores utilizaban un lenguaje de script exclusivo de SPSS Modeler, al que ahora se denomina Scripts de herencia. Según el tipo de script con el que trabaje, en la pestaña **Ejecución** seleccione la modalidad de ejecución **Predeterminada (script opcional)** y, a continuación, seleccione **Python** o **Legacy**.

Además, puede especificar si este script se debe ejecutar o no cuando se ejecuta la ruta. Puede seleccionar **Ejecutar este script** para que se ejecute cada vez que se ejecute la ruta y se use el orden de ejecución especificado en el script. De este modo se proporciona una automatización a nivel de ruta para acelerar la generación del modelo. Sin embargo, la configuración predeterminada es omitir el script durante la ejecución de la ruta. Incluso si selecciona la opción **Omitir este script**, siempre puede ejecutar la ruta directamente desde este cuadro de diálogo.

El editor de scripts incluye las siguientes características que ayudan a crear scripts:

- Resaltado de sintaxis. Se resaltan las palabras claves, los valores literales (tales como cadenas y números) y los comentarios.
- Numeración de líneas.
- Coincidencia de bloques. Cuando se coloca el cursor al inicio de un bloque de programa, también se resalta el bloque final correspondiente.
- Finalización automática sugerida.

Los colores y los estilos de texto que utiliza el resaltado de sintaxis se pueden personalizar utilizando las preferencias de visualización de IBM SPSS Modeler. Puede acceder a las preferencias de visualización seleccionando **Herramientas** > **Opciones** > **Opciones de usuario** y pulsando la pestaña **Sintaxis**.

Se puede acceder a una lista de finalizaciones de sintaxis sugeridas seleccionando la **Sugerencia automática** en el menú de contexto o pulsando Ctrl más espacio. Utilice las teclas de cursor para desplazarse hacia arriba y hacia abajo por la lista y, a continuación, pulse Intro para insertar el texto seleccionado. Pulse Esc para salir de la modalidad de sugerencia automática sin modificar el texto existente.

La pestaña **Depurar** muestra mensajes de depuración y puede utilizarse para evaluar el estado del script una vez ejecutado el script. La pestaña **Depurar** consta de un área de texto de sólo lectura y un campo de texto de entrada de una sola línea. El área de texto muestra el texto que se envía a la salida estándar o un error estándar mediante los scripts, por ejemplo, a través del texto del mensaje de error. El campo de texto de entrada toma la entrada del usuario. Esta entrada se evalúa dentro del contexto del script que se

ha ejecutado más recientemente en el diálogo (conocido como el *contexto de los scripts*). El área de texto contiene el mandato y la salida resultante, de modo que el usuario puede ver un rastreo de los comandos. El campo de texto de entrada siempre contiene el indicador de comandos (`-->` para scripts de herencia).

Un contexto de script nuevo se crea en las circunstancias siguientes:

- Se ejecuta un script utilizando el botón "Ejecutar este script" o el botón "Ejecutar líneas seleccionadas".
- Se modifica el lenguaje de script.

Si se crea un nuevo contexto de script, el área de texto se borra.

Nota: Si se ejecuta una ruta fuera del panel de scripts no se modificará el contexto del script contenido en el panel de scripts. Los valores de cualquier variable creada como parte de dicha ejecución no se podrán ver en el diálogo del script.

Ejemplo de script de ruta: Entrenamiento de una red neuronal

Una ruta se puede usar para entrenar un modelo de red neuronal cuando se ejecute. Normalmente, para comprobar el modelo, se inserta el nodo de modelado para agregar el modelo a la ruta, realizar las conexiones adecuadas y ejecutar el nodo Análisis.

Mediante un script de IBM SPSS Modeler se puede automatizar el proceso de comprobar el nugget de modelo tras crearlo. Por ejemplo, el siguiente script de ruta para la ruta de demostración *druglearn.str* (disponible en la carpeta */Demos/streams/* de su instalación de IBM SPSS Modeler) se puede ejecutar desde el cuadro de diálogo de propiedades de ruta (**Herramientas > Propiedades de ruta > Ruta**):

```
ruta = 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([])
```

Los puntos siguientes describen cada línea de este ejemplo de script.

- La primera línea define una variable que apunta a la ruta actual.
- En la línea 2, el script busca el nodo generador Red neuronal.
- En la línea 3, el script crea una lista donde los resultados de la ejecución se pueden almacenar.
- En la línea 4, se crea el nugget de modelo Red neuronal. Se almacena en la lista definida en línea 3.
- En la línea 5, se crea un nodo de aplicación de modelo para el nugget de modelo y se coloca en el lienzo de rutas.
- En la línea 6, se crea un nodo de análisis denominado `Drug`.
- En la línea 7, el script busca el nodo Type.
- En la línea 8, el script conecta el nodo de aplicación de modelo creado en la línea 5 entre el nodo Type y el nodo Analysis.
- Finalmente, el nodo Análisis se ejecuta para producir el informe Análisis.

Es posible utilizar un script para crear y ejecutar una ruta desde cero, comenzando con un lienzo vacío. Para obtener más información sobre el lenguaje de scripts en general, consulte Conceptos básicos del lenguaje de scripts.

Scripts autónomos

El cuadro de diálogo script autónomo se usa para crear o editar un script que se ha guardado como archivo de texto. En él se muestra el nombre del archivo y se proporcionan recursos para la carga, almacenamiento, importación y ejecución de scripts.

Para acceder al cuadro de diálogo del script autónomo:

En el menú principal, elija:

Herramientas > **Script autónomo**

Los scripts autónomos y los de ruta comparten las mismas opciones de comprobación de sintaxis de scripts y barra de herramientas. Consulte el tema "Scripts de ruta" en la página 1 para obtener más información.

Ejemplo de script autónomo: Guardar y cargar un modelo

Los scripts autónomos son útiles para la manipulación de rutas. Presuponga que tiene dos rutas, una que crea un modelo y otra que usa diagramas para examinar el conjunto de reglas generado a partir de la primera ruta con campos de datos existentes. Un script autónomo para este escenario tendría un aspecto similar a éste:

```
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/16/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([])
```

Nota: Para obtener más información sobre el lenguaje de scripts en general, consulte Conceptos básicos del lenguaje de scripts.

Ejemplo de script autónomo: Generar un modelo de selección de características

Comenzando con un lienzo vacío, este ejemplo crea una ruta que genera un modelo de selección de características, aplica el modelo y crea una tabla que indica los 15 campos más importantes respecto al objetivo especificado.

```
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([])
```

El script crea un nodo de origen para leer en los datos, utiliza un nodo Tipo para definir el rol del campo `response_01` hacia `Destino` y, a continuación, crea y ejecuta un nodo Selección de características. Este script también conecta cada nodo y posiciones en el lienzo de la ruta para producir un diseño legible. El nugget de modelo resultante se conecta al nodo Tabla, que indica los 15 campos más importantes, tal y como determinan las propiedades `selection_mode` y `top_n`. Consulte el tema “Propiedades de featureselectionnode” en la página 187 para obtener más información.

Scripts de Supernodo

Puede crear y guardar scripts con cualquier supernodo terminal utilizando el lenguaje de scripts de IBM SPSS Modeler. Estos scripts sólo están disponibles para supernodos terminales y se suelen utilizar cuando crea rutas de plantilla o para imponer un orden de ejecución especial del contenido del supernodo. Los scripts de supernodo también permiten ejecutar más de un script en una ruta.

Por ejemplo, supongamos que necesita especificar el orden de ejecución de una ruta compleja y su supernodo contiene varios nodos, incluyendo un nodo Val. globales, que se debe ejecutar antes de derivar un nuevo campo utilizado en un nodo Gráfico. En este caso, puede crear un script de supernodo que ejecute el nodo Val. globales en primer lugar. Los valores calculados por este nodo, como la media o la desviación estándar, se pueden usar posteriormente cuando se ejecute el nodo Gráfico.

En un script de Supernodo, puede especificar las propiedades del nodo de la misma manera que otros scripts. También puede cambiar y definir las propiedades de cualquier supernodo o sus nodos encapsulados directamente desde un script de ruta. Consulte el tema Capítulo 19, “Propiedades de Supernodos”, en la página 305 para obtener más información. Este método funciona para supernodos de origen y proceso y supernodos terminales.

Note: como sólo los supernodos terminales pueden ejecutar sus propios scripts, la pestaña Scripts del cuadro de diálogo Supernodo sólo está disponible para supernodos terminales.

Para abrir el cuadro de diálogo de script de supernodo desde el lienzo principal:

Seleccione un supernodo terminal en el lienzo de rutas y, en el menú de supernodo, seleccione:

Script de Supernodo...

Para abrir el cuadro de diálogo de script de supernodo desde el lienzo de supernodo aumentado:

Pulse con el botón derecho en el lienzo de supernodo y en el menú contextual seleccione:

Script de Supernodo...

Script de supernodo de ejemplo

El siguiente script de supernodo establece el orden en que se ejecutarán los nodos terminales del supernodo. Este orden garantiza que el nodo Val. globales se ejecuta primero para que los valores que calcula este nodo se puedan utilizar cuando se ejecute otro nodo.

```
execute 'Val. globales'
execute 'gains'
execute 'profit'
execute 'age v. $CC-pep'
execute 'Table'
```

Creación de bucles y ejecución condicional en rutas

A partir de la versión 16.0, SPSS Modeler permite crear scripts básicos desde dentro de una ruta seleccionando valores en varios cuadros de diálogo en lugar de tener que escribir instrucciones directamente en lenguaje de script. Los dos principales tipos de script que pueden crearse de este modo son los bucles sencillos y un modo de ejecutar nodos si se cumple una condición.

En una misma ruta pueden combinarse bucles y reglas de ejecución condicional. Por ejemplo, suponga que tiene datos relativos a ventas de vehículos de fabricantes de todo el mundo. Podría crearse un bucle para procesar los datos en una ruta, identificando los detalles por país del fabricante, y sacar los datos a distintas gráficas para mostrar detalles tales como volumen de ventas por modelo, niveles de emisión por fabricante y cilindrada, etc. Si solo le interesara analizar la información procedente de Europa, también podría añadir condiciones al bucle que impidieran la creación de gráficas de fabricantes procedentes de América y Asia.

Nota: Puesto que tanto un bucle como una ejecución condicional están basados en scripts de segundo plano, solo se aplican a una ruta entera cuando se ejecuta.

- **Bucles** Los bucles pueden utilizarse para automatizar tareas repetitivas. Por ejemplo, esto podría suponer añadir un determinado número de nodos a una ruta y modificar un parámetro del nodo cada vez. De forma opcional, podría controlarse la ejecución de una ruta o rama varias veces, como en los ejemplos siguientes:
  - Ejecutar la ruta un determinado número de veces y cambiar el origen cada vez.
  - Ejecutar la ruta un determinado número de veces cambiando el valor de una variable cada vez.
  - Ejecutar la ruta un determinado número de veces especificando un campo adicional en cada ejecución.
  - Construir un modelo un determinado número de veces y cambiar la configuración del modelo cada vez.
- **Ejecución condicional** Puede utilizarse para controlar cómo ejecutan los nodos en función de condiciones definidas previamente como, por ejemplo:
  - Dependiendo de si un determinado valor es verdadero o falso, se controla la ejecución de un nodo.
  - Definir si la iteración de nodos se ejecutará en paralelo o de forma secuencial.

Tanto bucles ejecuciones condicionales se configuran en la pestaña Ejecución dentro del cuadro de diálogo Propiedades de ruta. Los nodos que se utilicen en bucles o de forma condicional aparecerán con un símbolo adicional en el lienzo de rutas para indicar que forman parte de una ejecución por bucles o condicional.

Puede accederse a la pestaña Ejecución de tres maneras:

- Mediante los menús de la parte superior del cuadro de diálogo principal:
  1. En el menú Herramientas, seleccione:

     Propiedades de la ruta > **Ejecución**.
  2. Pulse en la pestaña Ejecución para trabajar con los scripts de la ruta actual.
- Dentro de una ruta:
  1. Pulse con el botón derecho en un nodo y seleccione **Bucles/Ejecución condicional**.
  2. Seleccione la opción de submenú que corresponda.
- En la barra de herramientas gráfica de la parte superior del cuadro de diálogo principal, pulse en el icono de propiedades de ruta.

Si es la primera vez que configura los detalles de un bucle o de una ejecución condicional, en la pestaña Ejecución seleccione el modo de ejecución **Ejecución de bucles/condicional** y después seleccione la subpestaña **Condicional** o **Bucles**.

Bucles en rutas

Con la creación de bucles puede automatizar las tareas repetitivas en las rutas, por ejemplo:

- Ejecutar la ruta un determinado número de veces y cambiar el origen cada vez.
- Ejecutar la ruta un determinado número de veces cambiando el valor de una variable cada vez.
- Ejecutar la ruta un determinado número de veces especificando un campo adicional en cada ejecución.
- Construir un modelo un determinado número de veces y cambiar la configuración del modelo cada vez.

Configurar las condiciones que deben cumplirse en la subpestaña **Bucle** de la pestaña Ejecución de la ruta. Para visualizar la subpestaña, seleccione el modo de ejecución **Ejecución en bucle/condicional**.

Los requisitos de bucle que defina entrarán en vigor cuando se ejecute la ruta, si se ha establecido la modalidad de ejecución **Ejecución en bucle/condicional**. De forma opcional, puede generar el código de script para los requisitos de bucle y pegarlo en el editor de scripts pulsando **Pegar...** en el ángulo inferior derecho de la subpestaña Bucle y la visualización de la pestaña Ejecución principal cambiará para mostrar la modalidad de ejecución **Predeterminada (script opcional)** con el script en la parte superior de la pestaña. Esto significa que puede definir bucles utilizando las diferentes opciones del cuadro de diálogo de bucle antes de generar un script que puede personalizar adicionalmente en el editor de scripts. Tenga en cuenta que cuando pulsa **Pegar...** los requisitos de bucle que ha definido también se mostrarán en el script generado.

Importante: Las variables de bucle que establezca en una ruta de SPSS Modeler pueden sustituirse si se ejecuta la corriente en un trabajo IBM SPSS Collaboration and Deployment Services. Esto se debe a que la entrada del editor del trabajo IBM SPSS Collaboration and Deployment Services sobrescribe la entrada de SPSS Modeler. Por ejemplo, si se establece una variable de bucle en la ruta para crear un nombre de archivo de salida diferente para cada bucle, los archivos se especifican correctamente en SPSS Modeler, pero son sustituidos por la entrada fija especificada en la pestaña Resultado del Gestor de despliegue de IBM SPSS Collaboration and Deployment Services.

Para configurar un bucle

1. Cree una clave de iteración para definir la estructura principal del bucle principal que se creará en una ruta. Consulte el tema Crear una clave de iteración para obtener más información.

2. Si es necesario, defina una o varias variables de iteración. Consulte el tema Crear una variable de iteración para obtener más información.
3. Las iteraciones y las variables que cree se muestran en el cuerpo principal de la subpestaña. De forma predeterminada, las iteraciones se ejecutan en el orden en que aparecen. Para subir o bajar una iteración en la lista, pulse la iteración para seleccionarla y, a continuación, utilice la flecha arriba o la flecha abajo de la columna de la derecha de la subpestaña para cambiar el orden.

Creación de una clave de iteración para bucles de rutas

Utilice una clave de iteración para definir la estructura principal del bucle principal que se creará en una ruta. Por ejemplo, si está analizando las ventas de automóviles, puede crear un parámetro de ruta *País de fabricación* y utilizarlo como la clave de la iteración. Cuando se ejecute la ruta, esta clave se establece en cada valor de país diferente de sus datos durante cada iteración. Utilice el cuadro de diálogo Definir clave de iteración para configurar la clave.

Para abrir el cuadro de diálogo, seleccione el botón **Clave de iteración...** en el ángulo inferior izquierdo de la subpestaña Bucle o pulse con el derecho cualquier nodo de la ruta y seleccione **Ejecución en bucle/condicional** > **Definir clave de iteración (campos)** o **Ejecución en bucle/condicional** > **Definir clave de iteración (valores)**. Si abre el cuadro de diálogo desde la ruta, algunos campos se completan automáticamente, tales como el nombre del nodo.

Para configurar una clave de iteración, complete los campos siguientes:

Iterar en. Puede seleccionar entre una de las opciones siguientes:

- **Parámetro de ruta - Campos**. Utilice esta opción para crear un bucle que establezca el valor de un parámetro de ruta existente en cada campo especificado de forma ordenada.
- **Parámetro de ruta - Valores**. Utilice esta opción para crear un bucle que establezca el valor de un parámetro de ruta existente en cada valor especificado de forma ordenada.
- **Propiedad del nodo - Campos**. Utilice esta opción para crear un bucle que establezca el valor de una propiedad de nodo en cada campo especificado de forma ordenada.
- **Propiedad del nodo - Valores**. Utilice esta opción para crear un bucle que establezca el valor de una propiedad de nodo en cada valor especificado de forma ordenada.

Qué se ha de establecer. Elija el elemento cuyo valor se establecerá cada vez que se ejecute el bucle. Puede seleccionar entre una de las opciones siguientes:

- **Parámetro**. Solo está disponible si se selecciona **Parámetro de ruta - Campos** o **Parámetro de ruta - Valores**. Seleccione el parámetro necesario en la lista disponible.
- **Nodo**. Solo está disponible si se selecciona **Propiedad del nodo - Campos** o **Propiedad del nodo - Valores**. Seleccione el nodo para el que desee configurar un bucle. Pulse el botón Examinar para abrir el diálogo Seleccionar nodo y elija el nodo que desee. Si hay demasiados nodos en la lista, puede filtrar la visualización para que únicamente se muestren los nodos de una de las siguientes categorías: Origen, Proceso, Gráfico, Modelado, Resultados o Aplicar modelo.
- **Propiedad**. Solo está disponible si se selecciona **Propiedad del nodo - Campos** o **Propiedad del nodo - Valores**. Seleccione la propiedad del nodo en la lista disponible.

Campos de uso. Solo está disponible si se selecciona **Parámetro de ruta - Campos** o **Propiedad del nodo - Campos**. Seleccione el campo o los campos de un nodo que se utilizarán para proporcionar los valores de iteración. Puede seleccionar entre una de las opciones siguientes:

- **Nodo**. Solo está disponible si se selecciona **Parámetro de ruta - Campos**. Seleccione el nodo que contiene los detalles para los que desea configurar un bucle. Pulse el botón Examinar para abrir el diálogo Seleccionar nodo y elija el nodo que desee. Si hay demasiados nodos en la lista, puede filtrar la visualización para que únicamente se muestren los nodos de una de las siguientes categorías: Origen, Proceso, Gráfico, Modelado, Resultados o Aplicar modelo.

- **Lista de campos**. Pulse el botón de lista de la columna derecha para visualizar el cuadro de diálogo Seleccionar campos, donde puede seleccionar los campos del nodo para proporcionar los datos de iteración. Para obtener más información, consulte "Selección de campos en iteraciones" en la página 10.

Valores de uso. Solo está disponible si se selecciona **Parámetro de ruta - Valores** o **Propiedad del nodo - Valores**. En el campo seleccionado, seleccione el o los valores que se utilizarán como valores de iteración. Puede seleccionar entre una de las opciones siguientes:

- **Nodo**. Solo está disponible si se selecciona **Parámetro de ruta - Valores**. Seleccione el nodo que contiene los detalles para los que desea configurar un bucle. Pulse el botón Examinar para abrir el diálogo Seleccionar nodo y elija el nodo que desee. Si hay demasiados nodos en la lista, puede filtrar la visualización para que únicamente se muestren los nodos de una de las siguientes categorías: Origen, Proceso, Gráfico, Modelado, Resultados o Aplicar modelo.
- **Lista de campos**. Seleccione el campo del nodo para proporcionar los datos de iteración.
- **Lista de valores**. Pulse el botón de lista de la columna derecha para visualizar el cuadro de diálogo Seleccionar valores, donde puede seleccionar los valores del campo para proporcionar los datos de iteración.

Creación de una variable de iteración para bucles de rutas

Puede utilizar variables de iteración para cambiar los valores de los parámetros o las propiedades de ruta de los nodos seleccionados en una ruta, cada vez que se ejecute un bucle. Por ejemplo, si el bucle de ruta está analizando los datos de ventas de automóviles y utiliza *País de fabricación* como clave de iteración, puede tener un gráfico de resultados que muestre las ventas por modelo y otro gráfico de resultados que muestre información sobre emisiones contaminantes. En estos casos puede crear variables de iteración que creen nuevos títulos para los gráficos resultantes, tales como *Emisiones de vehículos suecos* y *Ventas de automóviles japoneses por modelo*. Utilice el cuadro de diálogo Definir variable de iteración para configurar las variables que necesite.

Para abrir el cuadro de diálogo, seleccione el botón **Añadir variable...** en el ángulo superior izquierdo de la subpestaña Bucle, o pulse con el botón derecho cualquier nodo de la ruta y seleccione **Ejecución en bucle/condicional** > **Definir variable de iteración**.

Para configurar una variable de iteración, complete los campos siguientes:

Cambiar. Seleccione el tipo de atributo que desea enmendar. Puede elegir **Parámetro de ruta** o **Propiedad del nodo**.

- Si selecciona **Parámetro de ruta**, elija el parámetro necesario y, a continuación, utilizando una de las opciones siguientes, si están disponibles en su ruta, defina cómo se debe establecer el valor de dicho parámetro con cada iteración del bucle:
  - **Variable global**. Seleccione la variable global en la que se debe establecer el parámetro de ruta.
  - **Casilla de resultados de tabla**. Para que un parámetro de ruta sea el valor de una casilla de resultados de tabla, seleccione la tabla en la lista y especifique la **Fila** y la **Columna** que se han de utilizar.
  - **Especificar manualmente**. Seleccione esta opción si desea especificar manualmente el valor que tomará este parámetro en cada iteración. Cuando regrese a la subpestaña Bucle se habrá creado una columna nueva en la que puede especificar el texto necesario.
- Si selecciona **Propiedad del nodo**, elija el nodo necesario y una de sus propiedades, a continuación, establezca el valor que desea que se utilice para dicha propiedad. Establezca el nuevo valor de la propiedad utilizando una de las opciones siguientes:
  - **Solo**. El valor de la propiedad utilizará el valor de la clave de iteración. Para obtener más información, consulte "Creación de una clave de iteración para bucles de rutas" en la página 8.
  - **Como prefijo de tallo**. Utiliza el valor de la clave de iteración como un prefijo para lo que especifique en el campo **Tallo**.

– **Como sufijo de tallo**. Utiliza el valor de la clave de iteración como un sufijo para lo que especifique en el campo **Tallo**.

Si selecciona la opción de prefijo o de sufijo se le solicitará que añada el texto adicional en el campo **Tallo**. Por ejemplo, si el valor de la clave de iteración es *País de fabricación* y selecciona **Como prefijo de tallo**, puede entrar - *ventas por modelo* en este campo.

Selección de campos en iteraciones

Cuando se crean iteraciones pueden seleccionarse uno o más campos mediante el cuadro de diálogo Seleccionar campos.

Ordenar por: puede ordenar campos disponibles para su visualización eligiendo una de las siguientes opciones:

- **Natural**: el orden de los campos es aquél en que pasaron desde la parte anterior de la ruta de datos al nodo actual.
- **Nombre**: ordena los campos siguiendo un orden alfabético para su visualización.
- **Tipo**: ordena los campos en función de su nivel de medición. Esta opción es útil cuando se seleccionan campos con un nivel de medición en particular.

Seleccione los campos de la lista de uno en uno o mantenga pulsada la tecla Mayús o Ctrl mientras selecciona otros campos para seleccionar varios campos. También puede utilizar los botones que se muestran bajo la lista para seleccionar grupos de campos en función de su nivel de medición o seleccionar y anular la selección de todos los campos de la tabla.

Tenga en cuenta que los campos disponibles para su selección se filtran para mostrar solo los campos que son adecuados para el parámetro de ruta o la propiedad de nodo que se están utilizando. Por ejemplo, si se está utilizando un parámetro de ruta que tiene un tipo de almacenamiento de cadena, solo se mostrarán los campos que tengan un tipo de almacenamiento de cadena.

Ejecución condicional en rutas

Con la ejecución condicional puede controlar cómo se ejecutan los nodos terminales, en función de las condiciones de coincidencia de contenido de ruta que defina. Ejemplos de ello pueden ser los siguientes:

- Dependiendo de si un determinado valor es verdadero o falso, se controla la ejecución de un nodo.
- Definir si la iteración de nodos se ejecutará en paralelo o de forma secuencial.

Configurar las condiciones que deben cumplirse en la subpestaña **Condicional** de la pestaña Ejecución de la ruta. Para visualizar la subpestaña, seleccione el modo de ejecución **Ejecución en bucle/condicional**.

Los requisitos de ejecución condicional que defina entrarán en vigor cuando se ejecute la ruta, si se ha establecido la modalidad de ejecución **Ejecución en bucle/condicional**. De forma opcional, puede generar el código de script para los requisitos de ejecución condicional y pegarlo en el editor de scripts pulsando **Pegar...** en el ángulo inferior derecho de la subpestaña Condicional; la visualización de la pestaña Ejecución principal cambiará para mostrar la modalidad de ejecución **Predeterminada (script opcional)** con el script en la parte superior de la pestaña. Esto significa que puede definir condiciones utilizando las diferentes opciones del cuadro de diálogo de bucle antes de generar un script que puede personalizar adicionalmente en el editor de scripts. Tenga en cuenta que cuando pulsa **Pegar...** los requisitos de bucle que ha definido también se mostrarán en el script generado.

Para configurar una condición:

1. En la columna de la derecha de la subpestaña Condicional, pulse el botón Añadir condición nueva  para abrir el cuadro de diálogo Añadir sentencia de ejecución condicional. En este diálogo especifica la condición que se debe cumplir para que se ejecute el nodo.
2. En el cuadro de diálogo Sentencia de ejecución condicional, especifique lo siguiente:

a. **Nodo**. Seleccione el nodo para el que desee configurar una ejecución condicional. Pulse el botón Examinar para abrir el diálogo Seleccionar nodo y elija el nodo que desee. Si hay demasiados nodos en la lista, puede filtrar la visualización para que únicamente se muestren los nodos de una de las siguientes categorías: Exportar, Gráfico, Modelado o Resultados.

b. **Condición basada en**. Especifique la condición que se debe cumplir para que se ejecute el nodo. Puede elegir una de estas cuatro opciones: **Parámetro de ruta**, **Variable global**, **Casilla de resultados de tabla** o **Siempre verdadero**. Los detalles que especifique en la mitad inferior del cuadro de diálogo están controlados por la condición que elija.

   - **Parámetro de ruta**. Seleccione el parámetro de la lista disponible y, a continuación, seleccione el **Operador** para ese parámetro; por ejemplo, el operador puede ser Más, Igual, Menor que, Entre, etc. A continuación especifique el **Valor**, o los valores mínimos o máximos, dependiendo del operador seleccionado.
   - **Variable global**. Seleccione la variable de la lista disponible; por ejemplo, esto podría incluir: Media, Suma, Valor mínimo, Valor máximo o Desviación estándar. A continuación, seleccione **Operador** y los valores necesarios.
   - **Casilla de resultados de tabla**. Seleccione el nodo de tabla de la lista disponible y, a continuación, seleccione la **Fila** y la **Columna** en la tabla. A continuación, seleccione **Operador** y los valores necesarios.
   - **Siempre verdadero**. Seleccione esta opción si siempre se ha de ejecutar el nodo. Si selecciona esta opción, no hay parámetros adicionales que seleccionar.

3. Repita los pasos 1 y 2 tantas veces como sea necesario hasta que haya configurado todas las condiciones que requiere. El nodo que ha seleccionado y la condición que se debe cumplir antes de que se ejecute el nodo se muestran en el cuerpo principal de la subpestaña de las columnas **Ejecutar nodo** y **Si esta condición es verdadera** respectivamente.
4. De forma predeterminada, los nodos y las condiciones se ejecutan en el orden en que aparecen. Para subir o bajar un nodo y condición en la lista, pulse el nodo para seleccionarlo y, a continuación, utilice la flecha arriba o la flecha abajo en la columna de la derecha de la subpestaña para cambiar el orden.

Además, puede establecer las siguientes opciones en la parte inferior de la subpestaña Condicional:

- **Evaluar todo en orden**. Seleccione esta opción para evaluar cada condición en el orden en que se muestra en la subpestaña. Los nodos para los que se han encontrado condiciones que son "True" se ejecutarán una vez evaluadas todas las condiciones.
- **Ejecutar uno por uno**. Sólo está disponible si se selecciona **Evaluar todo en orden**. Si se selecciona significa que si la condición se evalúa como "True", el nodo asociado con esa condición se ejecutará antes de que se evalúe la siguiente condición.
- **Evaluar hasta primer acierto**. Si se selecciona, significa que solo se ejecutará el primer nodo cuya evaluación de las condiciones devuelva el valor "True".

Ejecución e interrupción de scripts

Existen diversas formas de ejecutar scripts. Por ejemplo, en el script de ruta o en el cuadro de diálogo del script, el botón "Ejecutar este script" ejecuta el script completo:

Figura 1. Botón Ejecutar este script

El botón "Ejecutar líneas seleccionadas" ejecuta una única línea, o un bloque de líneas adyacentes, que ha seleccionado en el script:

Figura 2. Botón Ejecutar líneas seleccionadas

Un script se puede ejecutar mediante cualquiera de los siguientes métodos:

- Pulse en el botón "Ejecutar este script" o "Ejecutar líneas seleccionadas" dentro de un script de ruta o un cuadro de diálogo de script.
- Ejecutando una ruta donde **Ejecutar este script** esté establecido como el método de ejecución predeterminado.
- Utilizando la marca `-execute` al inicio en modo interactivo. Consulte el tema "Uso de argumentos en la línea de comandos" en la página 61 para obtener más información.

Note: cuando se ejecuta el Supernodo, se ejecuta un script de Supernodo siempre que se haya seleccionado **Ejecutar este script** en el cuadro de diálogo del script de Supernodo.

Interrupción de la ejecución del script

En el cuadro de diálogo de scripts de ruta, se activará el botón rojo de detención durante la ejecución de scripts. Pulsando este botón, puede abandonar la ejecución del script y de cualquier ruta actual.

Buscar y reemplazar

El cuadro de diálogo Buscar/reemplazar está disponible en lugares donde edita texto de script o de expresión, incluido el editor de scripts, el generador de expresiones CLEM o cuando define una plantilla en el nodo Informe. Cuando edite texto en cualquiera de estas áreas, pulse `Ctrl` para acceder al cuadro de diálogo, asegurándose de que el cursor está centrado en un área de texto. Por ejemplo, si trabaja en un nodo Rellenar, puede acceder al cuadro de diálogo desde cualquiera de las áreas de texto de la pestaña Configuración o desde el campo de texto del generador de expresiones.

1. Con el cursor en un área de texto, pulse Ctrl+F para acceder al cuadro de diálogo Buscar/reemplazar.
2. Introduzca el texto que desee buscar o selecciónelo de la lista desplegable de elementos buscados recientemente.
3. Introduzca el texto de reemplazo, si lo tiene.
4. Pulse en **Buscar siguiente** para iniciar la búsqueda.
5. Pulse en **Reemplazar** para reemplazar la sección actual o en **Reemplazar todos** para actualizar todas las instancias o sólo las seleccionadas.
6. El cuadro de diálogo se cierra después de cada operación. Pulse F3 desde cualquier área de texto para repetir la operación de búsqueda más reciente o pulse Ctrl+F para volver a acceder al cuadro de diálogo.

Opciones de búsqueda

Coincidir mayúsculas y minúsculas. Especifica si la operación de búsqueda hace distinción entre mayúsculas y minúsculas; por ejemplo, si *mivar* es igual que *miVar*. El texto de reemplazo siempre se introduce exactamente como se ha introducido, independientemente de este ajuste.

Sólo palabras completas. Especifica si la operación de búsqueda tiene en cuenta el texto incluido dentro de las palabras. Por ejemplo, si se selecciona, la búsqueda de *fuego* no será igual que *cortafuegos* ni que *corta-fuegos*.

Expresiones regulares. Especifica si se utiliza la sintaxis de expresiones regulares (consulte la sección siguiente). Si está seleccionado, la opción **Sólo palabras completas** está desactivada y su valor se ignora.

Sólo texto seleccionado. Controla el ámbito de la búsqueda al utilizar la opción **Reemplazar todos**.

Sintaxis de expresiones regulares

Las expresiones regulares le permiten buscar caracteres especiales como caracteres de tabulador o de nueva línea, clases o rangos de caracteres como de la *a* a la *d*, cualquier dígito o no dígito y límites como el principio o el final de una línea. Se admiten los siguientes tipos de expresiones.

Tabla 1. Coincidencias de caracteres.

Caracteres	**Coincidencias**
x	El carácter x
\\	El carácter de barra inclinada invertida
\0n	El carácter con valor octal 0n (0 <= n <= 7)
\0nn	El carácter con valor octal 0nn (0 <= n <= 7)
\0mnn	El carácter con valor octal 0mnn (0 <= m <= 3, 0 <= n <= 7)
\xhh	El carácter con valor hexadecimal 0xhh
\uhhhh	El carácter con valor hexadecimal 0xhhhh
\t	El carácter de tabulador ('\u0009')
\n	El carácter de nueva línea (avance de línea) ('\u000A')
\r	El carácter de retorno de carro ('\u000D')
\f	El carácter de avance de página ('\u000C')
\a	El carácter de alerta (campana) ('\u0007')
\e	El carácter de escape ('\u001B')
\cx	El carácter de control correspondiente a x

Tabla 2. Clases de caracteres coincidentes.

Clases de caracteres	**Coincidencias**
[abc]	a, b o c (clase simple)
[^abc]	Cualquier carácter excepto a, b o c (resta)
[a-zA-Z]	De la a a la z o de la A a la Z, ambas inclusive (rango)
[a-d[m-p]]	De la a a la d o de la m a la p (unión). Esto también puede especificarse como [a-dm-p].
[a-z&&[def]]	De la a a la z y d, e o f (intersección)
[a-z&&[^bc]]	De la a a la z, excepto b y c (resta). Esto también puede especificarse como [ad-z].
[a-z&&[^m-p]]	De la a a la z y no de la m a la p (resta). Esto también puede especificarse como [a-lq-z].

Tabla 3. Clases de caracteres predefinidas.

Clases de caracteres predefinidas	**Coincidencias**
.	Cualquier carácter (puede o no coincidir con los terminadores de línea)
\d	Cualquier dígito: [0-9]
\D	Un no dígito: [^0-9]
\s	Un carácter de espacio en blanco: [\t\n\x0B\f\r]
\S	Un carácter de espacio en blanco: [^\s]

Tabla 3. Clases de caracteres predefinidas (continuación).

Clases de caracteres predefinidas	Coincidencias
\w	Un carácter de palabra: [a-zA-Z_0-9]
\W	Un carácter que no sea de palabra: [^\w]

Tabla 4. Coincidencias de límite.

Reconocedores de límite	Coincidencias
^	El comienzo de una línea
$	El final de una línea
\b	Un límite alfabético
\B	Un límite no alfabético
\A	El comienzo de la entrada
\Z	El final de la entrada pero para el terminador final, si lo hay
\z	El final de la entrada

Capítulo 2. Lenguage de scripts

Conceptos básicos del lenguaje de scripts

El recurso de scripts para IBM SPSS Modeler le permite crear scripts que funcionen en la interfaz de usuario de SPSS Modeler, manipular los objetos de salida y ejecutar la sintaxis de comandos. Puede ejecutar scripts directamente desde SPSS Modeler.

Los scripts de IBM SPSS Modeler están escritos en el lenguaje de script Python. La implementación Java de Python que IBM SPSS Modeler se denomina Jython. El lenguaje de script consta de las siguientes características:

- Un formato para hacer referencia a nodos, rutas, proyectos, resultados y otros objetos de IBM SPSS Modeler.
- Un conjunto de instrucciones o comandos de scripts que se puede utilizar para manipular tales objetos.
- Un lenguaje de expresión de script para establecer los valores de las variables, los parámetros y otros objetos.
- Compatibilidad con comentarios, continuaciones y bloques de texto literal.

Las secciones siguientes describen el lenguaje de scripts Python, la implementación de Jython por parte de Python y la sintaxis básica para empezar a crear scripts en IBM SPSS Modeler. Las secciones siguientes recogen información sobre comandos y propiedades específicas.

Python y Jython

Jython es una implementación del lenguaje de scripts Python, escrito en el lenguaje Java e integrado con la plataforma Java. Python es un potente lenguaje de script orientado a objetos. Jython es útil porque proporciona las características de productividad de un lenguaje de script maduro y, a diferencia de Python, se ejecuta en cualquier entorno que soporte una máquina virtual Java (JVM). Esto significa que las bibliotecas Java de la máquina virtual Java están disponibles para utilizarlas cuando se escriben programas. Con Jython, puede beneficiarse de esta diferencia y utilizar la sintaxis y la mayoría de las características del lenguaje Python.

Como lenguaje de script, Python (y su implementación Jython) es fácil de aprender y ofrece una codificación eficaz con la estructura mínima necesaria para crear un programa de ejecución. El código se puede entrar de forma interactiva, es decir, una línea cada vez. Python es un lenguaje de script interpretado; no hay ningún paso de precompilación, tal como existe en Java. Los programas de Python simplemente son archivos de texto que se interpretan a medida que se entran (después de analizar los errores de sintaxis). Las expresiones simples, tales como los valores definidos, y también las acciones más complejas, tales como las definiciones de función, se ejecutan y están disponibles para su uso de forma inmediata. Los cambios realizados en el código se pueden probar fácilmente. Sin embargo, la interpretación del script tiene algunas desventajas. Por ejemplo, utilizar una variable no definida no es un error del compilador, por lo tanto solo se detecta si (y cuando) se ejecuta la sentencia en la que se utiliza la variable. En este caso, se puede editar y ejecutar el programa para depurar el error.

Python lo ve todo como un objeto, incluidos todos los datos y el código. Por lo tanto, puede manipular estos objetos con líneas de código. Algunos tipos de selección, tales como los números y cadenas, se consideran valores y no objetos, lo cual resulta más práctico, y Python da soporte a todo ello. Se da soporte a un valor nulo. Este valor nulo tiene el nombre reservado de `None`.

Para obtener una introducción más detallada del lenguaje de script Python y Jython y algunos scripts de ejemplo, consulte el tema http://www.ibm.com/developerworks/java/tutorials/j-jython1/j-jython1.html y el tema http://www.ibm.com/developerworks/java/tutorials/j-jython2/j-jython2.html .

Scripts de Python

Esta guía del lenguaje de script de Python es una introducción a los componentes que tienen más probabilidad de ser utilizados cuando se ejecutan scripts en IBM SPSS Modeler, incluidos conceptos y principios básicos de programación. Le proporcionará los conocimientos suficientes para comenzar a desarrollar sus propios scripts Python y utilizarlos en IBM SPSS Modeler.

Operaciones

a asignación se realiza mediante un signo de igual (=). Por ejemplo, para asignar el valor "3" a una variable llamada "x" debe utilizar la siguiente sentencia:

```
x = 3
```

El signo igual también se utiliza para asignar datos de tipo de cadena a una variable. Por ejemplo, para asignar el valor "a string value" a la variable "y" utilice la sentencia siguiente:

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

La tabla siguiente enumera algunas de las operaciones numéricas y de comparación utilizadas con más frecuencia y sus descripciones.

Tabla 5. Operaciones numéricas y de comparación comunes

Operación	Descripción
`x < y`	¿Es x menor que y?
`x > y`	¿Es x mayor que y?
`x <= y`	¿Es x menor que o igual a y?
`x >= y`	¿Es x mayor que o igual a y?
`x == y`	¿Es x igual a y?
`x != y`	¿Es x no igual a y?
`x <> y`	¿Es x no igual a y?
`x + y`	Sumar y a x
`x - y`	Restar y de x
`x * y`	Multiplicar x por y
`x / y`	Dividir x por y
`x ** y`	Elevar x a la potencia de y

Listas

Las listas son secuencias de elementos. Una lista puede contener cualquier número de elementos, y los elementos de la lista pueden ser cualquier tipo de objeto. Las listas también se pueden considerar como matrices. El número de elementos de una lista puede aumentar o disminuir a medida que se añaden, eliminan o sustituyen elementos.

Ejemplos

`[]`	Cualquier lista vacía.
`[1]`	Una lista con un solo elemento, un entero.
`["Mike", 10, "Don", 20]`	Una lista con cuatro elementos, dos elementos de cadena y dos elementos de entero.
`[[],[7],[8,9]]`	Una lista de listas. Cada sublista es una lista vacía o una lista de elementos de enteros.

```
x = 7; y = 2; z = 3;
[1, x, y, x + y]
```

Una lista de enteros. Este ejemplo muestra el uso de variables y expresiones.

Puede asignar una lista a una variable, por ejemplo:

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

A continuación, puede acceder a los elementos específicos de la lista, por ejemplo:

```
mylist[0]
```

Esto genera el resultado siguiente:

```
uno
```

El número entre corchetes ([]) se considera un *index* y hace referencia a un elemento concreto de la lista. Los elementos de una lista se indexan a partir de 0.

También puede seleccionar un rango de elementos de una lista; esto se denomina *porciones*. Por ejemplo, `x[1:3]` selecciona el segundo y el tercer elemento de `x`. El índice final es uno más allá de la selección.

Cadenas

Una *cadena* es una secuencia inmutable de caracteres que se trata como un valor. Las cadenas dan soporte a todas las funciones de secuencias inmutables y operadores que generan como resultado una nueva serie. Por ejemplo, `"abcdef"[1:4]` da como resultado la salida `"bcd"`.

En Python, los caracteres se representan mediante cadenas de caracteres de longitud uno.

Los literales de cadenas se definen mediante comillas simples o triples. Las cadenas definidas mediante comillas simples no pueden abarcar líneas, mientras que las series definidas mediante comillas triples sí que pueden. Una cadena puede estar entre comillas simples (') o entre comillas dobles ("). Un carácter entrecomillado puede contener el otro carácter entrecomillado o el carácter entrecomillado de escape, que es el carácter de barra invertida (\).

Ejemplos

```
"Esta es una cadena"
'Esta también es una cadena'
"Es una cadena"
'Este manual se titula "Guía de Python Scripting and Automation".'
"Estas son comillas de escape (\") en una cadena entrecomillada"
```

El analizador de Python automáticamente concatena varias cadenas separadas por un espacio en blanco. Esto facilita la entrada se cadenas largas y la combinación de tipos de comillas en una sola cadena, por ejemplo:

```
"Esta cadena utiliza ' y " 'esta cadena utiliza ".'
```

Esto resulta en la siguiente salida:

```
Esta cadena utiliza ' y esa cadena utiliza ".
```

Las cadenas dan soporte a varios métodos útiles. Algunos de estos métodos se proporcionan en la tabla siguiente.

Tabla 6. Métodos de serie

Método	**Uso**
`s.capitalize()`	Mayúscula inicial `s`
`s.count(ss {,start {,end}})`	Recuento de apariciones de `ss` en `s[start:end]`

Tabla 6. Métodos de serie (continuación)

Método	Uso
`s.startswith(str {, start {, end}})` `s.endswith(str {, start {, end}})`	Probar si `s` comienza por `str` Probar si `s` acaba en `str`
`s.expandtabs({size})`	Sustituir tabulaciones por espacios, el valor predeterminado de `size` es 8
`s.find(str {, start {, end}})` `s.rfind(str {, start {, end}})`	Busca el primer índice de `str` en `s`; si no se encuentra, el resultado es -1. `rfind` busca de derecha a izquierda.
`s.index(str {, start {, end}})` `s.rindex(str {, start {, end}})`	Busca el primer índice de `str` en `s`; si no se encuentra: se genera `ValueError`. `rindex` busca de derecha a izquierda.
`s.isalnum`	Probar si la cadena es alfanumérica
`s.isalpha`	Probar si la cadena es alfabética
`s.isnum`	Probar si la cadena es numérica
`s.isupper`	Probar si la cadena está toda en mayúsculas
`s.islower`	Probar si la cadena está toda en minúsculas
`s.isspace`	Probar si la cadena está toda en espacios en blanco
`s.istitle`	Probar si la cadena es una secuencia de cadenas alfanuméricas con mayúscula inicial

`s.lower()`
`s.upper()`
`s.swapcase()`
`s.title()`
	Convertir todo a minúsculas
	Convertir todo a mayúsculas
	Convertir de mayúsculas a minúsculas o viceversa
	Convertir todo a mayúsculas o minúsculas del título

`s.join(seq)`	Unir las cadenas de `seq` con `s` como separador
`s.splitlines({keep})`	Dividir `s` en líneas, si keep es `true`, mantener las nuevas líneas
`s.split({sep {, max}})`	Dividir `s` en "palabras" utilizando `sep` (el valor predeterminado de `sep` es un espacio en blanco) para un máximo de `max` veces

`s.ljust(width)`
`s.rjust(width)`
`s.center(width)`
`s.zfill(width)`
	Justificar la cadena a la izquierda con un ancho de campo de `width`
	Justificar la cadena a la derecha con un ancho de campo de `width`
	Justificar la cadena al centro con un ancho de campo de `width`
	Rellenar con 0.

`s.lstrip()`
`s.rstrip()`
`s.strip()`
	Eliminar espacios en blanco iniciales
	Eliminar espacios en blanco de cola
	Eliminar espacios en blanco iniciales y de cola

`s.translate(str {,delc})`	Traducir `s` utilizando la tabla, después eliminar cualquier carácter de `delc`. `str` debe ser una cadena con una longitud de `== 256`.
`s.replace(old, new {, max})`	Sustituye todas las apariciones de `max` de la cadena `old` por la cadena `new`

Observaciones

Los comentarios se introducen con el signo de almohadilla (#) o hash. Todo el texto que sigue al signo de almohadilla en la misma línea se considera parte del comentario y se omite. Un comentario puede comenzar en cualquier columna. El ejemplo siguiente muestra el uso de los comentarios:

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

Sintaxis de las sentencias

La sintaxis de las sentencias para Python es muy sencilla. En general, cada línea de origen es una sola sentencia. A excepción de las sentencias `expression` y `assignment`, cada sentencia se introduce mediante un nombre de palabra clave, tal como `if` o `for`. Las líneas en blanco o las líneas de comentarios se pueden insertar en cualquier lugar entre cualquier sentencia del código. Si existe más de una sentencia en una línea, cada sentencia debe estar separada por un signo de punto y coma (`;`).

Las sentencias muy largas puede continuar en más de una línea. En este caso, la sentencia que ha de continuar en la línea siguiente debe acabar con una barra invertida (\), por ejemplo:

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

Cuando una estructura está encerrada entre paréntesis (`()`), corchetes (`[]`) o llaves (`{}`), la sentencia puede continuar en una línea nueva después de cualquier coma, sin tener que insertar una barra invertida, por ejemplo:

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

Identificadores

Los identificadores se utilizan para el nombre de las variables, funciones, clases y palabras clave. Los identificadores pueden tener cualquier longitud, pero debe empezar con un carácter alfabético en mayúsculas o minúsculas o el carácter de subrayado (_). Los nombres que empiezan con un carácter de subrayado están generalmente reservados para los nombres internos o privados. Después del primer carácter, el identificador puede contener cualquier número y combinación de caracteres alfabéticos, los números del 0-9, y el carácter de subrayado.

Existen algunas palabras reservadas en Jython que no se pueden utilizar para el nombre de variables, funciones o clases. Estas palabras entran en las siguientes categorías:

- **Introducciones de sentencias:** `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else`, `except`, `exec`, `finally`, `for`, `from`, `global`, `if`, `import`, `pass`, `print`, `raise`, `return`, `try` y `while`
- **Introducciones de parámetros:** `as`, `import` y `in`
- **Operadores:** `and`, `in`, `is`, `lambda`, `not` y `or`

El uso incorrecto de palabras claves suele generar `SyntaxError`.

Bloques de código

Bloques de código son grupos de sentencias que se utilizan donde se esperan sentencias individuales. Los bloques de código pueden seguir a cualquiera de las sentencias siguientes: `if`, `elif`, `else`, `for`, `while`, `try`, `except`, `def` y `class`. Estas sentencias introducen el bloque de código con el carácter de dos puntos (`:`), por ejemplo:

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

Se utiliza la indentación para delimitar los bloques de código (en lugar de las llaves que se utilizan en Java). Todas las líneas de un bloque han de indentarse en la misma posición. Esto es debido a que un cambio en la indentación indica el final de un bloque de código. Normalmente la indentación es de cuatro espacios por nivel. Se recomienda utilizar espacios para la indentación, en lugar de tabulaciones. No se deben combinar espacios y tabulaciones. Las líneas del bloque de un módulo situado más al extremo deben comenzar en la columna uno, de lo contrario, se genera el error `SyntaxError`.

Las sentencias que componen un bloque de código (y siguen el signo de dos puntos) también deben estar en una sola línea, separadas por signos de punto y coma, por ejemplo:

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

Pasar argumentos a un script

Pasar argumentos a un script puede resultar útil para poder utilizar un script reiteradamente sin modificarlo. Los argumentos se pasan en la línea de comandos como valores de la lista `sys.argv`. El número de valores que se pasan se puede obtener mediante el comando `len(sys.argv)`. Por ejemplo:

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

En este ejemplo, el comando `import` importa toda la clase `sys`, por lo que se pueden utilizar los métodos existentes para esta clase, tales como `argv`.

El script de este ejemplo se puede invocar utilizando la línea siguiente:

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

Esto genera el resultado siguiente:

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

Ejemplos

La palabra clave `print` imprime los argumentos situados inmediatamente después de la misma. Si la sentencia va seguida de una coma, no se incluye una línea nueva en los resultados. Por ejemplo:

```
print "Esto muestra el uso de una",
print " coma al final de una sentencia de impresión."
```

Esto genera el resultado siguiente:

```
Esto muestra el uso de una coma al final de una sentencia de impresión.
```

La sentencia `for` se utiliza para la iteración por un bloque de código. Por ejemplo:

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

En este ejemplo, se asignan tres cadenas a la lista `mylist1`. Los elementos de la lista se imprimen a continuación, con un elemento de cada línea. Esto genera el resultado siguiente:

```
uno
two
tres
```

En este ejemplo, el iterador `lv` toma el valor de cada elemento de la lista `mylist1` por orden, mientras el bucle for implementa el bloque de código de cada elemento. Un iterador puede ser cualquier identificador válido de cualquier longitud.

La sentencia `if` es una sentencia condicional. Evalúa la condición y devuelve true o false, en función del resultado de la evaluación. Por ejemplo:

```
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
```

En este ejemplo, se evalúa el valor del iterador `lv`. Si el valor de `lv` es `two` se devuelve una cadena diferente a la cadena que se devuelve si el valor de `lv` no es `two`. Esto resulta en la siguiente salida:

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

Métodos matemáticos

Desde el módulo `matemáticas` puede acceder a métodos matemáticos útiles. Algunos de estos métodos se proporcionan en la tabla siguiente. A menos que se especifique lo contrario, todos los valores se devuelven como valores flotantes.

Tabla 7. Métodos matemáticos

Método	Uso
`math.ceil(x)`	Devuelve el punto más alto de `x` como un valor flotante, que es el entero más pequeño mayor o igual a `x`
`math.copysign(x, y)`	Devuelve `x` con el signo de `y`. `copysign(1, -0.0)` devuelve `-1`
`math.fabs(x)`	Devuelve el valor absoluto de `x`
`math.factorial(x)`	Devuelve el factor de `x`. Si `x` es negativo o no es un entero, se genera `ValueError`.
`math.floor(x)`	Devuelve el punto más bajo de `x` como un valor flotante, que es el entero más alto menor o igual a `x`
`math.frexp(x)`	Devuelve la mantisa (`m`) y el exponente (`e`) de `x` como el par (`m, e`). `m` es un valor flotante y `e` es un entero, tal como `x == m * 2**e` exactamente. Si `x` es cero, devuelve `(0,0, 0)`, de lo contrario `0,5 <= abs(m) < 1`.
`math.fsum(iterable)`	Devuelve una suma de coma flotante precisa de los valores de `iterable`
`math.isinf(x)`	Comprueba si el valor flotante `x` es positivo o negativo infinito
`math.isnan(x)`	Comprueba si el valor flotante `x` es `NaN` (no es un número)
`math.ldexp(x, i)`	Devuelve `x * (2**i)`. Esencialmente es la función inversa de `frexp`.
`math.modf(x)`	Devuelve las partes de fracción y entero de `x`. Los dos resultados llevan el signo de `x` y son flotantes.
`math.trunc(x)`	Devuelve el valor `Real` de `x`, que se ha truncado en un `Integral`.
`math.exp(x)`	Devuelve `e**x`
`math.log(x[, base])`	Devuelve el logaritmo de `x` para el valor dado de `base`. Si no se especifica `base`, se devuelve el logaritmo natural de `x`.
`math.log1p(x)`	Devuelve el logaritmo natural de `1+x (base e)`
`math.log10(x)`	Devuelve el logaritmo de base-10 de `x`

Tabla 7. Métodos matemáticos (continuación)

Método	Uso
`math.pow(x, y)`	Devuelve `x` elevado a la potencia de `y`. `pow(1.0, x)` y `pow(x, 0.0)` siempre devuelve 1, incluso si `x` es cero o NaN.
`math.sqrt(x)`	Devuelve la raíz cuadrada de `x`

Además de las funciones matemáticas, hay algunos métodos trigonométricos útiles. Estos métodos se muestran en la siguiente tabla.

Tabla 8. Métodos trigonométricos

Método	Uso
`math.acos(x)`	Devuelve el arco coseno de `x` en radianes
`math.asin(x)`	Devuelve el arcoseno de `x` en radianes
`math.atan(x)`	Devuelve el arco tangente de `x` en radianes
`math.atan2(y, x)`	Devuelve `atan(y / x)` en radianes.
`math.cos(x)`	Devuelve el coseno de `x` en radianes.
`math.hypot(x, y)`	Devuelve la norma euclidiana de `sqrt(x*x + y*y)`. Esta es la longitud del vector desde el origen al punto `(x, y)`.
`math.sin(x)`	Devuelve el seno de `x` en radianes
`math.tan(x)`	Devuelve la tangente de `x` en radianes
`math.degrees(x)`	Convierte el ángulo `x` de radianes a grados
`math.radians(x)`	Convierte el ángulo `x` de grados a radianes
`math.acosh(x)`	Devuelve el coseno hiperbólico inverso de `x`
`math.asinh(x)`	Devuelve el seno hiperbólico inverso de `x`
`math.atanh(x)`	Devuelve la tangente hiperbólica inversa de `x`
`math.cosh(x)`	Devuelve el coseno hiperbólico de `x`
`math.sinh(x)`	Devuelve el coseno hiperbólico de `x`
`math.tanh(x)`	Devuelve la tangente hiperbólica de `x`

También hay constantes matemáticas. El valor de `math.pi` es la constante matemática pi. El valor de `math.e` es la constante matemática e.

Utilización de caracteres no ASCII

Para utilizar caracteres no ASCII, Python requiere la codificación y descodificación explícitas de las cadenas en Unicode. En IBM SPSS Modeler, se presupone que los scripts Python están codificados UTF-8, la cual es una codificación Unicode estándar que da soporte a caracteres no ASCII. El script siguiente se compilará porque SPSS Modeler ha establecido el compilador Python en UTF-8.

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

Sin embargo, el nodo resultante tendrá una etiqueta incorrecta.

Figura 3. Etiqueta del nodo que contiene caracteres no ASCII, visualiza incorrectamente

La etiqueta es incorrecta porque Python ha convertido el propio literal de serie en una cadena ASCII.

Python permite que los literales de cadenas Unicode se especifiquen añadiendo un prefijo con el carácter `u` antes del literal de cadena:

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

Esto crear una cadena Unicode y la etiqueta aparecerá correctamente.

Figura 4. Etiqueta de nodo que contiene caracteres no ASCII, visualizados correctamente

La utilización de Python y Unicode es un tema de gran volumen que no entra dentro del ámbito de este documento. Existen muchas publicaciones y recursos en línea disponibles que describen detalladamente este tema.

Programación orientada a objetos

La programación orientada a objetos se basa en el concepto de crear un modelo del problema de destino en sus programas. La programación orientada a objetos disminuye los errores y promociona la reutilización del código. Python es un lenguaje orientado a objetos. Los objetos definidos en Python tienen las características siguientes:

- **Identidad.** Cada objeto debe ser distinguido y ello debe poder demostrarse mediante pruebas. Las pruebas `is` e `is not` existen para este fin.
- **Estado** Cada objeto debe ser capaz de almacenar el estado. Para este fin, existen atributos, tales como variables de instancias y campos.
- **Comportamiento.** Cada objeto debe ser capaz de manipular su estado. Para este fin existen métodos.

Python incluye las características siguientes para dar soporte a la programación orientada a objetos:

- **Creación de objetos basada en clases.** Las clases son plantillas para la creación de objetos. Los objetos son estructuras de datos con el comportamiento asociado.
- **Herencia con polimorfismo.** Python da soporte a la herencia individual y múltiple. Todos los métodos de instancias de Python son polimórficos y se pueden alterar temporalmente mediante subclases.
- **Encapsulación con ocultación de datos.** Python permite ocultar los atributos. Cuando se ocultan los atributos, se puede acceder a los mismos desde fuera de la clase únicamente mediante los métodos de la clase. Las clases implementan métodos para modificar los datos.

Definición de una clase

En una clase Python, se pueden definir tanto variables como métodos. A diferencia de Java, en Python puede definir cualquier número de clases públicas por archivo de origen (o *module*). Por lo tanto, un módulo en Python puede considerarse similar a un paquete en Java.

En Python, las clases se definen utilizando la sentencia `class`. La sentencia `class` tiene el formato siguiente:

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

o

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

Cuando define una clase tiene la opción de proporcionar cero o más sentencias *assignment*. Estos crean atributos de clase que comparten todas las instancias de la clase. Puede proporcionar cero o más definiciones de *function*. Estas definiciones de función crean métodos. La lista de superclases es opcional.

El nombre de clase debe ser exclusivo en el mismo ámbito, esto es, dentro de un módulo, función o clase. Puede definir varias variables para que hagan referencia a la misma clase.

Creación de una instancia de clase

Las clases se utilizan para contener (o compartir) los atributos de clase o para crear instancias de clase. Para crear una instancia de una clase, debe llamar a la clase como si fuera una función. Por ejemplo, considere las clases siguientes:

```
class MyClass:
    pass
```

Aquí, se utiliza la sentencia `pass` por que se requiere una sentencia para completar la clase, pero no se requiere ninguna acción de programación.

La sentencia siguiente crea una instancia de la clase `MyClass`:

```
x = MyClass()
```

Añadir atributos a una instancia de clase

A diferencia de Java, en Python los clientes pueden añadir atributos a una instancia de una clase. Solo se cambia la instancia. Por ejemplo, para añadir atributos a una instancia de `x`, establezca valores nuevos en dicha instancia:

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

Definición de atributos de clase y métodos

Cualquier variable enlazada a una clase es un *atributo de clase*. Cualquier función definida en una clase es un *método*. Los métodos reciben como primer argumento una instancia de la clase, que convencionalmente se denomina `self`. Por ejemplo, para definir algunos atributos de clase y métodos, puede entrar el siguiente código:

```
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
```

Dentro de una clase, debe cualificar todas las referencias a los atributos de clase con el nombre de clase; por ejemplo, `MyClass.attr1`. Todas las referencias a los atributos de la instancia deben cualificarse con la variable `self`, por ejemplo, `self.text`. Fuera de la clase, debe cualificar todas las referencias a los atributos de clase con el nombre de clase (por ejemplo, `MyClass.attr1`) o con una instancia de la clase (por ejemplo, `x.attr1`, donde `x` es una instancia de la clase). Fuera de la clase, todas las referencias a las variables de la instancia deben cualificarse con una instancia de la clase, por ejemplo, `x.text`.

Variables ocultas

Los datos se pueden ocultar creando variables *privadas*. Solo la propia clase puede acceder a las variables privadas. Si declara nombres con el formato `__xxx` o `__xxx_yyy`, estos es, con dos signos de subrayado antes de los nombres, el analizador Python automáticamente añadirá el nombre de clase al nombre declarado y creará las variables ocultas, por ejemplo:

```
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
```

A diferencia de Java, en Python todas las referencias a variables de instancia deben estar calificadas con `self`; no existe un uso implícito de `this`.

Heredado

La posibilidad de herencia de las clases es fundamental en la programación orientada a objetos. Python da soporte a la herencia individual y múltiple. *Herencia individual* significa que solo puede haber una superclase. *Herencia múltiple* significa que puede haber más de una superclase.

La herencia se implementa generando subclases de otras clases. Cualquier número de clases Python pueden ser superclases. En la implementación de Jython en Python, solo se puede heredar directa o indirectamente de una clase Java. No es necesario suministrar una superclase.

Cualquier atributo o método de una superclase también está en cualquier subclase y lo puede utilizar la propia clase o cualquier cliente, siempre que el atributo o método no esté oculto. Se puede utilizar cualquier instancia de una subclase; esto se denomina *polimorfismo*. Estas características permiten la reutilización y facilitan la extensión.

Ejemplo

```
class Class1: pass    #no inheritance

class Class2: pass

class Class3(Class1): pass     #single inheritance

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

Capítulo 3. Scripts de IBM SPSS Modeler

Tipos de scripts

En IBM SPSS Modeler existen tres tipos de scripts:

- Los *scripts de ruta* se utilizan para controlar la ejecución de una sola ruta y se almacenan dentro de la ruta.
- Los *scripts Supernodo* se utilizan para controlar el comportamiento de los supernodos.
- Los *scripts autónomos o de sesión* se pueden utilizar para coordinar la ejecución entre un número de rutas diferentes.

Existen diferentes métodos disponibles que puede utilizar en scripts en IBM SPSS Modeler lo que le permite acceder a una amplia gama de funciones de SPSS Modeler. Estos métodos se utilizan también en Capítulo 4, "API de scripts", en la página 37 para crear funciones más avanzadas.

Rutas, rutas de supernodo y diagramas

La mayoría de las veces, el término *ruta* significa lo mismo independientemente de que se trate de una ruta cargada de un archivo o utilizada dentro de un supernodo. En general significa una colección de nodos conectados entre sí que puede ejecutarse. Sin embargo, en la creación de scripts no todas las operaciones se soportan en todos los sitios, lo que significa que el autor de un script deberá tener en cuenta qué variante de ruta está utilizando.

Rutas

Una ruta es el principal tipo de documento de IBM SPSS Modeler. Se puede guardar, cargar, editar y ejecutar. Las rutas también pueden tener parámetros, valores globales, un script y otra información asociada a ellos.

Rutas de Supernodo

Una *ruta de Supernodo* es el tipo de ruta que se utiliza en un Supernodo. Al igual que una ruta normal, contiene nodos enlazados entre sí. Las rutas de Supernodo tienen una serie de diferencias respecto de una ruta normal.

- Los parámetros y scripts están asociados al Supernodo propietario de la ruta de Supernodo en lugar de a la propia ruta de Supernodo.
- Las rutas de Supernodo tienen nodos de conector de entrada y salida adicionales dependiendo del tipo de Supernodo. Estos nodos de conector se utilizan en los flujos de información entrantes y salientes de la ruta de Supernodo y se crean automáticamente cuando se crea el Supernodo.

Diagramas

El término *diagrama* abarca las funciones soportadas en rutas normales y en rutas de supernodo como, por ejemplo, la adición y eliminación de nodos y la modificación de conexiones entre nodos.

Ejecución de una ruta

El ejemplo siguiente ejecuta todos los nodos ejecutables en la ruta y es el tipo de script de ruta más sencillo:

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

El ejemplo siguiente también se ejecuta todos los nodos ejecutables de la ruta:

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

En este ejemplo, la ruta se almacena en una variable denominada `stream`. Almacenar la ruta en una variable resulta útil ya que un script se utiliza generalmente para modificar la ruta o los nodos contenidos en una ruta. Si se crea una variable que almacena los resultados de la ruta, el script resultará más conciso.

El contexto de los scripts

El módulo `modeler.script` proporciona el contexto en el que se ejecuta un script. El módulo se importa automáticamente a un script de SPSS Modeler durante la ejecución. El módulo define cuatro funciones que proporcionan un script con acceso a su entorno de ejecución:

- La función `session()` devuelve la sesión para el script. La sesión define información, tal como el entorno local y el proceso de fondo de SPSS Modeler (ya sea un proceso local o un proceso de SPSS Modeler Server conectado a la red) que se está utilizando para ejecutar rutas.
- La función `stream()` se puede utilizar con la ruta y los scripts Supernodo. Esta función devuelve la ruta que es propietaria del script de ruta o el script Supernodo que se está ejecutando.
- La función `diagram()` se puede utilizar con los scripts Supernodo. Esta función devuelve el diagrama dentro del Supernodo. Para otros tipos de script, esta función devuelve el mismo que la función `stream()`.
- La función `supernode()` se puede utilizar con los scripts Supernodo. Esta función devuelve el Supernodo propietario del script que se está ejecutando.

En la tabla siguiente se resumen las cuatro funciones y sus resultados.

Tabla 9. Resumen de las funciones de `modeler.script`

Tipo de script	`session()`	`stream()`	`diagram()`	`supernode()`
Autónomo	Devuelve una sesión	Devuelve la ruta gestionada actual en el momento en que se invoca el script (por ejemplo, la ruta se pasa con la opción `-stream` de modalidad de proceso por lotes) o `None`.	Igual que para `stream()`	No es aplicable
Ruta	Devuelve una sesión	Devuelve una ruta	Igual que para `stream()`	No es aplicable
Supernodo	Devuelve una sesión	Devuelve una ruta	Devuelve una ruta Supernodo	Devuelve un Supernodo

El módulo `modeler.script` también define un modo de finalizar el script con un código de salida. La función `exit(exit-code)` detiene la ejecución del script y devuelve el código de salida de entero suministrado.

Uno de los métodos que se define para una ruta es `runAll(List)`. Este método ejecuta todos los nodos ejecutables. Los modelos o resultados que se generan mediante la ejecución de los nodos se añaden a la lista suministrada.

Es común que la ejecución de ruta genere resultados, tales como modelos, gráficos y otros. Para capturar este resultado, un script puede proporcionar una variable que se inicializa en una lista, por ejemplo:

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

Cuando se completa la ejecución, se puede acceder a todos los objetos generados por la ejecución en la lista `results`.

Referencia a nodos existentes

Una ruta suele estar construida previamente con algunos parámetros que se deben modificar antes de ejecutar la ruta. Para modificar estos parámetros se han de realizar las tareas siguientes:

1. Localizar los nodos en la ruta relevante.
2. Cambiar los valores de los nodos o de la ruta (o de ambas cosas).

Buscar nodos

Las rutas proporcionan varios modos de localizar un nodo existente. Estos métodos se resumen en la siguiente tabla.

Tabla 10. Métodos para localizar un nodo existente

Método	Tipo devuelto	Descripción
`s.findAll(type, label)`	Colección	Devuelve una lista de todos los nodos con el tipo y la etiqueta. El tipo o la etiqueta pueden ser `None`, en cuyo caso se utiliza el otro parámetro.
`s.findAll(filter, recursive)`	Colección	Devuelve una colección de todos los nodos que están aceptados por el filtro especificado. Si el distintivo recursivo es `True`, también se buscan los supernodos contenidos en la ruta especificada.
`s.findByID(id)`	Nodo	Devuelve el nodo con el ID proporcionado o `None` si no existe dicho nodo. La búsqueda se limita a la ruta actual.
`s.findByType(type, label)`	Nodo	Devuelve el nodo con el tipo, etiqueta o ambas cosas. El tipo o el nombre pueden ser `None`, en cuyo caso se utiliza el otro parámetro. Si varios nodos dan como resultado una coincidencia, se elige uno arbitrario y se devuelve. Si ningún nodo da como resultado una coincidencia, se devuelve el valor `None`.
`s.findDownstream(fromNodes)`	Colección	Busca en la lista de nodos suministrada y devuelve el conjunto de nodos en sentido descendente de los nodos suministrados. La lista devuelta incluye los nodos proporcionados originalmente.

Tabla 10. Métodos para localizar un nodo existente (continuación)

Método	Tipo devuelto	Descripción
`s.findUpstream(fromNodes)`	Colección	Busca en la lista de nodos suministrada y devuelve el conjunto de nodos en sentido ascendente de los nodos suministrados. La lista devuelta incluye los nodos proporcionados originalmente.

Por ejemplo, si una ruta contiene un nodo Filtro único que el script necesita para acceso, el nodo Filtro se puede encontrar utilizando el siguiente script:

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

Como alternativa, si se conoce el ID del nodo (tal como se muestra en la pestaña Anotaciones del cuadro de diálogo del nodo) se puede utilizar el ID para buscar el nodo, por ejemplo:

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

Definición de propiedades

Los nodos, rutas, modelos y resultados tienen propiedades a las que se puede acceder y que, en la mayor parte de los casos, se pueden establecer. Las propiedades suelen utilizarse para modificar el aspecto o el comportamiento del objeto. En la tabla siguiente se resumen los métodos disponibles para establecer y acceder a las propiedades de los objetos.

Tabla 11. Métodos para establecer y acceder a las propiedades de los objetos

Método	Tipo devuelto	Descripción
`p.getPropertyValue(propertyName)`	Object	Devuelve el valor de la propiedad con nombre, o `None` si no existe tal propiedad.
`p.setPropertyValue(propertyName, value)`	No aplicable	Establece el valor de la propiedad con nombre.
`p.setPropertyValues(properties)`	No aplicable	Establece los valores de la propiedad con nombre. Cada entrada de la correlación de propiedades consta de una clave que representa el nombre de la propiedad y del valor que debe asignarse a la propiedad.
`p.getKeyedPropertyValue( propertyName, keyName)`	Object	Devuelve el valor de la propiedad con nombre, o `None` si no existe dicha propiedad o clave.
`p.setKeyedPropertyValue( propertyName, keyName, value)`	No aplicable	Establece el valor de la propiedad con nombre y de la clave.

Por ejemplo, si desea establecer el valor de un nodo Archivo variable al comienzo de una ruta, puede utilizar el siguiente script:

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

Como alternativa, puede que desee filtrar un campo desde un nodo Filtrar. En este caso, el valor es con clave en el nombre de campo, por ejemplo:

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

Creación de nodos y modificación de rutas

En algunas situaciones, es posible que desee añadir nuevos nodos a rutas existentes. Para añadir nodos a rutas existentes suele ser necesario realizar las tareas siguientes:

1. Crear los nodos.
2. Enlazar los nodos con el flujo de ruta existente.

Creación de nodos

Las rutas proporcionan varios modos de crear nodos. Estos métodos se resumen en la siguiente tabla.

Tabla 12. Métodos para crear nodos

Método	Tipo devuelto	Descripción
`s.create(nodeType, name)`	Nodo	Crea un nodo del tipo especificado y lo añade a la ruta especificada.
`s.createAt(nodeType, name, x, y)`	Nodo	Crea un nodo del tipo especificado y lo añade a la ruta especificada en la ubicación especificada. Si x< 0 o y < 0, no se establece la ubicación.
`s.createModelApplier(modelOutput, name)`	Nodo	Crea un nodo aplicador de modelos que se deriva del objeto de resultados del modelo proporcionado.

Por ejemplo, para crear un tipo de nodo nuevo en una ruta puede utilizar el siguiente script:

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

Enlazar y desenlazar nodos

Cuando un nodo nuevo se crea dentro de una ruta, debe estar conectado a una ruta de nodos para poder utilizarlo. Las rutas proporciona varios métodos para enlazar y desenlazar nodos. Estos métodos se resumen en la siguiente tabla.

Tabla 13. Métodos para enlazar y desenlazar nodos

Método	Tipo devuelto	Descripción
`s.link(source, target)`	No es aplicable	Crea un nuevo enlace entre los nodos de origen y destino.
`s.link(source, targets)`	No es aplicable	Crea nuevos enlaces entre el nodo de origen y cada nodo de destino de la lista suministrada.

Tabla 13. Métodos para enlazar y desenlazar nodos (continuación)

Método	Tipo devuelto	Descripción
`s.linkBetween(inserted, source, target)`	No es aplicable	Conecta un nodo entre dos instancias de otro nodo (los nodos de origen y de destino) y establece la posición del nodo insertado de modo que quede entre ellos. Cualquier enlace directo entre los nodos de origen y de destino se elimina en primer lugar.
`s.linkPath(path)`	No es aplicable	Crea una nueva ruta entre instancias de nodo. El primer nodo se enlaza con el segundo, el segundo nodo se enlaza con el tercero y así sucesivamente.
`s.unlink(source, target)`	No es aplicable	Elimina cualquier enlace directo entre los nodos de origen y de destino.
`s.unlink(source, targets)`	No es aplicable	Elimina los enlaces directos entre el nodo de origen y cada objeto de la lista de destinos.
`s.unlinkPath(path)`	No es aplicable	Elimina cualquier ruta que existe entre las instancias del nodo.
`s.disconnect(node)`	No es aplicable	Elimina los enlaces entre el nodo suministrado y todos los demás nodos de la ruta especificada.
`s.isValidLink(source, target)`	*booleano*	Devuelve `True` si es válido crear un enlace entre los nodos de origen y de destino especificados. Este método comprueba que ambos objetos pertenezcan a la ruta especificada, que el nodo de origen puede proporcionar un enlace y que el nodo de destino puede recibir un enlace, y que la creación de un enlace de este tipo no creará un circularidad en la ruta.

El script de ejemplo siguiente realiza estas cinco tareas:

1. Crea un nodo de entrada Archivo de variables, un nodo Filtro y un nodo de salida Tabla.
2. Conecta los nodos entre sí.
3. Establece el nombre de archivo del nodo de entrada Archivo de variables.
4. Filtra el campo "Drug" de la salida resultante.
5. Ejecute el nodo Tabla.

```
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)
```

Importar, sustituir y eliminar nodos

Además de crear y conectar nodos, a menudo es necesario sustituir y suprimir nodos de la corriente. Los métodos que están disponibles para importar, sustituir y suprimir nodos se resumen en la tabla siguiente.

Tabla 14. Métodos para importar, sustituir y suprimir nodos

Método	Tipo devuelto	Descripción
`s.replace(originalNode, replacementNode, discardOriginal)`	No es aplicable	Sustituye el nodo especificado de la ruta actual. Tanto el nodo original como el nodo de sustitución deben ser propiedad de la ruta especificada.
`s.insert(source, nodes, newIDs)`	Lista	Inserta copias de los nodos en la lista suministrada. Se presupone que todos los nodos de la lista suministrada se encuentran dentro de la ruta especificada. El distintivo `newIDs` indica si se deben generar nuevos ID para cada nodo o si se debe copiar y utilizar el ID existente. Se presupone que todos los nodos de una ruta tienen ID exclusivos, por lo tanto este distintivo se debe establecer en `True` si la ruta de origen es la misma que la ruta especificada. El método devuelve la lista de nodos recién insertados, en la que el orden de los nodos está sin definir (es decir, el orden no es necesariamente el mismo que el orden de los nodos de la lista de entrada).
`s.delete(node)`	No es aplicable	Elimina el nodo especificado de la ruta especificada. El nodo debe ser propiedad de la ruta especificada.
`s.deleteAll(nodes)`	No es aplicable	Elimina todos los nodos especificados de la ruta especificada. Todos los nodos de la colección debe pertenecer a la ruta especificada.
`s.clear()`	No es aplicable	Elimina todos los nodos de la ruta especificada.

Atravesar los nodos de una ruta

Un requisito común es identificar los nodos que están en un punto de la ruta anterior o posterior a un determinado nodo. La ruta proporciona una serie de métodos que pueden utilizarse para identificar estos nodos. Estos métodos se resumen en la siguiente tabla.

Tabla 15. Métodos para identificar los nodos en sentido ascendente y descendente

Método	Tipo devuelto	Descripción
`s.iterator()`	Iterador	Devuelve un iterador de los objetos de nodo que están contenidos en la ruta especificada. Si la ruta se modifica entre llamadas a la función `next()`, el comportamiento del iterador no está definido.

Tabla 15. Métodos para identificar los nodos en sentido ascendente y descendente (continuación)

Método	Tipo devuelto	Descripción
`s.predecessorAt(node, index)`	Nodo	Devuelve el predecesor inmediato especificado del nodo suministrado o `None` si el índice está fuera de los límites.
`s.predecessorCount(node)`	*int*	Devuelve el número de predecesores inmediatos del nodo suministrado.
`s.predecessors(node)`	Lista	Devuelve los predecesores inmediatos del nodo suministrado.
`s.successorAt(node, index)`	Nodo	Devuelve el sucesor inmediato especificado del nodo suministrado o `None` si el índice está fuera de los límites.
`s.successorCount(node)`	*int*	Devuelve el número de sucesores inmediatos del nodo suministrado.
`s.successors(node)`	Lista	Devuelve los sucesores inmediatos del nodo suministrado.

Borrado o eliminación de elementos

Los scripts de herencia admiten varios usos del mandato `claro`, por ejemplo:

- `clear outputs` Para suprimir todos los elementos de salida de la paleta del gestor.
- `clear generated palette` Para borrar todos los nugget de modelos de la paleta de modelos.
- `clear stream` Para eliminar el contenido de una ruta.

Los scripts Python admiten un conjunto de funciones parecido; el mandato `removeAll()` se utiliza para borrar las rutas, las salidas y los gestores de modelos. Por ejemplo:

- Para borrar el gestor de rutas:

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

- Para borrar el gestor de salidas:

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

- Para borrar el gestor de modelos:

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

Obtener información sobre los nodos

Los nodos entran en diferentes categorías, tales como nodos de importación y exportación de datos, nodos de construcción de modelos y otros tipos de nodos. Cada nodo proporciona una serie de métodos que pueden utilizarse para obtener información sobre el nodo.

Los métodos que se pueden utilizar para obtener el ID, el nombre y la etiqueta de un nodo se resumen en la tabla siguiente.

Tabla 16. Métodos para obtener el ID, el nombre y la etiqueta de un nodo

Método	Tipo devuelto	Descripción
`n.getLabel()`	*cadena*	Devuelve la etiqueta de visualización del nodo especificado. La etiqueta es el valor de la propiedad `custom_name` sólo si la propiedad es una serie no vacía y la propiedad `use_custom_name` no está establecida; de lo contrario, la etiqueta es el valor de `getName()`.
`m.setLabel(label)`	No es aplicable	Establece la etiqueta de visualización del nodo especificado. Si la nueva etiqueta es una cadena no vacía se asigna a la propiedad `custom_name`, y `False` se asigna a la propiedad `use_custom_name`, de tal modo que la etiqueta especificada tiene prioridad; de lo contrario, se asigna una serie vacía a la propiedad `custom_name` y se asigna `True` a la propiedad `use_custom_name`.
`n.getName()`	*cadena*	Devuelve el nombre del nodo especificado.
`n.getID()`	*cadena*	Devuelve el ID del nodo especificado. Se crea un ID nuevo cada vez que se crea un nuevo nodo. El ID se conserva con el nodo cuando se guarda como parte de una ruta de modo que, cuando se abre la ruta, los ID de nodo se conservan. Sin embargo, si un nodo guardado se inserta en una ruta, el nodo insertado se considera un nuevo objeto y se le asigna un nuevo ID.

Los métodos que se pueden utilizar para obtener información acerca de un nodo se resumen en la tabla siguiente.

Tabla 17. Métodos para obtener información acerca de un nodo

Método	Tipo devuelto	Descripción
`n.getTypeName()`	*cadena*	Devuelve el nombre de script de este nodo. Este es el mismo nombre que puede utilizarse para crear una nueva instancia de este nodo.
`n.isInitial()`	*Booleana*	Devuelve `True` si es un nodo *initial*, esto es, un nodo que aparece al inicio de una ruta.
`n.isInline()`	*Booleana*	Devuelve `True` si es un nodo *in-line*, esto es, un nodo que aparece a mitad de una ruta.
`n.isTerminal()`	*Booleana*	Devuelve `True` si es un nodo *terminal*, esto es, un nodo que aparece al final de una ruta.
`n.getXPosition()`	*int*	Devuelve el desplazamiento de la posición x del nodo en la ruta.

Tabla 17. Métodos para obtener información acerca de un nodo (continuación)

Método	Tipo devuelto	Descripción
`n.getYPosition()`	*int*	Devuelve el desplazamiento de la posición y del nodo en la ruta.
`n.setXYPosition(x, y)`	No es aplicable	Devuelve la posición del nodo en la ruta.
`n.setPositionBetween(source, target)`	No es aplicable	Establece la posición del nodo en la ruta, de modo que esté posicionado entre los nodos suministrados.
`n.isCacheEnabled()`	*Booleana*	Devuelve `True` si la memoria caché está habilitada; devuelve `False` de lo contrario.
`n.setCacheEnabled(val)`	No es aplicable	Habilita o inhabilita la memoria caché para este objeto. Si la memoria caché está llena y la memoria caché pasa a estar inhabilitada, la memoria caché se vacía.
`n.isCacheFull()`	*Booleana*	Devuelve `True` si la memoria caché está llena; devuelve `False` de lo contrario.
`n.flushCache()`	No es aplicable	Vacía la memoria caché de este nodo. No tiene efecto si la memoria caché no está activada o no está llena.

Capítulo 4. API de scripts

Introducción a la API de scripts

La API de scripts proporciona acceso a una amplia gama de funciones de SPSS Modeler. Todos los métodos descritos hasta ahora forman parte de la API y se puede acceder a los mismos de forma implícita en el script sin importaciones adicionales. Sin embargo, si desea hacer referencia a las clases de la API, debe importar la API explícitamente con la sentencia siguiente:

```
import modeler.api
```

Esta sentencia import es necesaria para muchos de los ejemplos de la API de scripts.

En el documento *Guía de referencia de la API de scripts Python de IBM SPSS Modeler 17* podrá encontrar una guía completa a las clases, métodos y parámetros que disponibles por medio de la API de scripts.

Ejemplo: Buscar nodos utilizando un filtro personalizado

La sección "Buscar nodos" en la página 29 se incluye un ejemplo de cómo buscar un nodo en una ruta utilizando el nombre de tipo del nodo como criterio de búsqueda. En algunas situaciones, se requiere una búsqueda más genérica y ésta se puede implementar utilizando la clase `NodeFilter` y el método `findAll()` de la ruta. Este tipo de búsqueda requiere los pasos siguientes:

1. Crear una clase nueva que amplíe `NodeFilter` e implemente una versión personalizada del método `accept()`.
2. Llamar al método `findAll()` de la ruta con una instancia de esta clase nueva. Esto devuelve todos los nodos que cumplen el criterio definido en el método `accept()`.

El ejemplo siguiente muestra cómo buscar nodos de una ruta que tienen habilitada la memoria caché de nodo. La lista de nodos devuelta se puede utilizar para vaciar o inhabilitar las memorias caché de estos nodos.

```
import modeler.api

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

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

Metadatos: información sobre datos

Puesto que en una ruta los nodos se conectan entre sí, está disponible la información relativa a las columnas o los campos disponibles en cada nodo. Por ejemplo, en la interfaz de usuario de Modeler esto permite seleccionar por qué campos hay que ordenar o agregar. Esta información se llama modelo de datos.

Los scripts también pueden acceder al modelo de datos inspeccionando los campos que entran en un nodo o que salen de él. En algunos nodos coinciden los modelos de datos de entrada y salida; por ejemplo, un nodo Ordenar se limita a cambiar el orden de los registros, sin alterar el modelo de datos. Otros, como el nodo Derivar, pueden añadir nuevos campos. Otros, como el nodo Filtrar, pueden renombrar o eliminar campos.

En el ejemplo siguiente, el script toma la ruta estándar IBM SPSS Modeler `druglearn.str` y, para cada campo, construye un modelo descartando uno de los campos de entrada. Lo hace de la siguiente manera:

1. Accede al modelo de datos de salida del nodo Tipo.
2. Itera cada campo del modelo de datos de salida.
3. Modifica el nodo Filtro de cada campo de entrada.
4. Cambia el nombre del modelo que se construye.
5. Ejecuta el nodo de construcción de modelos.

Nota: Antes de ejecutar el script en la ruta `druglean.str`, no olvide establecer el lenguaje de script a Python (la ruta se creó en una versión anterior de IBM SPSS Modeler, de modo que el lenguaje de script de la ruta está establecido a Herencia).

```
import modeler.api

stream = modeler.script.stream()
filternode = stream.findByType("filter", None)
typenode = stream.findByType("type", None)
c50node = stream.findByType("c50", None)
# Usar siempre un nombre de modelo personalizado
c50node.setPropertyValue("use_model_name", True)

lastRemoved = None
fields = typenode.getOutputDataModel()
for field in fields:
    # Si este es el campo de destino, se hace caso omiso del mismo
    if field.getModelingRole() == modeler.api.ModelingRole.OUT:
        continue

    # Se rehabilita el campo eliminado más recientemente
    if lastRemoved != None:
        filternode.setKeyedPropertyValue("include", lastRemoved, True)

    # Se elimina el campo
    lastRemoved = field.getColumnName()
    filternode.setKeyedPropertyValue("include", lastRemoved, False)

    # Se establece el nombre del nuevo modelo y se ejecuta la construcción
    c50node.setPropertyValue("model_name", "Exclude " + lastRemoved)
    c50node.run([])
```

El objeto DataModel (modelo de datos) proporciona una serie de métodos de acceso a la información relativa a los campos y columnas del modelo de datos. Estos métodos se resumen en la siguiente tabla.

Tabla 18. Métodos del objeto DataModel de acceso a la información relativa a campos o columnas

Método	Tipo devuelto	Descripción
`d.getColumnCount()`	*int*	Devuelve el número de columnas del modelo de datos.
`d.columnIterator()`	Iterator	Devuelve un iterador que devuelve cada columna en el orden "natural" de inserción. El iterador devuelve instancias de Column.
`d.nameIterator()`	Iterator	Devuelve un iterador que devuelve el nombre de cada columna en el orden "natural" de inserción.
`d.contains(nombre)`	*Boolean*	Devuelve `True` si en este DataModel existe una columna con el nombre proporcionado y `False` en caso contrario.
`d.getColumn(nombre)`	Column	Devuelve la columna cuyo nombre es el especificado.

Tabla 18. Métodos del objeto DataModel de acceso a la información relativa a campos o columnas (continuación)

Método	Tipo devuelto	Descripción
`d.getColumnGroup(nombre)`	ColumnGroup	Devuelve el grupo de columnas nombrado o `None` si no existe dicho grupo de columnas.
`d.getColumnGroupCount()`	*int*	Devuelve el número de grupos de columnas de este modelo de datos.
`d.columnGroupIterator()`	Iterator	Devuelve un iterador que devuelve a su vez cada grupo de columnas.
`d.toArray()`	Column[]	Devuelve el modelo de datos como un vector de columnas. Las columnas van ordenadas según su orden "natural" de inserción.

Cada campo (objeto Column) incluye una serie de métodos de acceso a la información de la columna. La tabla que se muestra a continuación muestra una selección de los mismos.

Tabla 19. Métodos del objeto Column de acceso a la información de una columna

Método	Tipo devuelto	Descripción
`c.getColumnName()`	*string*	Devuelve el nombre de la columna.
`c.getColumnLabel()`	*string*	Devuelve la etiqueta de la columna o una cadena vacía si no hay ninguna etiqueta asociada a la columna.
`c.getMeasureType()`	MeasureType	Devuelve el tipo de medición de la columna.
`c.getStorageType()`	StorageType	Devuelve el tipo de almacenamiento de la columna.
`c.isMeasureDiscrete()`	*Boolean*	Devuelve `True` si la columna es discreta. Se consideran discretas las columnas que son un conjunto o un distintivo.
`c.isModelOutputColumn()`	*Boolean*	Devuelve `True` si la columna es una columna de resultado del modelo.
`c.isStorageDatetime()`	*Boolean*	Devuelve `True` si el almacenamiento de la columna es un valor de hora, fecha o indicación de fecha y hora.
`c.isStorageNumeric()`	*Boolean*	Devuelve `True` si el almacenamiento de la columna es un entero o un número real.
`c.isValidValue(value)`	*Boolean*	Devuelve `True` si el valor especificado es válido para este almacenamiento y `valid` cuando se conocen los valores de columna válidos.
`c.getModelingRole()`	ModelingRole	Devuelve el rol de modelado de la columna.
`c.getSetValues()`	Object[]	Devuelve un vector de valores válidos para la columna o `None` si se desconocen los valores o si la columna no es un conjunto.

Tabla 19. Métodos del objeto Column de acceso a la información de una columna (continuación)

Método	Tipo devuelto	Descripción
`c.getValueLabel(valor)`	*string*	Devuelve la etiqueta del valor de la columna o una cadena vacía si no hay ninguna etiqueta asociada al valor.
`c.getFalseFlag()`	Object	Devuelve el valor indicador de "falso" de la columna o `None` si se desconoce el valor o la columna no es un indicador.
`c.getTrueFlag()`	Object	Devuelve el valor indicador de "verdadero" de la columna o `None` si se desconoce el valor o la columna no es un indicador.
`c.getLowerBound()`	Object	Devuelve el valor del límite inferior de los valores de la columna o `None` si se desconoce el valor o si la columna no es continua.
`c.getUpperBound()`	Object	Devuelve el valor del límite superior de los valores de la columna o `None` si se desconoce el valor o si la columna no es continua.

Observe que la mayoría de los métodos de acceso a la información de una columna tienen métodos equivalentes definidos en el propio objeto DataModel. Por ejemplo, las dos sentencias siguientes son equivalentes:

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

Acceso a objetos generados

Ejecución de una corriente normalmente implica producir objetos de salida adicionales. Estos objetos adicionales pueden ser un nuevo modelo o un fragmento de la salida que proporciona información para utilizarla en las ejecuciones posteriores.

En el ejemplo siguiente, la ruta `druglearn.str` se utiliza de nuevo como punto de partida para la ruta. En este ejemplo, se ejecutan todos los nodos de la ruta y los resultados se almacenan en una lista. A continuación, el script crea un bucle por los resultados y se guarda cualquier salida del modo resultante de la ejecución como un archivo de modelo de IBM SPSS Modeler (`.gm`) y se exporta como PMML.

```
import modeler.api

ruta = 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)
```

La clase taskrunner proporciona un modo práctico de ejecutar diferentes tareas comunes. Los métodos que están disponibles en esta clase se resumen en la tabla siguiente.

Tabla 20. Métodos de la clase taskrunner para realizar tareas comunes

Método	Tipo devuelto	Descripción
`t.createStream(name, autoConnect, autoManage)`	Ruta	Crea y devuelve una nueva ruta. Tenga en cuenta que el código que debe crear las rutas de forma privada sin que las vea el usuario debe establecer el distintivo `autoManage` en `False`.
`t.exportDocumentToFile( documentOutput, filename, fileFormat)`	No es aplicable	Exporta la descripción de la ruta a un archivo utilizando el formato de archivo especificado.
`t.exportModelToFile(modelOutput, filename, fileFormat)`	No es aplicable	Exportsa el modelo a un archivo utilizando el formato de archivo especificado.
`t.exportStreamToFile(stream, filename, fileFormat)`	No es aplicable	Exporta la ruta a un archivo utilizando el formato de archivo especificado.
`t.insertNodeFromFile(filename, diagram)`	Nodo	Lee un nodo en el archivo especificado y lo devuelve insertándolo en el diagrama suministrado. Tenga en cuenta que lo pueden utilizar los objetos Nodo y Supernodo.
`t.openDocumentFromFile(filename, autoManage)`	DocumentOutput	Lee un nodo en el archivo especificado y lo devuelve.
`t.openModelFromFile(filename, autoManage)`	ModelOutput	Lee un modelo en el archivo especificado y lo devuelve.
`t.openStreamFromFile(filename, autoManage)`	Ruta	Lee una ruta en el archivo especificado y la devuelve.
`t.saveDocumentToFile( documentOutput, filename)`	No es aplicable	Guarda el documento en la ubicación de archivo especificada.
`t.saveModelToFile(modelOutput, filename)`	No es aplicable	Guarda el modelo en la ubicación de archivo especificada.
`t.saveStreamToFile(stream, filename)`	No es aplicable	Guarda la ruta en la ubicación de archivo especificada.

Manejo de errores

El lenguaje Python proporciona manejo de errores mediante el bloque de código `try...except`. Se puede utilizar en los scripts para capturar excepciones y manejar los problemas que podrían ocasionar la finalización del script.

En el script de ejemplo siguiente, se realiza un intento para recuperar un modelo desde IBM SPSS Collaboration and Deployment Services Repository. Esta operación puede hacer que se genere una excepción, por ejemplo, es posible que las credenciales de inicio de sesión en el repositorio no se hayan configurado correctamente o que la ruta del repositorio sea errónea. En el script, esto puede generar una excepción `ModelerException` (todas las excepciones que genera IBM SPSS Modeler se derivan de `modeler.api.ModelerException`).

```
import modeler.api

session = modeler.script.session()
try:
    tepo = 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()
```

Nota: Algunas operaciones de scripts pueden generar excepciones Java estándar, estas excepciones no se derivan de `ModelerException`. Para capturar estas excepciones, se puede utilizar un bloque except adicional que capture todas las excepciones Java, por ejemplo:

```
import modeler.api

session = modeler.script.session()
try:
    tepo = 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()
```

Parámetros de ruta, sesión y Supernodo

Los parámetros proporcionan una forma útil de pasar valores en el momento de la ejecución, en lugar de codificarlos directamente en un script. Los parámetros y sus valores se definen de la misma forma que las corrientes, es decir, como entradas de la tabla de los parámetros de una corriente o Supernodo o como parámetros de la línea de comandos. Las clases de Corriente y Supernodo implementan un conjunto de funciones definidas por el objeto ParameterProvider, como se muestra en la tabla siguiente. La sesión proporciona una llamada `getParameters()` que devuelve un objeto que define dichas funciones.

Tabla 21. Funciones definidas por el objeto ParameterProvider

Método	Tipo devuelto	Descripción
`p.parameterIterator()`	Iterator	Devuelve un iterador de nombres de parámetro para este objeto.

Tabla 21. Funciones definidas por el objeto ParameterProvider (continuación)

Método	Tipo devuelto	Descripción
`p.getParameterDefinition( parameterName)`	ParameterDefinition	Devuelve la definición de parámetro para el parámetro con el nombre especificado, o `None` si no existe tal parámetro en este proveedor. El resultado puede ser una instantánea de la definición en el momento en que el método se ha llamado y que no necesariamente refleja las modificaciones posteriores realizadas en el parámetro a través de este proveedor.
`p.getParameterLabel(parameterName)`	*cadena*	Devuelve la etiqueta del parámetro con nombre, o `None` si no existe tal parámetro.
`p.setParameterLabel(parameterName, label)`	No es aplicable	Establece la etiqueta del parámetro con nombre.
`p.getParameterStorage( parameterName)`	ParameterStorage	Devuelve el almacenamiento del parámetro con nombre, o `None` si no existe tal parámetro.
`p.setParameterStorage( parameterName, storage)`	No es aplicable	Establece el almacenamiento del parámetro con nombre.
`p.getParameterType(parameterName)`	ParameterType	Devuelve el tipo del parámetro con nombre, o `None` si no existe tal parámetro.
`p.setParameterType(parameterName, type)`	No es aplicable	Establece el tipo del parámetro con nombre.
`p.getParameterValue(parameterName)`	Object	Devuelve el valor del parámetro con nombre, o `None` si no existe tal parámetro.
`p.setParameterValue(parameterName, value)`	No es aplicable	Establece el valor del parámetro con nombre.

En el ejemplo siguiente, el script agrega algunos datos Telco para averiguar qué región tiene los datos de promedio de ingresos más bajos. A continuación, se establece un parámetro de ruta con esta región. Este parámetro de ruta se utiliza en un nodo Seleccionar para excluir dicha región de los datos, antes de que se cree un modelo de abandono en el resto.

El ejemplo es artificial porque el script genera el propio nodo Seleccionar y, por lo tanto, podría haber generado el valor correcto directamente en la expresión del nodo Seleccionar. Sin embargo, las rutas se suelen construir previamente, de modo que establecer los parámetros de este modo proporciona un ejemplo útil.

La primera parte del script de ejemplo crea el parámetro de ruta que contendrá la región con el promedio de ingresos más bajo. El script también crea los nodos de la rama de agregación y la rama de creación de modelos y los conecta.

```
import modeler.api

stream = modeler.script.stream()

# Inicializar un parámetro de ruta
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)
```

```
# Crear primero la rama de agregación para calcular el promedio de ingresos por región
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")
# Hacer referencia al parámetro de ruta en la selección
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)
```

El script de ejemplo crea la ruta siguiente.

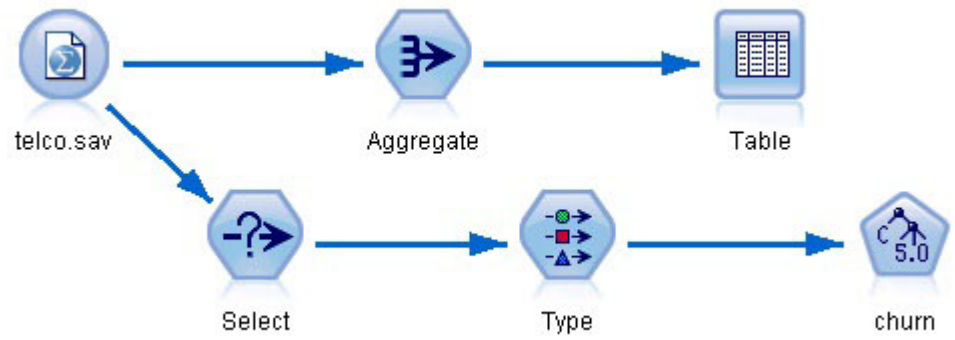

Figura 5. Ruta resultante del script de ejemplo

La parte siguiente del script de ejemplo ejecuta el nodo Tabla al final de la rama de agregación.

```
# Ejecutar primero el nodo Tabla
results = []
tablenode.run(results)
```

La parte siguiente del script de ejemplo accede a la salida de la tabla que ha generado la ejecución del nodo Tabla. A continuación, el script itera por las filas de la tabla, buscando la región con el promedio de ingresos más bajo.

```
# Ejecutar el nodo tabla para generar una sola tabla como salida
table = results[0]

# la salida de la tabla contiene un RowSet que permite acceder a valores como filas y columnas
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None
```

```
# Del modo que se ha definido el nodo, la primera columna
# contiene la región y la segunda contiene el promedio de ingresos
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
```

La parte siguiente del script utiliza la región con el promedio de ingresos más bajo para establecer el parámetro de ruta "LowestRegion" creado anteriormente. El script ejecuta el constructor de modelos excluyendo la región especificada de los datos de formación.

```
# Comprobar que se ha asignado un valor
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# Finalmente, ejecutar el constructor de modelos con el criterio de selección
c50node.run([])
```

A continuación, se muestra el script de ejemplo.

```
import modeler.api

stream = modeler.script.stream()

# Crear un parámetro de ruta
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)

# Crear primero la rama de agregación para calcular el promedio de ingresos por región
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")
# Hacer referencia al parámetro de ruta en la selección
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)


# Ejecutar primero el nodo Tabla
results = []
tablenode.run(results)

# Ejecutar el nodo tabla para generar una sola tabla como salida
table = results[0]
```

```
# la salida de la tabla contiene un RowSet que permite acceder a valores como filas y columnas
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None

# Del modo que se ha definido el nodo, la primera columna
# contiene la región y la segunda contiene el promedio de ingresos
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

# Comprobar que se ha asignado un valor
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# Finalmente, ejecutar el constructor de modelos con el criterio de selección
c50node.run([])
```

Valores globales

Los valores globales se utilizan para calcular diferentes estadísticas de resumen para los campos especificados. Se puede acceder a estos valores de resumen desde cualquier lugar de la ruta. Los valores globales son similares a los parámetros de ruta, ya que se puede acceder a los mismos por nombre a través de la ruta. Se diferencian de los parámetros de ruta en que los valores asociados se actualizan automáticamente cuando se ejecuta uno nodo Establecer globales, en lugar de asignarlos mediante script o desde la línea de comandos. Se accede a los valores globales de una ruta invocando el método `getGlobalValues()` de la ruta.

El objeto GlobalValues define las funciones que se muestran en la tabla siguiente.

Tabla 22. Funciones definidas por el objeto GlobalValues

Método	Tipo devuelto	Descripción
`g.fieldNameIterator()`	Iterador	Devuelve un iterador para cada nombre de campo con al menos un valor global.
`g.getValue(type, fieldName)`	Objeto	Devuelve el valor global para el tipo especificado y nombre de campo, o `None` si no se puede localizar ningún valor. Generalmente se espera que el valor devuelto sea un número, aunque en las funciones futuras se pueden devolver tipos de valores diferentes.
`g.getValues(fieldName)`	Mapa	Devuelve un mapa que contiene las entradas conocidas para el nombre de campo especificado o `None` si no hay entradas para el campo.

`GlobalValues.Type` define el tipo de estadísticas de resumen disponibles. Están disponibles las siguientes estadísticas de resumen:

- `MAX`: el valor máximo del campo.

- MEAN: el valor medio del campo.
- MIN: el valor mínimo del campo.
- STDDEV: la desviación estándar del campo.
- SUM: la suma de los valores del campo.

Por ejemplo, el script siguiente accede el valor medio del campo "income" que calcula un nodo Val. globales:

```
import modeler.api

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

Trabajar con varias rutas: Scripts autónomos

Para trabajar con varias rutas se debe utilizar un script autónomo. El script autónomo se puede editar y ejecutar dentro de la interfaz de usuario de IBM SPSS Modeler o se puede pasar como un parámetro de línea de comandos en modalidad de proceso por lotes.

El siguiente script autónomo abre dos rutas. Una de estas rutas crea un modelo y la segunda ruta traza la distribución de los valores predichos.

```
# Change to the appropriate location for your system
demosDir = "C:/Program Files/IBM/SPSS/Modeler/17/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()
```

Capítulo 5. Sugerencias sobre scripts

Esta sección proporciona una visión general de las técnicas y sugerencias para utilizar scripts, incluida la modificación de la ejecución de la ruta, la utilización de una contraseña codificada en un script y el acceso a objetos en IBM SPSS Collaboration and Deployment Services Repository.

Modificación de la ejecución de una ruta

Cuando se ejecuta una ruta, sus nodos terminales se ejecutan en un orden optimizado para la situación predeterminada. En algunos casos, es posible que prefiera un orden de ejecución diferente. Para modificar el orden de ejecución de una ruta, lleve a cabo los siguientes pasos desde la pestaña Ejecución del cuadro de diálogo Propiedades de ruta:

1. Comience con un script vacío.
2. Pulse en el botón **Añadir script predeterminado** de la barra de herramientas y añada el script de ruta predeterminado.
3. Cambie el orden de las instrucciones del script de ruta predeterminado por el orden en que desee que se ejecuten las instrucciones.

Recorrido en bucle de nodos

Puede utilizar un bucle `for` para recorrer en bucle todos los nodos de una ruta. Por ejemplo, los siguientes dos ejemplos de script recorren en bucle todos los nodos y cambia los nombres de campos de cualquier nodo Filtrar a mayúsculas.

Estos scripts pueden emplearse en cualquier ruta que tenga un nodo Filtrar, incluso si no hay campos filtrados. Simplemente, añada un nodo Filtrar que recorra todos los campos para cambiar todos los nombres de campo a mayúsculas.

```
# Alternative 1: using the data model nameIterator() function
ruta = 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
ruta = 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)
```

El script recorre en bucle los nodos de la ruta actual y comprueba cada nodo para ver si es un Filtro. Si es así, el script recorre en bucle cada campo del nodo y utiliza la función `field.upper()` o `field.getColumnName().upper()` para cambiar el nombre a mayúsculas.

Acceso a objetos en el IBM SPSS Collaboration and Deployment Services Repository

Si ha obtenido una licencia para IBM SPSS Collaboration and Deployment Services Repository, puede almacenar, recuperar, bloquear y desbloquear objetos del repositorio mediante comandos de script. El repositorio permite administrar el ciclo vital de los modelos de minería de datos y los objetos predictivos relacionados dentro del contexto de las aplicaciones, herramientas y soluciones empresariales.

Conexión con IBM SPSS Collaboration and Deployment Services Repository

Para acceder al repositorio, primero debe establecer una válida conexión con el mismo, mediante el menú Herramientas de la interfaz de usuario de IBM SPSS Modeler o mediante la línea de comandos. Consulte el tema "IBM SPSS Collaboration and Deployment Services Repository Argumentos de conexión" en la página 65 para obtener más información.

Almacenamiento y recuperación de objetos

En un script, los comandos `retrieve` y `store` permiten acceder a varios objetos, entre los que se incluyen rutas, modelos, resultados, nodos y proyectos. La sintaxis es la siguiente:

```
store object as REPOSITORY_PATH {label LABEL}
store object as URI [#l.label]

retrieve object REPOSITORY_PATH {label LABEL | version VERSION}
retrieve object URI [(#m.marker | #l.label)]
```

`RUTA_REPOSITORIO` proporciona la ubicación del objeto en el repositorio. La ruta debe estar entre comillas y utilizar barras inclinadas como delimitadores. No distingue entre mayúsculas y minúsculas.

```
store stream as "/folder_1/folder_2/mystream.str"
store model Drug as "/micarpeta/modelomedicamento"
store model Drug as "/micarpeta/modelomedicamento.gm" label "final"
store node MEDICAMENTO1n as "/samples/drug1ntypenode"
store project as "/CRISPDM/DrugExample.cpj"
store output "Data Audit of [6 fields]" as "/my folder/My Audit"
```

Si lo desea, se puede incluir una extensión como *.str* o *.gm* en el nombre de objeto, aunque no es necesario siempre que el nombre sea coherente. Por ejemplo, si un modelo se almacena sin una extensión, deberá recuperarse por el mismo nombre:

```
store model "/micarpeta/modelomedicamento"
retrieve model "/micarpeta/modelomedicamento"
```

frente a:

```
store model "/micarpeta/modelomedicamento.gm"
retrieve model "/micarpeta/modelomedicamento.gm" version "0:2005-10-12 14:15:41.281"
```

Tenga en cuenta que cuando recupera objetos, la versión más reciente del objeto siempre se devuelve a menos que especifique una versión o etiqueta. Al recuperar un objeto de nodo, el nodo se introduce automáticamente en la ruta actual. Al recuperar un objeto de ruta, debe utilizar un script autónomo. No puede recuperar un objeto de ruta de un script de ruta.

Bloqueo y desbloqueo de objetos

Puede bloquear un objeto desde un script para evitar que otros usuarios actualicen cualquiera de las versiones existentes o creen nuevas versiones. También puede desbloquear un objeto que haya bloqueado.

La sintaxis para bloquear y desbloquear un objeto es:

```
lock RUTA_REPOSITORIO
lock URI

unlock RUTA_REPOSITORIO
unlock URI
```

Como cuando se almacenan y se recuperan objetos, `RUTA_REPOSITORIO` le ofrece la ubicación del objeto en el repositorio. La ruta debe estar entre comillas y utilizar barras inclinadas como delimitadores. No distingue entre mayúsculas y minúsculas.

```
lock "/myfolder/Stream1.str"

unlock "/myfolder/Stream1.str"
```

Si lo prefiere, puede utilizar un identificador de recursos uniforme (URI) en lugar de una ruta de repositorio para proporcionar la ubicación del proyecto. El URI debe incluir el prefijo `spsscr:` y debe estar entre comillas. Sólo se puede utilizar barras inclinadas como delimitadores, y los espacios deben estar codificados. Es decir, utilizar `%20` en lugar de un espacio en la ruta. El URI no distingue entre mayúsculas y minúsculas. A continuación aparecen algunos ejemplos:

```
lock "spsscr:///myfolder/Stream1.str"

unlock "spsscr:///myfolder/Stream1.str"
```

Tenga en cuenta que el bloqueo de objetos se aplica a todas las versiones de un objeto: no puede bloquear o desbloquear versiones por separado.

Generación de una contraseña codificada

En algunos casos, puede que necesite incluir una contraseña en un script. Por ejemplo, es posible que desee acceder a un origen de datos protegido con contraseña. Las contraseñas codificadas pueden utilizarse en:

- Propiedades de los nodos para un origen de base de datos y nodos de resultado
- Argumentos de línea de comando para conectarse al servidor
- Propiedades de conexión con la base de datos almacenadas en un archivo *.par* (archivo de parámetro generado desde la pestaña Publicar de un nodo de exportación)

En la interfaz de usuario se proporciona una herramienta para generar contraseñas codificadas en función del algoritmo Blowfish (si desea obtener más información, consulte *http://www.schneier.com/blowfish.html*). Una vez codificada, puede copiar y almacenar la contraseña en archivos de script y argumentos de líneas de comando. La propiedad de nodo `epassword` utilizada para `databasenode` y `databaseexportnode` almacena la contraseña codificada.

1. Para generar una contraseña codificada, en el menú Herramientas seleccione:
   Codificar contraseña...
2. Especifique una contraseña en el cuadro de texto Contraseña.
3. Pulse en **Codificar** para generar una codificación aleatoria de la contraseña.
4. Pulse en el botón Copiar para copiar la contraseña codificada al Portapapeles.
5. Pegue la contraseña en el script o parámetro deseado.

Comprobación por script

Puede comprobar rápidamente la sintaxis de todos los tipos de scripts pulsando en el botón de comprobación de la barra de herramientas del cuadro de diálogo Script de ruta.

Figura 6. Iconos de barra de herramientas del script de ruta

En la comprobación por script se avisa de cualquier error que se detecte en el código y se sugieren recomendaciones de mejora. Para ver la línea con errores, pulse en los comentarios, en la mitad inferior del cuadro de diálogo. Los errores se señalan en rojo.

Scripts desde la línea de comandos

El uso scripts permite ejecutar operaciones típicamente desarrolladas en la interfaz de usuario. Simplemente especifique y ejecute una ruta independiente en la línea de comandos cuando ejecute IBM SPSS Modeler. Por ejemplo:

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

La marca `-script` carga el script especificado, mientras que la marca `-execute` ejecuta todos los comandos del archivo de script.

Compatibilidad con versiones anteriores

Los scripts creados en versiones anteriores de IBM SPSS Modeler deberían funcionar normalmente sin cambios en la versión actual. Sin embargo, los nuggets de modelos podrán ahora insertarse en la ruta automáticamente (es el comportamiento predeterminado) y podrán sustituir o complementar un nugget existente del tipo en la ruta. El que esto ocurra depende de la configuración de las opciones **Añadir modelo a ruta** y **Sustituir modelo anterior** (**Herramientas > Opciones > Opciones de usuario > Notificaciones**). Por ejemplo, es posible que tenga que modificar un script de una versión anterior en el que la sustitución del nugget se trate borrando el nugget existente e insertando uno nuevo.

Es posible que los scripts creados en esta versión no funcionen en versiones anteriores.

Si un script creó una liberación antigua utiliza un comando que se ha sustituido desde entonces (o desaprobado), la forma antigua se seguirá admitiendo, pero aparecerá un mensaje de advertencia. Por ejemplo, la palabra clave antigua `generated` se ha sustituido por `model`, y `clear generated` se ha sustituido por `clear generated palette`. Los scripts que utilizan las formas antiguas se seguirán ejecutando, pero se mostrará una advertencia.

Acceder a resultados de ejecución de la secuencia

Muchos nodos de IBM SPSS Modeler producen datos de salida tales como modelos, diagramas y datos tabulares. Muchos de estos datos de salida contienen valores útiles que pueden ser utilizados por scripts para guiar la ejecución subsiguiente. Estos valores se agrupan en contenedores de contenido (denominados simplemente contenedores) a los que se puede acceder utilizando etiquetas o identificadores que identifican cada contenedor. La forma en que se accede a estos valores depende del formato o "modelo de contenido" utilizado por el contenedor.

Por ejemplo, muchos resultados de modelo predictivo utilizan una variante de XML llamada PMML para representar información sobre el modelo, tal como qué campos utiliza un árbol de decisiones en cada bifurcación o cómo están conectadas las neuronas de una red neuronal y con qué intensidades. Los resultados del modelo que utilizan PMML proporcionan un modelo de contenido XML que se puede utilizar para acceder a esa información. Por ejemplo:

```
stream = modeler.script.stream()
# Suponga que la secuencia contiene un nodo generador de modelos C5.0
# y que el origen de datos, los predictores y los destinos ya se han
# configurado
modelbuilder = stream.findByType("c50", None)
```

```
results = []
modelbuilder.run(results)
modeloutput = results[0]

# Ahora que tenemos el objeto de salida del modelo C5.0, acceda al modelo
# de contenido pertinente
cm = modeloutput.getContentModel("PMML")

# El modelo de contenido PMML es un modelo de contenido genérico basado en XML
# que utiliza sintaxis del lenguaje XPath. Utilice ese modelo para encontrar
# los nombres de los campos de datos.
# La llamada devuelve una lista de series de caracteres correspondientes a los valores XPath
dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField", "name")
```

IBM SPSS Modeler es compatible con los modelos de contenido siguientes en los scripts:

- **Modelo de contenido de tabla**: proporciona acceso a datos tabulares simples representados como filas y columnas
- **Modelo de contenido XML**: proporciona acceso a contenido almacenado en formato XML
- **Modelo de contenido JSON**: proporciona acceso a contenido almacenado en formato JSON
- **Modelo de contenido de estadísticas de columna**: proporciona acceso a estadísticas de resumen sobre un campo determinado
- **Modelo de contenido de estadísticas de columna por pares**: proporciona acceso a estadísticas de resumen entre dos campos o valores entre dos campos separados

Modelo de contenido de tabla

El modelo de contenido de tabla proporciona un modelo sencillo para acceder a los datos simples de fila y columna. Los valores en una columna determinada deben tener todos el mismo tipo de almacenamiento (por ejemplo, series o enteros).

API

Tabla 23. API

Devolver	Método	Descripción
`int`	`getRowCount()`	Devuelve el número de filas en esta tabla.
`int`	`getColumnCount()`	Devuelve el número de columnas en esta tabla.
`Cadena`	`getColumnName(int columnIndex)`	Devuelve el nombre de la columna en el índice de columna especificado. El índice de columna comienza en el 0.
`StorageType`	`getStorageType(int columnIndex)`	Devuelve el tipo de almacenamiento de la columna en el índice especificado. El índice de columna comienza en el 0.
`Object`	`getValueAt(int rowIndex, int columnIndex)`	Devuelve el valor en los índices de fila y columna especificados. Los índices de fila y columna comienzan en el 0.
`void`	`reset()`	Desecha cualquier almacenamiento interno asociado con este modelo de contenido.

Nodos y salidas

Esta tabla lista los nodos que crean salidas que incluyen este tipo de modelo de contenido.

Tabla 24. Nodos y salidas

Nombre de nodo	Nombre de resultado	ID de contenedor
tabla	tabla	"tabla"

Script de ejemplo

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

# Establecer el nodo de importación de archivo de variable
varfilenode = stream.createAt("variablefile", "DRUG Data", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")

# Crear el nodo Agregar y conectarlo al nodo de archivo de variable
aggregatenode = stream.createAt("aggregate", "Aggregate", 192, 96)
stream.link(varfilenode, aggregatenode)

# Configurar el nodo Agregar
aggregatenode.setPropertyValue("keys", ["Drug"])
aggregatenode.setKeyedPropertyValue("aggregates", "Age", ["Min", "Max"])
aggregatenode.setKeyedPropertyValue("aggregates", "Na", ["Mean", "SDev"])

# Crear el nodo de salida de tabla y conectarlo al nodo Agregar
tablenode = stream.createAt("table", "Table", 288, 96)
stream.link(aggregatenode, tablenode)

# Ejecutar el nodo de tabla y capturar el objeto de salida de resultado de la tabla
results = []
tablenode.run(results)
tableoutput = results[0]

# Acceder al modelo de contenido del resultado de la tabla
tablecontent = tableoutput.getContentModel("table")

# Para cada columna, imprimir el nombre de columna, tipo y la primera fila
# de valores del contenido de la tabla
col = 0
while col < tablecontent.getColumnCount():
     print tablecontent.getColumnName(col), \
     tablecontent.getStorageType(col), \
     tablecontent.getValueAt(0, col)
     col = col + 1
```

La salida en la pestaña Depuración de scripts tendrá un aspecto similar a este:

```
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
```

Modelo de contenido XML

El modelo de contenido XML proporciona acceso a contenido basado en XML.

El modelo de contenido XML permite acceder a componentes utilizando expresiones XPath. Las expresiones XPath son series que definen qué elementos o atributos son necesarios para el solicitante. El modelo de contenido XML hace que sea transparente la creación de diversos objetos y expresiones de compilación que normalmente son necesarios para el soporte de XPath. Esto hace que sea más sencillo hacer llamadas desde scripts Python.

El modelo de contenido XML incluye una función que devuelve el documento XML como serie de caracteres. Esto permite que los usuarios del script Python utilicen su biblioteca preferida de Python para analizar XML.

API

Tabla 25. API

Resultado devuelto	Método	Descripción
`Serie de caracteres`	`getXMLAsString()`	Devuelve el XML en forma de serie de caracteres.
`Número`	`getNumericValue(String xpath)`	Devuelve un resultado de tipo numérico al analizar la vía de acceso (por ejemplo, contar el número de elementos que coinciden con la expresión de vía de acceso).
`Booleano`	`getBooleanValue(String xpath)`	Devuelve un resultado de tipo booleano al evaluar la expresión de vía de acceso especificada.
`Serie de caracteres`	`getStringValue(String xpath, String attribute)`	Devuelve el valor de atributo o valor de nodo XML que coincide con la vía de acceso especificada.
`Lista de series de caracteres`	`getStringValues(String xpath, String attribute)`	Devuelve una lista de todos los valores de atributo o valores de nodo XML que coinciden con la vía de acceso especificada.
`Lista de series de caracteres`	`getValuesList(String xpath, <Lista de series de caracteres> attributes, boolean includeValue)`	Devuelve una lista de todos los valores de atributo que coinciden con la vía de acceso especificada junto con el valor de nodo XML si es necesario.
`Hash table (key:string, value:list of string)`	`getValuesMap(String xpath, String keyAttribute, <Lista de series de caracteres> attributes, boolean includeValue)`	Devuelve una tabla hash que utiliza el atributo de clave o valor de nodo XML como clave y la lista de valores de atributo especificados como valores de la tabla.
`Booleano`	`isNamespaceAware()`	Indica si los analizadores XML deben tener en cuenta los espacios de nombres. El valor predeterminado es `False`.
`void`	`setNamespaceAware(boolean value)`	Establece si los analizadores XML deben tener en cuenta los espacios de nombres. Esto también invoca a `reset()` para asegurarse de que los cambios sean captados por llamadas posteriores.

Tabla 25. API (continuación)

Resultado devuelto	Método	Descripción
`void`	`reset()`	Desecha cualquier almacenamiento interno asociado con el modelo de contenido (por ejemplo, un objeto DOM almacenado en memoria caché).

Nodos y datos de salida

Esta tabla lista los nodos que generan datos de salida y que incluyen este tipo de modelo de contenido.

Tabla 26. Nodos y datos de salida

Nombre de nodo	Nombre del resultado	ID de contenedor
`La mayoría de constructores de modelos`	`La mayoría de modelos generados`	`"PMML"`
`"autodataprep"`	`n/d`	`"PMML"`

Script de ejemplo

El código del script Python para acceder al contenido puede tener este aspecto:

```
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")
```

Modelo de contenido JSON

El modelo de contenido JSON se utiliza para proporcionar soporte para contenido con formato JSON. Esto proporciona una API básica para permitir que los solicitantes extraigan valores bajo la asunción de que saben qué valores se deben acceder.

API

Tabla 27. API

Resultado devuelto	Método	Descripción
`Serie de caracteres`	`getJSONAsString()`	Devuelve el contenido JSON como serie de caracteres.
`Objeto`	`getObjectAt(<Lista de objetos> path, JSONArtifact artifact) throws Exception`	Devuelve el objeto situado en la vía de acceso especificada. El artefacto raíz proporcionado puede ser nulo, en cuyo caso se utiliza la raíz del contenido. El valor devuelto puede ser una serie literal, un entero, un número real, un valor booleano o un artefacto JSON (ya sea un objeto JSON o una matriz JSON).

Tabla 27. API (continuación)

Resultado devuelto	Método	Descripción
`Tabla hash (key:object, value:object>`	`getChildValuesAt(<Lista de objetos> path, JSONArtifact artifact) throws Exception`	Devuelve los valores hijo de la vía de acceso especificada si la vía conduce a un objeto JSON, o devuelve nulo en caso contrario. Las claves de la tabla son series, mientras que el valor asociado puede ser una serie literal, un entero, un número real, un valor booleano o un artefacto JSON (ya sea un objeto JSON o una matriz JSON).
`Lista de objetos`	`getChildrenAt(<Lista de objetos> path path, JSONArtifact artifact) throws Exception`	Devuelve la lista de objetos situados en la vía de acceso especificada si la vía conduce a una matriz JSON, o devuelve nulo en caso contrario. Los valores devueltos pueden ser una serie literal, un entero, un número real, un valor booleano o un artefacto JSON (ya sea un objeto JSON o una matriz JSON).
`void`	`reset()`	Desecha cualquier almacenamiento interno asociado con el modelo de contenido (por ejemplo, un objeto DOM almacenado en memoria caché).

Script de ejemplo

Si existe un nodo generador de salida que crea salida en formato JSON, se puede utilizar lo siguiente para acceder a información sobre un conjunto de libros:

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

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

# Como alternativa, obtenga el objeto de libro y utilícelo como raíz
# para entradas subsiguientes
book = cm.getObjectAt(["books", "ISIN123456"], None)
bookTitle = cm.getObjectAt(["title"], book)

# Obtener todos los valores hijos para un libro determinado
bookInfo = cm.getChildValuesAt(["books", "ISIN123456"], None)

# Obtener la tercera entrada de libro. Se supone el valor de nivel superior "books"
# contiene una matriz JSON que se puede indexar
bookInfo = cm.getObjectAt(["books", 2], None)

# Obtener una lista de todas las entradas hijas
allBooks = cm.getChildrenAt(["books"], None)
```

Modelo de contenido de estadísticas de columna y modelo de contenido de estadísticas por pares

El modelo de contenido de estadísticas de columna proporciona acceso a estadísticas que se pueden calcular para cada campo (estadísticas univariables). El modelo de contenido de estadísticas por pares proporciona acceso a estadísticas que se pueden calcular para pares de campos o pares de valores de un campo.

Las medidas estadísticas posibles son:

- `Count`
- `UniqueCount`
- `ValidCount`
- `Mean`
- `Sum`
- `Min`
- `Max`
- `Range`
- `Variance`
- `StandardDeviation`
- `StandardErrorOfMean`
- `Skewness`
- `SkewnessStandardError`
- `Kurtosis`
- `KurtosisStandardError`
- `Median`
- `Mode`
- `Pearson`
- `Covariance`
- `TTest`
- `FTest`

Algunos valores sólo son adecuados para estadísticas de una sola columna, mientras que otros sólo son adecuados para estadísticas por pares.

Los nodos que generan estadísticas son los siguientes:

- El **nodo Estadísticas** produce estadísticas de columna y puede producir estadísticas por pares cuando se especifican campos de correlación
- El **nodo Auditoría de datos** produce estadísticas de columna y puede producir estadísticas por pares cuando se especifica un campo de preformato.
- El **nodo Medias** produce estadísticas por pares cuando compara pares de campos o cuando compara valores de un campo con otros resúmenes de campo.

Qué modelos de contenido y estadísticas se pueden utilizar depende de las prestaciones del nodo en cuestión y de los valores contenidos en ese nodo.

La API ColumnStatsContentModel

Tabla 28. La API ColumnStatsContentModel.

Resultado devuelto	Método	Descripción
`List<StatisticType>`	`getAvailableStatistics()`	Devuelve las estadísticas disponibles en este modelo. No todos los campos tendrán necesariamente valores para todas las estadísticas.
`List<String>`	`getAvailableColumns()`	Devuelve los nombres de columna para los que se han calculado estadísticas.
`Número`	`getStatistic(String column, StatisticType statistic)`	Devuelve los valores estadísticos asociados a la columna.
`void`	`reset()`	Desecha cualquier almacenamiento interno asociado con el modelo de contenido.

La API PairwiseStatsContentModel

Tabla 29. La API PairwiseStatsContentModel.

Resultado devuelto	Método	Descripción
`List<StatisticType>`	`getAvailableStatistics()`	Devuelve las estadísticas disponibles en este modelo. No todos los campos tendrán necesariamente valores para todas las estadísticas.
`List<String>`	`getAvailablePrimaryColumns()`	Devuelve los nombres de columna primaria para los que se han calculado estadísticas.
`List<Object>`	`getAvailablePrimaryValues()`	Devuelve los valores de la columna primaria para la que se han calculado estadísticas.
`List<String>`	`getAvailableSecondaryColumns()`	Devuelve los nombres de columna secundaria para los que se han calculado estadísticas.
`Número`	`getStatistic(String primaryColumn, String secondaryColumn, StatisticType statistic)`	Devuelve los valores estadísticos asociados a las columnas.
`Número`	`getStatistic(String primaryColumn, Object primaryValue, String secondaryColumn, StatisticType statistic)`	Devuelve los valores estadísticos asociados al valor de la columna primaria y la columna secundaria.
`void`	`reset()`	Desecha cualquier almacenamiento interno asociado con el modelo de contenido.

Nodos y datos de salida

Esta tabla lista los nodos que generan datos de salida y que incluyen este tipo de modelo de contenido.

Tabla 30. Nodos y datos de salida.

Nombre de nodo	Nombre del resultado	ID de contenedor	Notas
"means" (nodo Medias)	"means"	"columnStatistics"	
"means" (nodo Medias)	"means"	"pairwiseStatistics"	
"dataaudit" (nodo Auditoría de datos)	"means"	"columnStatistics"	
"statistics" (nodo Estadísticas)	"statistics"	"columnStatistics"	Sólo se genera cuando se examinan campos determinados.
"statistics" (nodo Estadísticas)	"statistics"	"pairwiseStatistics"	Sólo se genera cuando se correlacionan campos.

Script de ejemplo

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

# Definir los datos de entrada
varfile = stream.createAt("variablefile", "File", 96, 96)
varfile.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")

# Crear el nodo de estadísticas. Este nodo puede producir
# estadísticas de columna y estadísticas por pares
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
```

Capítulo 6. Argumentos de la línea de comandos

Invocación del software

Puede utilizar la línea de comandos del sistema operativo para ejecutar IBM SPSS Modeler de la siguiente manera:

1. En un ordenador en el que se haya instalado IBM SPSS Modeler, abra una ventana de DOS o del indicador de comandos.
2. Para iniciar la interfaz de IBM SPSS Modeler en modo interactivo, escriba el comando `clementine` seguido de los argumentos necesarios; por ejemplo:

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

Los argumentos disponibles (modificadores) permiten conectar con un servidor, cargar rutas, ejecutar scripts o especificar otros parámetros, según sea necesario.

Uso de argumentos en la línea de comandos

Se pueden añadir argumentos de línea de comandos (también denominados *marcas*) al comando inicial `modelerclient` o para alterar la invocación de IBM SPSS Modeler.

Hay varios tipos de argumentos de línea de comandos disponibles que se describen más adelante en esta sección.

Tabla 31. Tipos de argumentos de línea de comandos.

Tipo de argumento	Dónde se describe
Argumentos del sistema	Consulte el tema "Argumentos del sistema" en la página 62 para obtener más información.
Argumentos de parámetros	Consulte el tema "Argumentos de parámetros" en la página 63 para obtener más información.
Argumentos de conexión del servidor	Consulte el tema "Argumentos de conexión con el servidor" en la página 64 para obtener más información.
Argumentos de conexión de IBM SPSS Collaboration and Deployment Services Repository	Consulte el tema "IBM SPSS Collaboration and Deployment Services Repository Argumentos de conexión" en la página 65 para obtener más información.
Argumentos de conexión de IBM SPSS Analytic Server	Consulte el tema "IBM SPSS Analytic Server Argumentos de conexión" en la página 65 para obtener más información.

Por ejemplo, se pueden utilizar las marcas `-server`, `-stream` y `-execute` para conectar con un servidor y, a continuación, cargar y ejecutar una ruta, de la siguiente forma:

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

Tenga en cuenta que al ejecutarse en una instalación cliente local, no se necesitan argumentos de conexión con el servidor.

Los valores de parámetros que contienen espacios se pueden poner entre comillas dobles, por ejemplo:

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

También puede ejecutar scripts y estados de IBM SPSS Modeler de esta forma, utilizando las marcas `-state` y `-script` respectivamente.

Nota: Si utiliza un parámetro estructurado en un mandato, delante de las comillas debe poner barras invertidas. Así evitará que se quiten las comillas durante la interpretación de la serie.

Depuración de argumentos de la línea de comandos

Para depurar una línea de comandos, utilice el comando `modelerclient` para iniciar IBM SPSS Modeler con los argumentos deseados. Esto permite comprobar que los comandos se ejecutarán como se espera. También puede confirmar los valores de cualquier parámetro pasado desde la línea de comandos en el cuadro de diálogo Parámetros de sesión (menú Herramientas, Definir parámetros de sesión).

Argumentos del sistema

En la siguiente tabla se describen los argumentos del sistema disponibles para la invocación de la línea de comandos de la interfaz de usuario.

Tabla 32. Argumentos del sistema

Argumento	Comportamiento/Descripción
`@` <archivo de comandos>	El carácter @ seguido de un nombre de archivo especifica una lista de comandos. Cuando `modelerclient` o encuentra un argumento que comienza por @, opera en los comandos de este archivo como si hubieran estado en la línea de comandos. Consulte el tema "Combinación de varios argumentos" en la página 66 para obtener más información.
`-directory` <dir>	Define el directorio de trabajo predeterminado. En el modo local, este directorio se utiliza tanto para datos como para resultados. Ejemplo: `-directory c:/` o `-directory c:\\`
`-server_directory` <dir>	Define el directorio de servidor predeterminado para datos. El directorio de trabajo, especificado con la marca `-directory`, se utiliza para resultados.
`-execute`	Después del inicio, ejecuta cualquier ruta, estado o script que se haya cargado en el inicio. Si se carga un script además de una ruta o un estado, el script se ejecutará solo.
`-stream` <ruta>	Carga en el inicio la ruta especificada. Se pueden especificar varias rutas, pero la última se definirá como la actual.
`-script` <script>	Carga en el inicio el script autónomo especificado. Se puede especificar además de una ruta o un estado, tal y como se describe a continuación, pero sólo se puede cargar un único script en el inicio.
`-model` <modelo>	En el inicio, carga el modelo generado (archivo de formato *.gm*) especificado.
`-state` <estado>	Carga en el inicio el estado especificado guardado.
`-project` <proyecto>	Carga el proyecto especificado. Sólo se puede cargar un único proyecto en el inicio.
`-output` <resultado>	Carga en el inicio el objeto de resultados guardado (archivo de formato *.cou*).
`-help`	Muestra una lista de argumentos de la línea de comandos. Cuando se especifica esta opción, todos los demás argumentos se ignoran y se muestra la pantalla Ayuda.
`-P` <nombre>=<valor>	Se utiliza para definir un parámetro de inicio. También se puede utilizar para definir propiedades de nodos (parámetros de intervalo).

Note: los directorios predeterminados también se pueden definir en la interfaz de usuario. Para acceder a las opciones en el menú Archivo, seleccione **Definir directorio** o **Definir directorio de servidor**.

Carga de varios archivos

Desde la línea de comandos puede cargar varias rutas, estados y resultados en el inicio repitiendo el argumento relevante para cada objeto cargado. Por ejemplo, para cargar y ejecutar dos rutas llamadas *report.str* y *train.str*, utilizaría el siguiente comando:

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

Carga de objetos desde IBM SPSS Collaboration and Deployment Services Repository

Dado que puede cargar determinados objetos de un archivo o desde IBM SPSS Collaboration and Deployment Services Repository (si dispone de licencia), el prefijo de nombre de archivo `spsscr:` y, si lo desea, `file:` (para objetos en disco) indica a IBM SPSS Modeler donde buscar el objeto. El prefijo funciona con las siguientes marcas:

- `-stream`
- `-script`
- `-output`
- `-model`
- `-project`

Puede utilizar el prefijo para crear un URI que especifique la ubicación del objeto, por ejemplo, `-stream "spsscr:///folder_1/scoring_stream.str"`. La presencia del prefijo `spsscr:` requiere que se especifique una conexión válida a IBM SPSS Collaboration and Deployment Services Repository en el mismo comando. Así, por ejemplo, el comando completo sería:

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

Recuerde que en la línea de comandos *debe* utilizar un URI. El `REPOSITORY_PATH` más simple no se admite. (Sólo funciona en scripts.) Para obtener más detalles sobre los URI para objetos en el IBM SPSS Collaboration and Deployment Services Repository, consulte el tema “Acceso a objetos en el IBM SPSS Collaboration and Deployment Services Repository” en la página 50.

Argumentos de parámetros

Los parámetros se pueden utilizar como marcas durante la ejecución de la línea de comandos de IBM SPSS Modeler. En los argumentos de la línea de comandos, la marca `-P` se utiliza para denotar un parámetro del tipo `-P` *<nombre>=<valor>*.

Los parámetros pueden ser:

- **Parámetros simples** (o parámetros utilizados directamente en expresiones CLEM).
- **Parámetros de intervalo**, también denominados **propiedades de nodos**. Estos parámetros se utilizan para modificar la configuración de los nodos en la ruta. Consulte el tema “Conceptos básicos de las propiedades de nodos” en la página 69 para obtener más información.
- **Parámetros de la línea de comandos**, que son parámetros utilizados para alterar la invocación de IBM SPSS Modeler.

Por ejemplo, puede proporcionar contraseñas y nombres de usuario de orígenes de datos como una marca de la línea de comandos, por ejemplo:

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

El formato es el mismo que el del parámetro `datasource` de la propiedad de nodo `databasenode`. Para obtener más información, consulte: “Propiedades de databasenode” en la página 81.

Nota: Si el nodo tiene nombre, debe encerrar el nombre del nodo entre comillas dobles y utilizar una barra inclinada invertida como carácter de escape antes de las comillas. Por ejemplo, si el nodo de origen de datos del ejemplo anterior tiene el nombre *Source_ABC*, la entrada sería la siguiente:

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

También se requiere una barra inclinada invertida delante de las comillas que identifican un parámetro estructurado, como en el siguiente ejemplo de origen de datos de TM1:

```
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%\"}
```

Argumentos de conexión con el servidor

La marca `-server` indica a IBM SPSS Modeler que debe conectar con un servidor público, y las marcas `-hostname`, `-use_ssl`, `-port`, `-username`, `-password` y `-domain` se utilizan para indicar a IBM SPSS Modeler cómo conectar con el servidor público. Si no se especifica ningún argumento -server, se utilizará el servidor predeterminado o local.

Ejemplos

Para conectarse con un servidor público:

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

Para conectarse con un clúster de servidores:

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

Tenga en cuenta que para conectarse a un clúster de servidores necesita Coordinator of Processes a través de IBM SPSS Collaboration and Deployment Services, de modo que debe utilizar el argumento `-cluster` junto con las opciones de conexión de repositorio (`spsscr_*`). Consulte el tema "IBM SPSS Collaboration and Deployment Services Repository Argumentos de conexión" en la página 65 para obtener más información.

Tabla 33. Argumentos de conexión del servidor.

Argumento	Comportamiento/Descripción
`-server`	Ejecuta IBM SPSS Modeler en el modo servidor, conectando con un servidor público utilizando las marcas `-hostname`, `-port`, `-username`, `-password` y `-domain`.
`-hostname` <nombre>	Nombre de host del equipo servidor. Disponible en el modo servidor solamente.
`-use_ssl`	Especifica que la conexión debería utilizar SSL (secure socket layer). La marca es opcional, el parámetro predeterminado *no* utiliza SSL.
`-port` <número>	Número de puerto del servidor especificado. Disponible en el modo servidor solamente.
`-cluster` <nombre>	Especifica una conexión a un clúster de servidores en lugar de un servidor especificado; este argumento es una alternativa a los argumentos `hostname`, `port` y `use_ssl`. El nombre es el del clúster o un URI exclusivo que identifica el clúster en IBM SPSS Collaboration and Deployment Services Repository. Coordinator of Processes gestiona el clúster de servidores a través de IBM SPSS Collaboration and Deployment Services. Consulte el tema "IBM SPSS Collaboration and Deployment Services Repository Argumentos de conexión" en la página 65 para obtener más información.

Tabla 33. Argumentos de conexión del servidor (continuación).

Argumento	Comportamiento/Descripción
`-username` <nombre>	Nombre de usuario con el que iniciar sesión en el servidor. Disponible en el modo servidor solamente.
`-password` <contraseña>	Contraseña con la que iniciar sesión en el servidor. Disponible en el modo servidor solamente. *Note*: si no se utiliza el argumento `-password`, se le solicitará una contraseña.
`-epassword` <cadena de contraseña codificada>	Contraseña codificada con la que iniciar sesión en el servidor. Disponible en el modo servidor solamente. *Note*: Las contraseñas codificadas se pueden generar desde el menú Herramientas de la aplicación IBM SPSS Modeler.
`-domain` <nombre>	Dominio utilizado para iniciar sesión en el servidor. Disponible en el modo servidor solamente.
`-P` <nombre>=<valor>	Se utiliza para definir un parámetro de inicio. También se puede utilizar para definir propiedades de nodos (parámetros de intervalo).

IBM SPSS Collaboration and Deployment Services Repository Argumentos de conexión

Si desea almacenar o recuperar objetos de IBM SPSS Collaboration and Deployment Services a través de la línea de comandos, debe especificar una conexión válida con IBM SPSS Collaboration and Deployment Services Repository. Por ejemplo:

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

La siguiente tabla enumera los argumentos que pueden utilizarse para configurar la conexión.

Tabla 34. Argumentos de conexión de IBM SPSS Collaboration and Deployment Services Repository

Argumento	Comportamiento/Descripción
`-spsscr_hostname` <nombre del host o dirección IP>	El nombre del host o la dirección IP del servidor en que se ha instalado IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_port` <número>	Número de puerto en el que el IBM SPSS Collaboration and Deployment Services Repository acepta las conexiones (normalmente 8080 de forma predeterminada).
`-spsscr_use_ssl`	Especifica que la conexión debería utilizar SSL (secure socket layer). La marca es opcional, el parámetro predeterminado *no* utiliza SSL.
`-spsscr_username` <nombre>	Nombre de usuario con el que iniciar sesión en el IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_password` <contraseña>	Contraseña con la que iniciar sesión en el IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_epassword` <contraseña codificada>	Contraseña cifrada con la que iniciar sesión en el IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_domain` <nombre>	Dominio utilizado para iniciar sesión en el IBM SPSS Collaboration and Deployment Services Repository. Esta marca es opcional, no la utilice a menos que inicie sesión mediante LDAP o Active Directory.

IBM SPSS Analytic Server Argumentos de conexión

Si desea almacenar o recuperar objetos de IBM SPSS Analytic Server mediante la línea de mandatos, debe indicar una conexión válida con IBM SPSS Analytic Server.

Nota: La ubicación de Analytic Server se obtiene a partir de SPSS Modeler Server y no se puede cambiar en el cliente.

La siguiente tabla enumera los argumentos que pueden utilizarse para configurar la conexión.

Tabla 35. Argumentos de conexión de IBM SPSS Analytic Server

Argumento	Comportamiento/Descripción
`-analytic_server_username`	Nombre de usuario con el que iniciar sesión en IBM SPSS Analytic Server.
`-analytic_server_password`	La contraseña con la que se inicia sesión en IBM SPSS Analytic Server.
`-analytic_server_epassword`	La contraseña codificada con la que se inicia sesión en IBM SPSS Analytic Server.
`-analytic_server_credential`	Las credenciales utilizadas para iniciar sesión en IBM SPSS Analytic Server.

Combinación de varios argumentos

Es posible combinar varios argumentos en un único archivo de comandos especificado en la invocación utilizando el símbolo @ seguido del nombre de archivo. De este modo podrá acortar la invocación de la línea de comandos y superar cualquier limitación del sistema operativo en la longitud del comando. Por ejemplo, el siguiente comando de inicio utiliza todos los argumentos especificados en el archivo de referencia `<nombre de archivo de comandos>`.

```
modelerclient @<commandFileName>
```

Ponga el nombre del archivo y la ruta del archivo de comandos entre comillas si hay que incluir espacios, de la siguiente forma:

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

El archivo de comandos puede contener todos los argumentos especificados previamente a nivel individual en el inicio. Por ejemplo:

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

Cuando escriba y referencie archivos de comandos, asegúrese de cumplir estas restricciones:

- Utilice sólo un comando por línea.
- No incruste un argumento `@archivo de comandos` en un archivo de comandos.

Capítulo 7. Referencia de propiedades

Conceptos básicos de referencia de propiedades

Puede especificar varias propiedades diferentes para los nodos, rutas, supernodos y proyectos. Algunas propiedades son comunes a todos los nodos, como el nombre, la anotación y la información sobre herramientas, mientras que otras son específicas para determinados tipos de nodos. Otras propiedades hacen referencia a operaciones de rutas de alto nivel, como el comportamiento del Supernodo o el almacenamiento en caché. Se puede acceder a las propiedades a través de la interfaz de usuario estándar (por ejemplo, al abrir un cuadro de diálogo para editar opciones para un nodo) y se pueden utilizar también de varias otras formas.

- Las propiedades se pueden modificar a través de los scripts, como se describe en esta sección. Si desea obtener más información, consulte "Sintaxis de las propiedades".
- Las propiedades de los nodos se pueden utilizar en los parámetros de Supernodo.
- Asimismo, las propiedades de los nodos se pueden utilizar como parte de una opción de línea de comandos (mediante la marca `-P`) al iniciar IBM SPSS Modeler.

En el contexto de los scripts de IBM SPSS Modeler, las propiedades de nodos y rutas se suelen llamar **parámetros de intervalo**. En este manual, se denominan propiedades de nodos y rutas.

Si desea obtener más información sobre el lenguaje de scripts, consulte Lenguaje de scripts.

Sintaxis de las propiedades

Las propiedades se pueden establecer con la sintaxis siguiente

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

o:

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

El valor de propiedades se puede recuperar usando la sintaxis siguiente:

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

o:

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

donde `OBJECT` es un nodo o salida, `PROPERTY` es el nombre de la propiedad de nodo al que la expresión se refiere, y `KEY` es el valor de la clave para las propiedades clave. Por ejemplo, la siguiente sintaxis se utiliza para buscar el nodo de filtro y, a continuación, establecer el valor predeterminado para incluir todos los campos y filtrar el campo `Age` en los datos en sentido descendente:

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

Todos los nodos utilizados en IBM SPSS Modeler pueden encontrarse utilizando la función `findByType(TYPE, LABEL)` de la ruta. Al menos debe especificarse `TYPE` o `LABEL`.

Propiedades estructuradas

Hay dos formas en las que los scripts utilizan propiedades estructuradas para mejorar la claridad durante el análisis:

- Otorgando estructura a los nombres de las propiedades para los nodos complejos, como Tipo, Filtro o Equilibrar.

- Proporcionando un formato para especificar varias propiedades a la vez.

Estructuración para las interfaces complejas

Los scripts para los nodos con tablas y otras interfaces complejas, como, por ejemplo, los nodos Tipo, Filtro o Equilibrar, deben seguir una estructura determinada para realizar el análisis correctamente. Estas propiedades necesitan un nombre más complejo que el de un solo identificador. Este nombre se denomina clave. Por ejemplo, en un nodo Filtrar, cada campo disponible (en la parte superior) se activa o desactiva. Para poder consultar esta información, el nodo Filtrar almacena un elemento de información por campo (independientemente de que el campo sea verdadero o falso). Esta propiedad debe tener (o se le ha dado) el valor `True` o `False`. Supongamos que un nodo Filtrar denominado `minodo` tiene (en la parte superior) un campo denominado `Edad`. Para desactivar esto, establezca la propiedad `include`, con la clave `Age`, en el valor `False`, del modo siguiente:

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

Estructuración para definir varias propiedades

Si hay muchos nodos, puede asignar más de una propiedad de nodo o ruta al mismo tiempo. Esto se denomina **comando de conjunto múltiple** o **bloque de conjuntos**.

En algunos casos, una propiedad estructurada puede ser bastante compleja. A continuación se muestra un ejemplo:

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

Otra ventaja de las propiedades estructuradas es la capacidad de definir varias propiedades en un nodo antes de que éste sea estable. De forma predeterminada, un conjunto múltiple define todas las propiedades del bloque antes de realizar una acción basada en una configuración de propiedades individuales. Por ejemplo, al definir un nodo Archivo fijo, el uso de dos pasos para definir las propiedades del campo daría lugar a errores porque el nodo no será constante hasta que las dos configuraciones sean válidas. La definición de las propiedades como un conjunto múltiple salva este problema al definir ambas propiedades antes de actualizar el modelo de datos.

Abreviaturas

Las abreviaturas estándar se utilizan en la sintaxis para las propiedades de nodos. El aprendizaje de las abreviaturas le ayudará en la creación de scripts.

Tabla 36. Abreviaturas estándar utilizadas en toda la sintaxis

Abreviatura	Significado
abs	Valor absoluto
lon	Longitud
min	Mínimo
máx	Máximo
correl	Correlation
covar	Covariance
núm	Número o numérico
pct	Porcentaje
transp	Transparencia
xval	Validación cruzada
var	Varianza o variable (en nodos de origen)

Ejemplos de propiedades de nodos y rutas

Las propiedades de nodos y rutas se pueden utilizar de varias formas con IBM SPSS Modeler. Normalmente se utilizan como parte de un script, bien un **script autónomo**, utilizado para automatizar rutas u operaciones o un **script de ruta**, utilizado para automatizar procesos en una sola ruta. Los parámetros de nodo se pueden especificar también utilizando las propiedades para los nodos del Supernodo. En el nivel más básico, las propiedades se pueden utilizar también como una opción de línea de comandos para iniciar IBM SPSS Modeler. Si utiliza el argumento `-p` como parte de la invocación de la línea de comandos, podrá utilizar una propiedad de ruta para cambiar una configuración de la ruta.

Tabla 37. Ejemplos de las propiedades node y stream

Propiedad	Significado
`s.max_size`	Hace referencia a la propiedad `max_size` del nodo denominado `s`.
`s:samplenode.max_size`	Hace referencia a la propiedad `max_size` del nodo denominado `s` que debe ser un nodo Muestrear.
`:samplenode.max_size`	Hace referencia a la propiedad `max_size` del nodo Muestrear de la ruta actual (debe haber sólo un nodo Muestrear).
`s:sample.max_size`	Hace referencia a la propiedad `max_size` del nodo denominado `s` que debe ser un nodo Muestrear.
`t.direction.Age`	Hace referencia al rol del campo *Edad* del nodo Tipo `t`.
`:.max_size`	*** NO ES LEGAL *** Debe especificar el nombre o el tipo de nodo.

El ejemplo `s:sample.max_size` muestra que no es necesario deletrear los tipos de nodos al completo.

El ejemplo `t.direction.Age` muestra que algunos nombres de intervalo se pueden estructurar por sí mismos, en aquellos casos en que los atributos de un nodo sean más complejos que los intervalos individuales con valores individuales. Dichos intervalos se denominan **estructurados** o **complejos**.

Conceptos básicos de las propiedades de nodos

Cada tipo de nodo tiene su propio conjunto de propiedades legales y cada propiedad tiene un tipo. Este tipo puede ser un tipo general, número, marca o cadena, en cuyo caso, las configuraciones de la propiedad se forzarán en el tipo correcto. Surgirá un error en caso de que no se puedan forzar. También se puede dar el caso de que la referencia de la propiedad pueda especificar el rango de valores legales como `Discard`, `PairAndDiscard` e `IncludeAsText`, en cuyo caso se producirá un error si se utiliza otro valor. Las propiedades de marcas se deben leer o definir mediante los valores `True` y `False`. (Las variaciones que contengan `Off`, `OFF`, `off`, `No`, `NO`, `no`, `n`, `N`, `f`, `F`, `false`, `False`, `FALSE` o `0` también se reconocen al configurar los valores, pero pueden provocar errores al leer los valores de propiedad en algunos casos. El resto de valores se consideran verdaderos. El uso de `verdadero` y `falso` de forma consistente evitará confusiones). En las tablas de referencia de este manual, las propiedades estructuradas se indican como tales en la columna *Descripción de la propiedad* y se proporcionan los formatos de uso.

Propiedades de nodos comunes

Existen varias propiedades que son comunes a todos los nodos (incluidos los Supernodos) en IBM SPSS Modeler.

Tabla 38. Propiedades comunes de nodos.

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
`use_custom_name`	*flag*	

Tabla 38. Propiedades comunes de nodos (continuación).

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
`name`	*cadena*	Propiedad de sólo lectura que lee el nombre (automático o personalizado) para un nodo del lienzo.
`custom_name`	*cadena*	Especifica un nombre personalizado para el nodo.
`tooltip`	*cadena*	
`annotation`	*cadena*	
`keywords`	*cadena*	Intervalo estructurado que especifica una lista de palabras clave asociadas al objeto (por ejemplo, `["PalabraClave1" "PalabraClave2"]`).
`cache_enabled`	*flag*	

`tipo_nodo`	`source_supernode`
	`process_supernode`
	`terminal_supernode`
	todos los nombres de nodos tal como se especifican para script
		Propiedad de sólo lectura utilizada para hacer referencia a un nodo por tipo. Por ejemplo, en lugar de hacer referencia al nodo sólo por el nombre, como `ingresos_reales`, puede también especificar el tipo, como `userinputnode` o `filternode`.

Las propiedades específicas del Supernodo se tratan aparte como con los demás nodos. Consulte el tema Capítulo 19, “Propiedades de Supernodos”, en la página 305 para obtener más información.

Capítulo 8. Propiedades de ruta

Los scripts pueden controlar una serie de propiedades de la ruta. Para hacer referencia a propiedades de ruta, debe establecer el método de ejecución para que utilice scripts:

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

Ejemplo

La propiedad de nodo se utiliza para hacer referencia a los nodos en la ruta actual. El siguiente script de ruta muestra un ejemplo:

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

annotation = annotation + "\n\nEsta ruta se llama \"" + stream.getLabel() + "\" y
 contiene los nodos siguientes:\n"

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

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

El ejemplo anterior utiliza la propiedad node para crear una lista con todos los nodos de la ruta y escribir dicha lista en las anotaciones. La anotación generada tendrá el siguiente aspecto:

```
Esta ruta se llama "druglearn" y contiene los siguientes nodos:

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"
```

Las propiedades de la ruta se describen en la tabla siguiente.

Tabla 39. Propiedades de ruta.

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
execute_method	Normal Script	

Tabla 39. Propiedades de ruta (continuación).

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
date_format	"DDMMAA" "MMDDYY" "AAMMDD" "YYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-AAAA" "DD-MES-YY" "DD-MES-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.AAAA" "MM.DD.YYYY" "DD.MES.YY" "DD.MES.YYYY" "DD/MM/YY" "DD/MM/AAAA" "MM/DD/YY" "MM/DD/YYYY" "DD/MES/YY" "DD/MES/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	*tag*	
import_datetime_as_string	*tag*	
decimal_places	*number*	
decimal_symbol	Predeterminado Period Comma	
angles_in_radians	*tag*	
use_max_set_size	*tag*	
max_set_size	*number*	
ruleset_evaluation	Voting FirstHit	

Tabla 39. Propiedades de ruta (continuación).

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
refresh_source_nodes	*tag*	Se utiliza para actualizar los nodos de origen de forma automática al realizar la ejecución de la ruta.
script	*cadena*	
annotation	*cadena*	
name	*cadena*	**Nota:** Esta propiedad es de sólo lectura. Si desea cambiar el nombre de una ruta, debe guardarla con un nombre diferente.
parameters		Utilice esta propiedad para actualizar los parámetros de ruta desde dentor de un script autónomo.
nodos		Consulte la información detallada que se muestra a continuación.
codificación	SystemDefault "UTF-8"	
stream_rewriting	*booleano*	
stream_rewriting_maximise_sql	*booleano*	
stream_rewriting_optimise_clem_ejecución	*booleano*	
stream_rewriting_optimise_syntax_ejecución	*booleano*	
enable_parallelism	*booleano*	
sql_generation	*booleano*	
database_caching	*booleano*	
sql_logging	*booleano*	
sql_generation_logging	*booleano*	
sql_log_native	*booleano*	
sql_log_prettyprint	*booleano*	
record_count_suppress_input	*booleano*	
record_count_feedback_interval	*entero*	
use_stream_auto_create_node_valores	*booleano*	Si es true, se utilizan los valores específicos de la ruta, de lo contrario, se utilizan las preferencias de usuario.
create_model_applier_for_new_modelos	*booleano*	Si es true, cuando un constructor de modelos crea un modelo nuevo y no tiene enlaces de actualización activos, se añade un nuevo aplicador de modelos. **Nota:** Si utiliza IBM SPSS Modeler Batch versión 15, debe añadir de forma explícita el aplicador de modelos dentro de su script.
create_model_applier_update_links	createEnabled createDisabled doNotCreate	Define el tipo de enlace creado cuando se añade automáticamente un nodo aplicador de modelos.

Tabla 39. Propiedades de ruta (continuación).

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
`create_source_node_from_builders`	*booleano*	Si es true, cuando un constructor de modelos crea un resultado de origen nuevo y no tiene enlaces de actualización activos, se añade un nuevo nodo de origen.

`create_source_node_update_links`	`createEnabled`
	`createDisabled`
	`doNotCreate`
		Define el tipo de enlace creado cuando se añade automáticamente un nodo de origen.

`has_coordinate_system`	*booleano*	Si es verdadero, aplica un sistema de coordenadas a la ruta completa.
`coordinate_system`	*string*	El nombre del sistema de coordenadas proyectadas seleccionado.

Capítulo 9. Propiedades de nodos de origen

Propiedades comunes de nodos de origen

Las propiedades comunes a todos los nodos de origen se enumeran a continuación, con información sobre nodos específicos en los temas siguientes.

Ejemplo 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")
```

Ejemplo 2

Este script presupone que el archivo de datos especificado contiene un campo `Region` que representa una cadena de varias líneas.

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

# Cree un nodo Archivo de variables que lee el conjunto de datos que contiene
# el campo "Region"
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)

# Sustituya el tipo de almacenamiento para que sea una lista...
varfilenode.setKeyedPropertyValue("custom_storage_type", "Region", StorageType.LIST)
# ...y especifique el tipo de los valores de la lista y la profundidad de lista
varfilenode.setKeyedPropertyValue("custom_list_storage_type", "Region", StorageType.INTEGER)
varfilenode.setKeyedPropertyValue("custom_list_depth", "Region", 2)

# Cambie ahora la medición para identificar el campo como un valor geoespacial...
varfilenode.setKeyedPropertyValue("measure_type", "Region", MeasureType.GEOSPATIAL)
# ...y, finalmente, especifique la información necesaria sobre el tipo
# específico de objeto geoespacial
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")
```

Tabla 40. Propiedades comunes de nodos de origen.

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad

`dirección`	`Input`
	`Destino`
	`Both`
	`Ninguno`
	`Partition`
	`Split`
	`Frequency`
	`RecordID`
		Propiedad con clave para los roles de los campos.
		
		Formato de uso:
		`NODO.direction.NOMBRECAMPO`
		Nota: Los valores `In` y `Out` han quedado en desuso. Deberán de ser compatibles en versiones posteriores.

Tabla 40. Propiedades comunes de nodos de origen (continuación).

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
type	Range Flag Set Sin tipo Discrete Conjunto ordenado Predeterminado	Tipo de campo. Si se establece esta propiedad como *Default*, se borrará cualquier configuración del parámetro values y si value_mode tiene el valor *Specify*, se restablecerá a *Read*. Si value_mode se establece en *Pass* o *Read*, la configuración de type no le afectará. Formato de uso: NODO.type.NOMBRECAMPO
storage	Desconocido Cadena Entero Real Hora Fecha Marca de tiempo	Propiedad con clave de solamente lectura para el tipo de almacenamiento de campos. Formato de uso: NODO.storage.NOMBRECAMPO
check	Ninguno Nullify Coerce Descartar Warn Abort	Propiedad con clave para la comprobación del rango y el tipo de campo. Formato de uso: NODO.check.NOMBRECAMPO
values	[*value value*]	Para un campo continuo (rango), el primer valor es el mínimo y el último valor es el máximo. Para campos nominales (conjunto), especifique todos los valores. Para los campos marca, el primer valor representa *falso* y el último, *verdadero*. La configuración de esta propiedad establece de forma automática la propiedad value_mode en *Specify*. El almacenamiento se determina en función del primer valor de la lista, por ejemplo, si el primer valor es una *cadena*, el almacenamiento se establece en Cadena. Formato de uso: NODO.values.NOMBRECAMPO
value_mode	Leer Pasar Leer+ Actual Especifica	Determina la forma en que se han establecido los valores para un campo en la siguiente lectura de datos. Formato de uso: NODO.value_mode.NOMBRECAMPO Tenga en cuenta que no puede establecer esta propiedad directamente en *Specify*. Para utilizar valores específicos, establezca la propiedad values.
default_value_mode	Leer Pasar	Especifica el método predeterminado para configurar los valores de todos los campos. Formato de uso: NODO.default_value_mode Esta configuración puede anularse para determinados campos mediante la propiedad value_mode.

Tabla 40. Propiedades comunes de nodos de origen (continuación).

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
`extend_values`	*tag*	Se aplica cuando `value_mode` se establece en *Read*. Establézcala en *T* para añadir nuevos valores de lectura a los valores existentes del campo. Establézcala en *F* para descartar los valores existentes y favorecer a los nuevos valores de lectura. Formato de uso: `NODO.extend_values.NOMBRECAMPO`
`value_labels`	*cadena*	Se utiliza para especificar una etiqueta de valor. Tenga en cuenta que estos valores se deben especificar primero.
`enable_missing`	*tag*	Cuando está definida como *T*, activa el seguimiento de los valores perdidos para el campo. Formato de uso: `NODO.enable_missing.NOMBRECAMPO`
`missing_values`	[*value value ...*]	Especifica los valores de datos que denotan los datos perdidos. Formato de uso: `NODO.missing_values.NOMBRECAMPO`
`range_missing`	*tag*	Cuando esta propiedad se establece como *T*, especifica si se define un rango de valores perdidos (en blanco) para un campo. Formato de uso: `NODO.range_missing.NOMBRECAMPO`
`missing_lower`	*cadena*	Si `range_missing` es verdadero, especifica el límite inferior del rango de valores perdidos. Formato de uso: `NODO.missing_lower.NOMBRECAMPO`
`missing_upper`	*cadena*	Si `range_missing` es verdadero, especifica el límite superior del rango de valores perdidos. Formato de uso: `NODO.missing_upper.NOMBRECAMPO`
`null_missing`	*tag*	Cuando esta propiedad se establece en *T*, los valores nulos (valores no definidos que se muestran como `$null$` en el software) se consideran valores perdidos. Formato de uso: `NODO.null_missing.NOMBRECAMPO`
`whitespace_missing`	*tag*	Cuando esta propiedad está definida como *T*, los valores que solamente contienen un espacio en blanco (espacios, tabulaciones y líneas nuevas) se consideran valores perdidos. Formato de uso: `NODO.whitespace_missing.NOMBRECAMPO`
`description`	*cadena*	Se utiliza para especificar la descripción o etiqueta de un campo.

Tabla 40. Propiedades comunes de nodos de origen (continuación).

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
default_include	*tag*	Propiedad con clave para especificar si el comportamiento predeterminado es para pasar o filtrar los campos: NODO.default_include Ejemplo: set minodo:filternode.default_include = false
include	*tag*	Propiedad con clave que se utiliza para determinar si los campos individuales se han incluido o se han filtrado: NODO.include.NOMBRECAMPO.
new_name	*cadena*	
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	Esta propiedad con clave es similar a type en cuanto a que puede utilizarse para definir la medición asociada al campo. La diferencia es que, en los scripts Python, la función de establecimiento puede pasar también uno de los valores MeasureType, mientras que la función de obtención siempre devolverá los valores MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Para campos de recopilación (listas con profundidad 0), esta propiedad con clave define el tipo de medición asociado con los valores subyacentes.
geo_type	Point Multipunto Cadena lineal Cadena multilínea Polígono Multipolígono	En campos geoespaciales, esta propiedad con clave define el tipo del objeto geoespacial representado por este campo. Debería ser coherente con la profundidad de lista de los valores.
has_coordinate_system	*booleano*	En campos geoespaciales, esta propiedad define si este campo tiene un sistema de coordenadas
coordinate_system	*cadena*	En campos geoespaciales, esta propiedad con clave define el sistema de coordenadas para este campo.

Tabla 40. Propiedades comunes de nodos de origen (continuación).

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
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	Esta propiedad con clave es similar a custom_storage en cuanto a que puede utilizarse para definir el almacenamiento de alteración temporal para el campo. La diferencia es que, en los scripts Python, la función de establecimiento puede pasar también uno de los valores StorageType, mientras que la función de obtención siempre devolverá los valores StorageType.
custom_list_storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	Para campos de lista, esta propiedad con clave especifica el tipo de almacenamiento de los valores subyacentes.
custom_list_depth	*entero*	Para campos de lista, esta propiedad con clave especifica la profundidad del campo.

Propiedades de asimport

El origen de Analytic Server permite ejecutar una ruta en un sistema de archivos distribuido de Hadoop (HDFS en sus siglas inglesas).

Ejemplo

```
node = stream.create("asimport", "My node")
node.setPropertyValue("data_source", "Drug1n")
```

Tabla 41. Propiedades de asimport.

Propiedades de asimport	Tipo de datos	Descripción de la propiedad
data_source	*cadena*	Nombre del origen de datos.

Propiedades del nodo cognosimport

El nodo de origen de IBM Cognos BI importa datos desde las bases de datos de Cognos BI.

Ejemplo

```
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]"])
```

Tabla 42. Propiedades del nodo cognosimport.

Propiedades del nodo cognosimport	**Tipo de datos**	**Descripción de la propiedad**
mode	Datos Informe	Especifica si se importarán los datos de Cognos BI (valor predeterminado) o informes.
cognos_connection	*["string",flag,"string", "string" ,"string"]*	Una propiedad de la lista que contiene los detalles de conexión para el servidor de Cognos. El formato es: `["Cognos_server_URL", login_mode, "namespace", "username", "password"]` donde: `Cognos_server_URL` es la URL del servidor de Cognos que contiene el origen. `login_mode` indica si se utiliza el inicio de sesión anónimo, y es `true` o `false`; si se establece en `true`, los campos siguientes deben establecerse en `""`. `namespace` especifica el proveedor de autenticación de seguridad utilizado para registrarse en el servidor. `username` y `password` son los utilizados para registrarse en el servidor de Cognos. En lugar de `login_mode`, también hay disponibles las modalidades siguientes: • `anonymousMode`. Por ejemplo: `['Cognos_server_url', 'anonymousMode', "namespace", "username", "password"]` • `credentialMode`. Por ejemplo: `['Cognos_server_url', 'credentialMode', "namespace", "username", "password"]` • `storedCredentialMode`. Por ejemplo: `['Cognos_server_url', 'storedCredentialMode', "stored_credential_name"]` Donde `stored_credential_name` es el nombre de una credencial de Cognos del repositorio.
cognos_package_name	*cadena*	La ruta y el nombre del paquete de Cognos desde el que está importando objetos de datos, por ejemplo: `/Public Folders/GOSALES` **Nota:** solo es válida la barra diagonal.
cognos_items	*["campo","campo", ... ,"campo"]*	El nombre de uno o más objetos de datos que van a importarse. El formato de *campo* es [*espacio de nombre*].[*asunto de consulta*].[*elemento de consulta*]
cognos_filters	*campo*	El nombre de uno o más filtros que van a aplicarse antes de importar datos.

Tabla 42. Propiedades del nodo cognosimport (continuación).

Propiedades del nodo **cognosimport**	Tipo de datos	Descripción de la propiedad
cognos_data_parameters	*lista*	Valores de parámetros de solicitud de datos. Los pares nombre-valor van encerrados entre corchetes, los pares múltiples están separados por comas y toda la cadena está encerrada entre corchetes. Formato: [["*param1*", "*valor*"],...,["*paramN*", "*valor*"]]
cognos_report_directory	*campo*	La ruta de Cognos de una carpeta o paquete de la que importar informes, por ejemplo: `/Public Folders/GOSALES` **Nota:** Solo son válidas las barras diagonales.
cognos_report_name	*campo*	La ruta y nombre dentro de la ubicación del informe de un informe que se ha de importar.
cognos_report_parameters	*lista*	Valores de parámetros de informe. Los pares nombre-valor van encerrados entre corchetes, los pares múltiples están separados por comas y toda la cadena está encerrada entre corchetes. Formato: [["*param1*", "*valor*"],...,["*paramN*", "*valor*"]]

Propiedades de databasenode

El nodo Base de datos se puede utilizar para importar datos desde otros paquetes mediante ODBC (del inglés, Open Database Connectivity), incluidos Microsoft SQL Server, DB2, Oracle, etc.

Ejemplo

```
import modeler.api
ruta = 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")
```

Tabla 43. Propiedades de databasenode.

Propiedades de **databasenode**	Tipo de datos	Descripción de la propiedad
mode	`Tabla` `Query`	Especifique *Table* para conectarse a una tabla de base de datos mediante los controles del cuadro de diálogo, o *Query* para realizar una consulta a la base de datos seleccionada mediante SQL.
datasource	*cadena*	Nombre de la base de datos (consulte la siguiente nota).

Tabla 43. Propiedades de databasenode (continuación).

Propiedades de databasenode	Tipo de datos	Descripción de la propiedad
nombre de usuario	*cadena*	Detalles de conexión de la base de datos (consulte la siguiente nota).
contraseña	*cadena*	
credencial	*cadena*	Nombre de la credencial almacenada en IBM SPSS Collaboration and Deployment Services. Se puede utilizar en lugar de las propiedades username y password. El nombre de usuario y la contraseña de la credencial deben coincidir con el nombre de usuario y la contraseña necesarios para acceder a la base de datos
use_credential		Se establece en True o False.
epassword	*cadena*	Especifica una contraseña codificada como una alternativa a codificar una contraseña en un script. Consulte el tema "Generación de una contraseña codificada" en la página 51 para obtener más información. Esta propiedad es de sólo lectura durante la ejecución.
tablename	*cadena*	Nombre de la tabla a la que se desea tener acceso.
strip_spaces	Ninguno Left Right Both	Opciones para descartar los espacios iniciales y finales en las cadenas.
use_quotes	AsNeeded Always Nunca	Especifica si los nombres de columna y tabla aparecen entre comillas cuando las consultas se envían a la base de datos (en el caso, por ejemplo, de que contengan espacios o signos de puntuación).
query	*cadena*	Especifica el código SQL para la consulta que desea enviar.

Nota: Si el nombre de la base de datos (en la propiedad datasource) contiene uno o más espacios, puntos (llamados también "punto y aparte") o subrayados, puede utilizar el formato de "barra inclinada invertida y comillas dobles" para tratarlo como una cadena. Por ejemplo: "{\"db2v9.7.6_linux\"}" o: "{\"TDATA 131\"}". Además, encierre siempre los valores de cadena de datasource entre dobles comillas y llaves, como en el ejemplo siguiente: "{\"SQL Server\",spssuser,abcd1234,false}".

Nota: Si el nombre de la base de datos (en la propiedad datasource) contiene espacios, entonces en vez de las propiedades individuales para datasource, username y password, utilice un único origen de datos en el siguiente formato:

Tabla 44. Propiedades de databasenode - específicas de datasource.

Propiedades de databasenode	Tipo de datos	Descripción de la propiedad
datasource	*cadena*	Formato: [database_name,username,password[,true | false]] El último parámetro se usa con contraseñas cifradas. Si se define como true, la contraseña se cifrará antes de usarse.

Utilice este formato también si está cambiando el origen de datos, sin embargo, si tan sólo desea cambiar el nombre de usuario o contraseña, puede usar las propiedades username o password.

Propiedades de datacollectionimportnode

El nodo de importación de datos IBM SPSS Data Collection importa datos de encuesta basados en el modelo de datos de IBM SPSS Data Collection que utilizan los productos de investigación de mercados de IBM Corp.. Se debe instalar la biblioteca de datos de IBM SPSS Data Collection para utilizar este nodo.

Figura 7. Nodo de importación de datos Dimensiones

Ejemplo

```
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")
```

Tabla 45. Propiedades de datacollectionimportnode.

Propiedades de datacollectionimportnode	**Tipo de datos**	**Descripción de la propiedad**
metadata_name	*cadena*	El nombre del MDSC. El valor especial DimensionsMDD indica que se debería utilizar el documento de metadatos de IBM SPSS Data Collection estándar. Otro posibles valores podrían ser: mrADODsc mrI2dDsc mrLogDsc mrQdiDrsDsc mrQvDsc mrSampleReportingMDSC mrSavDsc mrSCDsc mrScriptMDSC El valor especial none indica que no existe ningún MDSC.
metadata_file	*cadena*	Nombre del archivo en el que se almacenan los metadatos.
casedata_name	*cadena*	El nombre del CDSC. Entre los posibles valores se encuentran: mrADODsc mrI2dDsc mrLogDsc mrPunchDSC mrQdiDrsDsc mrQvDsc mrRdbDsc2 mrSavDsc mrScDSC mrXmlDsc El valor especial none indica que no existe ningún CDSC.
casedata_source_type	Desconocido File Folder UDL DSN	Indica el tipo de origen del CDSC.
casedata_file	*cadena*	Cuando casedata_source_type es *File*, especifica el archivo que contiene los datos de casos.
casedata_folder	*cadena*	Cuando casedata_source_type es *Folder*, especifica la carpeta que contiene los datos de casos.
casedata_udl_string	*cadena*	Cuando casedata_source_type es *UDL*, especifica la cadena de conexión OLD-DB del origen de datos que contiene los datos de casos.
casedata_dsn_string	*cadena*	Cuando casedata_source_type es *DSN*, especifica la cadena de conexión ODBC del origen de datos.

Tabla 45. Propiedades de datacollectionimportnode (continuación).

Propiedades de datacollectionimportnode	**Tipo de datos**	**Descripción de la propiedad**
casedata_project	*cadena*	Al leer datos de casos de una base de datos de IBM SPSS Data Collection, puede escribir el nombre del proyecto. Para el resto de tipos de datos de casos, esta configuración se deberá dejar en blanco.
version_import_mode	Todos Latest Especifica	Define el modo en que deben tratarse las versiones.
specific_version	*cadena*	Cuando version_import_mode es *Specify*, define la versión de los datos de casos que se van a importar.
use_language	*cadena*	Determina si deben usarse las etiquetas de un idioma concreto.
language	*cadena*	Si use_language es verdadero, define el código de idioma que se va a usar en la importación. Este código de idioma debe incluirse entre aquellos disponibles en los datos de casos.
use_context	*cadena*	Determina si se debe importar un contexto específico. Los contextos se utilizan para modificar la descripción asociada con las respuestas.
context	*cadena*	Cuando use_context es verdadero, define el contexto de la importación. Este contexto debe encontrarse entre aquellos disponibles en los datos de casos.
use_label_type	*cadena*	Determina si se debe importar un tipo de etiqueta específico.
label_type	*cadena*	Cuando use_label_type es verdadero, define el tipo de etiqueta de la importación. Este tipo de etiqueta debe encontrarse entre aquellos disponibles en los datos de casos.
user_id	*cadena*	En el caso de las bases de datos que requieren un inicio de sesión explícito, puede proporcionar un ID de usuario y una contraseña para acceder al origen de datos.
contraseña	*cadena*	
import_system_variables	Común Ninguno Todos	Especifica las variables del sistema que se importan.
import_codes_variables	*tag*	
import_sourcefile_variables	*tag*	
import_multi_response	MultipleFlags Single	

Propiedades de excelimportnode

El nodo de importación Excel importa datos de Microsoft Excel en formato de archivo .xlsx. No es necesario un origen de datos ODBC.

Ejemplos

```
#Para usar un rango con nombre:
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)

#Para usar un rango explícito:
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")
```

Tabla 46. Propiedades de excelimportnode.

Propiedad de excelimportnode	Tipo de datos	Descripción de la propiedad
excel_file_type	Excel2007	
full_filename	*cadena*	El nombre completo del archivo, incluyendo la ruta.
use_named_range	*Booleana*	Si usar un rango con nombre. Si es verdadero, la propiedad named_range se utiliza para especificar el rango de lectura y se ignoran el resto de configuraciones de rango de datos y hojas de trabajo.
named_range	*cadena*	
worksheet_mode	Índice Name	Determina si la hoja de trabajo se define por el índice o por el nombre.
worksheet_index	*entero*	Índice de la hoja de trabajo que se va a leer, siendo 0 la primera hoja de trabajo, 1 la segunda, etc.
worksheet_name	*cadena*	Nombre de la hoja de trabajo que se va a leer.
data_range_mode	FirstNonBlank ExplicitRange	Especifica cómo debe establecerse el rango.
blank_rows	StopReading ReturnBlankRows	Cuando data_range_mode es *FirstNonBlank*, especifica cómo deben tratarse las filas en blanco.
explicit_range_start	*cadena*	Cuando data_range_mode es *ExplicitRange*, especifica el punto de partida del rango de lectura.
explicit_range_end	*cadena*	

Tabla 46. Propiedades de excelimportnode (continuación).

Propiedad de excelimportnode	Tipo de datos	Descripción de la propiedad
read_field_names	*Booleana*	Determina si la primera fila del rango concreto debería usarse como nombres de campo (columna).

Propiedades de evimportnode

El nodo Enterprise View crea una conexión a IBM SPSS Collaboration and Deployment Services Repository, lo que permite leer los datos de Enterprise View en una ruta y empaquetar un modelo en un escenario al que otros usuarios pueden acceder desde el repositorio.
Nota: El nodo Enterprise View se ha sustituido en SPSS Modeler 16.0 por el nodo Data View (Vista de datos). Todavía se admite el nodo Enterprise View para rutas guardadas en ediciones anteriores. No obstante, al actualizar o crear nuevas rutas se recomienda que utilice el nodo Data View.

Ejemplo

```
node = stream.create("evimport", "My node")
node.setPropertyValue("connection", ["Training data","/Application views/Marketing","LATEST",
"Analytic", "/Data Providers/Marketing"])
node.setPropertyValue("tablename", "cust1")
```

Tabla 47. Propiedades de evimportnode.

Propiedad de evimportnode	Tipo de datos	Descripción de la propiedad
connection	*list*	Propiedad estructura: lista de parámetros que componen una conexión de Enterprise View. Formato de uso: evimportnode.connection = [description,app_view_path, app_view_version_label, environment,DPD_path]
tablename	*cadena*	Nombre de una tabla de la vista de aplicación.

Propiedades de fixedfilenode

El nodo Archivo fijo importa datos desde archivos de texto de campo fijo; esto es, archivos cuyos campos no están delimitados pero empiezan en la misma posición y tienen una longitud fija. Los datos heredados o generados por la máquina se suelen almacenar en formato de campo fijo.

Ejemplo

```
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)
```

Tabla 48. Propiedades de fixedfilenode.

Propiedad de **fixedfilenode**	Tipo de datos	Descripción de la propiedad
record_len	*number*	Especifica el número de caracteres de cada registro.
line_oriented	*tag*	Omite el carácter de nueva línea al final de cada registro.
decimal_symbol	Predeterminado Comma Period	Tipo de separador decimal utilizado en el origen de datos.
skip_header	*number*	Especifica el número de líneas que se ignorarán al principio del primer registro. Esto resulta útil para ignorar las cabeceras de columna.
auto_recognize_datetime	*tag*	Especifica si las fechas o las horas se identifican automáticamente en los datos de origen.
lines_to_scan	*number*	
campos	*list*	Propiedad estructurada.
full_filename	*cadena*	Nombre completo del archivo que se va a leer, incluido el directorio.
strip_spaces	Ninguno Left Right Both	Descarta los espacios iniciales y finales en las cadenas de importación.
invalid_char_mode	Descartar Replace	Elimina los caracteres no válidos (nulo, 0 o cualquier carácter que no exista en la codificación actual) de la entrada de datos o sustituye los caracteres no válidos con el símbolo especificado de un carácter.
invalid_char_replacement	*cadena*	
use_custom_values	*tag*	
custom_storage	Desconocido Cadena Entero Real Hora Fecha Marca de tiempo	

Tabla 48. Propiedades de fixedfilenode (continuación).

Propiedad de **fixedfilenode**	Tipo de datos	Descripción de la propiedad
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MES-YY" "DD-MES-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MES.YY" "DD.MES.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MES/YY" "DD/MES/YYYY" MON YYYY q Q YYYY ww WK YYYY	Aplicable solamente si ha especificado un almacenamiento personalizado.
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"	Aplicable solamente si ha especificado un almacenamiento personalizado.
custom_decimal_symbol	*campo*	Aplicable solamente si ha especificado un almacenamiento personalizado.
codificación	StreamDefault SystemDefault "UTF-8"	Especifica el método de codificación de textos.

Propiedades del nodo gsdata_import

Utilice el nodo de origen Geospacial para llevar datos de mapa o espaciales en su sesión de minería de datos.

Tabla 49. Propiedades del nodo gsdata_import

Propiedades del nodo gsdata_import	Tipo de datos	Descripción de la propiedad
full_filename	*string*	Entre la vía de acceso del archivo .shp que desea cargar.
map_service_URL	*string*	Escriba el URL de servicio de mapas al que conectarse.
map_name	*string*	Sólo si se utiliza map_service_URL; esto contiene la estructura de carpeta de nivel superior del servicio de mapa.

Propiedades de sasimportnode

El nodo importar SAS importa datos SAS a IBM SPSS Modeler.

Ejemplo

```
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)
```

Tabla 50. Propiedades de sasimportnode.

Propiedad de sasimportnode	Tipo de datos	Descripción de la propiedad
format	Windows UNIX Transport SAS7 SAS8 SAS9	Formato del archivo que se va a importar.
full_filename	*cadena*	El nombre completo del archivo introducido, incluyendo su ruta.
member_name	*cadena*	Especifique el miembro para importar desde el archivo de transporte SAS especificado.
read_formats	*tag*	Lee formatos de datos (como etiquetas de variables) del archivo de formatos especificado.
full_format_filename	*cadena*	
import_names	NamesAndLabels LabelsasNames	Especifica el método para la correlación de nombres y etiquetas de variables en la importación.

Propiedades de simgennode

El nodo de generación de simulación proporciona una manera fácil de generar datos, ya sea desde cero utilizando las distribuciones o estadísticas especificada por el usuario o automáticamente utilizando las distribuciones obtenidas de la ejecución de un nodo de simulación de ajuste sobre datos históricos. Esto es útil cuando desea evaluar el resultado de un modelo predictivo en caso de dudas sobre las entradas del modelo.

Tabla 51. Propiedades de simgennode.

Propiedades de simgennode	Tipo de datos	Descripción de la propiedad
campos	Propiedad estructurada	Ver ejemplo
correlaciones	Propiedad estructurada	Ver ejemplo
keep_min_max_setting	*booleano*	
refit_correlations	*booleano*	
max_cases	*entero*	El valor mínimo es 1000, el valor máximo es 2,147,483,647.
create_iteration_field	*booleano*	
iteration_field_name	*cadena*	
replicate_results	*booleano*	
random_seed	*entero*	
parameter_xml	*cadena*	Devuelve el XML del parámetro como una cadena.

Ejemplo de fields

Este es un parámetro de ranura estructurado con la sintaxis siguiente:

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

distribution es una declaración de nombre de distribución seguido por una lista con parejas de nombres de atributo y valores. Cada distribución se define de la siguiente forma:

```
[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"]]], "", ""]])
```

Por ejemplo, para crear un nodo que genere un solo campo con una distribución binomial puede utilizar el siguiente script:

```
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]]], "", ""]])
```

La distribución binomial tiene 2 parámetros: n y prob. Puesto que binomial no admite los valores mínimo y máximo, éstos se suministran como una serie vacía.

Nota: No se puede establecer distribution directamente; utilícelo en combinación con la propiedad fields.

Los ejemplos siguientes muestran todos los tipos de distribución posibles. Tenga en cuenta que el umbral se especifica como thresh en NegativeBinomialFailures y en NegativeBinomialTrial.

```
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
])
```

Ejemplo de correlations

Este es un parámetro de ranura estructurado con la sintaxis siguiente:

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

La correlación puede ser cualquier número entre +1 y -1. Puede especificar tantas correlaciones como desee. Las correlaciones no especificadas se establecen en cero. Si alguno de los campos se desconocen, el valor de la correlación debe establecerse en la matriz de correlación (o tabla) y se muestra en texto rojo. Cuando hay campos desconocidos, no es posible ejecutar el nodo.

Propiedades de statisticsimportnode

El nodo IBM SPSS StatisticsArchivo lee los datos desde un formato de archivo *.sav* que utiliza IBM SPSS Statistics y archivos caché guardados en IBM SPSS Modeler, que también puede utilizar el mismo formato.

Las propiedades de este nodo están descritas en “Propiedades de statisticsimportnode” en la página 301.

Propiedades del nodo tm1import

El nodo de origen de IBM Cognos TM1 importa datos desde las bases de datos de Cognos TM1.

Tabla 52. Propiedades del nodo tm1import.

Propiedades del nodo tm1import	Tipo de datos	Descripción de la propiedad
pm_host	*string*	Nombre del host. Por ejemplo: `TM1_import.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')`
tm1_connection	*["campo","campo", ... ,"campo"]*	Una propiedad de la lista que contiene los detalles de conexión para el servidor de TM1. El formato es: `[ "TM1_Server_Name","tm1_ username","tm1_ password"]` Por ejemplo: `TM1_import.setPropertyValue("tm1_connection", ['Planning Sample', "admin", "apple"])`
selected_view	*["campo" "campo"]*	Una propiedad de la lista que contiene los detalles del cubo TM1 seleccionado y el nombre de la vista de cubo donde los datos se importarán en SPSS. Por ejemplo: `TM1_import.setPropertyValue("selected_view", ['plan_BudgetPlan', 'Goal Input'])`

Propiedades de userinputnode

El nodo Datos de usuario proporciona una manera fácil de crear datos sintéticos, ya sea partiendo de cero o modificando los datos existentes. Esto resulta útil, por ejemplo, cuando desee crear un conjunto de datos de comprobación para el modelado.

Ejemplo

```
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")
```

Tabla 53. Propiedades de userinputnode.

Propiedad de userinputnode	Tipo de datos	Descripción de la propiedad
data		
names		Intervalo estructurado que establece o devuelve una lista de nombres de campos generados por el nodo.

Tabla 53. Propiedades de userinputnode (continuación).

Propiedad de userinputnode	Tipo de datos	Descripción de la propiedad
custom_storage	Desconocido Cadena Entero Real Hora Fecha Marca de tiempo	Intervalo con clave que establece o devuelve el almacenamiento para un campo.
data_mode	Combined Ordered	Si se especifica Combined, los registros se generarán para cada combinación de valores del conjunto y valores mínimos y máximos. El número de registros generados será igual al producto del número de valores de cada campo. Si se especifica Ordered, se tomará un valor de cada columna para cada registro con el fin de generar una fila de datos. El número de registros generados será igual al número más grande de valores asociados a un campo. Los campos que tengan menos valores de datos se rellenarán con valores nulos.
values		**Nota:** Esta propiedad ya no se utiliza, y no debe usarse; en su lugar, se usa userinputnode.data.

Propiedades de variablefilenode

El nodo Archivo variable lee datos desde los archivos de texto de campo libre, esto es, campos cuyos registros contienen un número constante de campos pero un número variado de caracteres. Este nodo resulta también útil para los archivos con texto de cabecera de longitud fija y determinados tipos de anotaciones.

Ejemplo

```
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])
```

Tabla 54. Propiedades de variablefilenode.

Propiedad de variablefilenode	Tipo de datos	Descripción de la propiedad
skip_header	*número*	Especifica el número de caracteres que se ignorarán al principio del primer registro.

Tabla 54. Propiedades de variablefilenode (continuación).

Propiedad de variablefilenode	Tipo de datos	Descripción de la propiedad
num_fields_auto	*tag*	Determina el número de campos de cada registro de forma automática. Los registros deben terminar con un carácter de nueva línea.
num_fields	*número*	Especifica manualmente el número de campos de cada registro.
delimit_space	*tag*	Especifica el carácter utilizado para delimitar los límites de los campos del archivo.
delimit_tab	*tag*	
delimit_new_line	*tag*	
delimit_non_printing	*tag*	
delimit_comma	*tag*	En aquellos casos en los que la coma sea el delimitador del campo y el separador decimal para rutas, establezca delimit_other en *True* y especifique una coma como delimitador mediante la propiedad other.
delimit_other	*tag*	Permite especificar un delimitador personalizado mediante la propiedad other.
other	*cadena*	Especifica el delimitador utilizado cuando delimit_other es *True*.
decimal_symbol	Predeterminado Comma Period	Especifica el separador decimal utilizado en el origen de datos.
multi_blank	*tag*	Trata varios caracteres delimitadores vacíos adyacentes como un único delimitador.
read_field_names	*tag*	Trata la primera fila del archivo de datos como etiquetas para la columna.
strip_spaces	Ninguno Left Right Both	Descarta los espacios iniciales y finales en las cadenas de importación.
invalid_char_mode	Descartar Replace	Elimina los caracteres no válidos (nulo, 0 o cualquier carácter que no exista en la codificación actual) de la entrada de datos o sustituye los caracteres no válidos con el símbolo especificado de un carácter.
invalid_char_replacement	*cadena*	
break_case_by_newline	*flag*	Especifica que el delimitador de línea es el carácter de nueva línea.
lines_to_scan	*número*	Especifica cuántas líneas se van a explorar para los tipos de datos especificados.
auto_recognize_datetime	*tag*	Especifica si las fechas o las horas se identifican automáticamente en los datos de origen.
quotes_1	Descartar PairAndDiscard IncludeAsText	Especifica cómo se tratarán las comillas simples en la importación.

Tabla 54. Propiedades de variablefilenode (continuación).

Propiedad de variablefilenode	Tipo de datos	Descripción de la propiedad
1uotes_2	Descartar PairAndDiscard IncludeAsText	Especifica cómo se tratarán las comillas dobles en la importación.
full_filename	*cadena*	Nombre completo del archivo que se va a leer, incluido el directorio.
use_custom_values	*tag*	
custom_storage	Desconocido Cadena Entero Real Hora Fecha Marca de tiempo	
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MES-YY" "DD-MES-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MES.YY" "DD.MES.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MES/YY" "DD/MES/YYYY" MON YYYY q Q YYYY ww WK YYYY	Aplicable solamente si ha especificado un almacenamiento personalizado.

Tabla 54. Propiedades de variablefilenode (continuación).

Propiedad de variablefilenode	Tipo de datos	Descripción de la propiedad
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"	Aplicable solamente si ha especificado un almacenamiento personalizado.
custom_decimal_symbol	*campo*	Aplicable solamente si ha especificado un almacenamiento personalizado.
codificación	StreamDefault SystemDefault "UTF-8"	Especifica el método de codificación de textos.

Propiedades de xmlimportnode

El nodo de origen XML importa datos en formato XML en la ruta. Puede importar un único archivo o todos los archivos en un directorio. Puede especificar opcionalmente un archivo de esquema para leer la estructura XML.

Ejemplo

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

Tabla 55. Propiedades de xmlimportnode.

Propiedades de xmlimportnode	Tipo de datos	Descripción de la propiedad
read	single directory	Lee un único archivo de datos (valor predeterminado) o todos los archivos XML de un directorio.
recurse	*tag*	Especifica si leer además archivos XML de todos los subdirectorios del directorio especificado.
full_filename	*cadena*	(obligatorio) Ruta completa y nombre de archivo del archivo XML a importar (si read = single).
directory_name	*cadena*	(obligatorio) Ruta completa y nombre del directorio desde el que importar los archivos XML (si read = directory).

Tabla 55. Propiedades de xmlimportnode (continuación).

Propiedades de xmlimportnode	Tipo de datos	Descripción de la propiedad
full_schema_filename	*cadena*	Ruta completa y nombre de archivo del archivo XSD o DTD desde el que leer la estructura XML. Si omite este parámetro, se leerá la estructura desde el archivo de origen XML.
records	*cadena*	Expresión XPath (p.ej. /author/name) para definir el límite del registro. Cada vez que este elemento se encuentra en el archivo de origen se crea un nuevo registro.
mode	read specify	Lee todos los datos (valor predeterminado) o especifica qué elementos leer.
campos		Lista de elementos (elementos y atributos) para importar. Cada elemento de la lista es una expresión XPath.

Propiedades de dataviewimport

El nodo Vista de datos importa datos de Vista de datos en IBM SPSS Modeler.

Ejemplo

```
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")
```

Tabla 56. Propiedades de dataviewimport

Propiedades de dataviewimport	Tipo de datos	Descripción de la propiedad
analytic_data_source	*string*	La objeto Vista de datos analíticos almacenado en IBM SPSS Collaboration and Deployment Services. El nombre de vía de acceso y la etiqueta de la versión que se va a utilizar. ["Object ID","Full path", "Version"]

Tabla 56. Propiedades de dataviewimport (continuación)

Propiedades de dataviewimport	Tipo de datos	Descripción de la propiedad
table_name	*string*	La tabla de vista de datos utilizada en la Vista de datos analíticos. El nombre de la tabla debe estar cualificada para paquete. Puede obtener el paquete exportando la lista de materiales del cliente de gestor de despliegue de IBM SPSS Collaboration and Deployment Services y buscando en el archivo default.bom en el archivo ZIP exportado. El nombre del paquete siempre debe ser el mismo a menos que la lista de materiales se haya importado de IBM Operational Decision Management (iLOG). ["Object ID", "Name"]
data_access_plan	*string*	El plan de acceso de datos que se utiliza para proporcionar los datos para la Vista de datos analíticos. ["Object ID","Name"]
optional_attributes	*string*	Una lista de los atributos derivados a incluir. [["ID1","Name1"], ["ID2", "Name2"]]
include_xml	*booleano*	True si se va a incluir un campo con datos de instancia XOM. A menos que se utilicen nodos de IBM Analytical Decision Management iLOG, el valor recomendado es false. La activación de esta opción puede añadir una gran cantidad de proceso adicional.
include_xml_field	*string*	El nombre del campo a añadir cuando include_xml se establece en true.

Capítulo 10. Propiedades de nodos de operaciones con registros

Propiedades de appendnode

El nodo Añadir concatena conjuntos de registros. Es útil para combinar conjuntos de datos con estructuras parecidas, pero con datos diferentes.

Ejemplo

```
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")
```

Tabla 57. Propiedades de appendnode.

Propiedad de appendnode	Tipo de datos	Descripción de la propiedad
match_by	Position Name	Se pueden añadir conjuntos de datos basándose en la posición que tienen los campos en el origen de datos principal o el nombre de los campos en los conjuntos de datos de entrada.
match_case	*tag*	Activa la coincidencia de mayúsculas y minúsculas al hacer coincidir nombres de campos.
include_fields_from	Main All	
create_tag_field	*tag*	
tag_field_name	*cadena*	

Propiedades de aggregatenode

El nodo Agregar reemplaza una secuencia de registros de entrada con registros de salida agregados y resumidos.

Ejemplo

```
node = stream.create("aggregate", "My node")
# dbnode es un nodo de importación de base de datos configurado
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")
```

Tabla 58. Propiedades de aggregatenode.

Propiedad de aggregatenode	Tipo de datos	Descripción de la propiedad
keys	*lista*	Enumera los campos que se pueden usar como claves en la agregación. Por ejemplo, si Sexo y Región son los campos clave, cada combinación exclusiva de V y M con las regiones N y S (cuatro combinaciones exclusivas) tendrá un registro agregado.
contiguous	*tag*	Seleccione esta opción si sabe que todos los registros con los mismos valores clave se agrupan en la entrada (por ejemplo, si la entrada se clasifica en los campos clave). Con ello puede mejorar el rendimiento.
aggregates		Enumera los campos numéricos cuyos valores se añadirán, así como los modos de agregación elegidos.
aggregate_exprs		Propiedad con clave que aplica claves al nombre de campo derivado con la expresión agregada utilizada para calcularla. Por ejemplo: `aggregatenode.setKeyedPropertyValue ("aggregate_exprs", "Na_MAX", "MAX('Na')")`
extension	*cadena*	Especifica un prefijo o sufijo para campos agregados duplicados (consulte un ejemplo a continuación).
add_as	Suffix Prefix	
inc_record_count	*tag*	Crea un campo adicional que indica la cantidad de registros de entrada agregados para conformar cada registro agregado.
count_field	*cadena*	Especifica el nombre del campo de recuento de registros.
allow_approximation	*Boolean*	Permite la aproximación de estadísticas de ordenación cuando se realiza la agregación en Analytic Server
bin_count	*entero*	Especifica el número de intervalos a utilizar en la aproximación

Propiedades de balancenode

El nodo Equilibrar corrige los desequilibrios de un conjunto de datos para que cumpla una condición determinada. La directiva de equilibrado ajusta la proporción de registros si una condición es verdadera por el factor determinado.

Ejemplo

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

Tabla 59. Propiedades de balancenode.

Propiedad de balancenode	Tipo de datos	Descripción de la propiedad
directives		Propiedad estructurada para equilibrar la proporción de los valores de campos basados en un número determinado (consulte el ejemplo a continuación).
training_data_only	*tag*	Especifica que sólo se deben equilibrar los datos de entrenamiento. Si no se incluye ningún campo de partición en la ruta, se ignorará esta opción.

Esta propiedad de nodo utiliza el siguiente formato:

[[*número, cadena*] \ [*número, cadena*] \ ... [*número, cadena*]].

Nota: Si las cadenas (que utilizan comillas dobles) están incrustadas en la expresión, han de estar precedidas del carácter de escape " \ ". El carácter " \ " es también el carácter de continuación de línea, que puede utilizar para alinear los argumentos para mayor claridad.

Propiedades de derive_stbnode

El nodo Cajas-espacio-tiempo deriva Cajas-espacio-tiempo de los campos latitud, longitud e indicación de fecha y hora. Las Cajas-espacio-tiempo también pueden identificarse como lugares comunes.

Ejemplo

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

# Modalidad Individual Records (registros individuales)
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_")

# Modalidad Hangouts
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)
```

Tabla 60. Propiedades del nodo Cajas-Espacio-Tiempo

Propiedades de derive_stbnode	Tipo de datos	Descripción de la propiedad
mode	IndividualRecords Hangouts	
latitude_field	*campo*	
longitude_field	*campo*	
timestamp_field	*campo*	

Tabla 60. Propiedades del nodo Cajas-Espacio-Tiempo (continuación)

Propiedades de derive_stbnode	Tipo de datos	Descripción de la propiedad
hangout_density	*densidad*	Una sola densidad. Consulte en densities los valores de densidad válidos.
densities	[*densidad*,*densidad*,..., *densidad*]	Cada densidad es una cadena, por ejemplo, STB_GH8_1DAY. **Nota:** Existen límites para que las densidades sean válidas. En geohash, se pueden utilizar los valores de GH1 a GH15. Para la parte temporal, se pueden utilizar los valores siguientes: 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	*campo*	
qualifying_duration	1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2Hours 1HOUR 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	Debe ser una cadena.
min_events	*entero*	El valor de entero válido mínimo es de 2.
qualifying_pct	*entero*	Debe estar en el rango de 1 a 100.
add_extension_as	Prefix Suffix	
name_extension	*string*	

Propiedades de distinctnode

El nodo Distinguir se puede usar para eliminar registros duplicados pasando el primero de los registros distintos a la ruta de datos o descartando el primer registro y pasando cualquier duplicado a la ruta de datos en su lugar.

Ejemplo

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

Tabla 61. Propiedades de distinctnode.

Propiedad de distinctnode	Tipo de datos	Descripción de la propiedad
mode	Incluir Discard	Se puede incluir el primer nodo distinto en la ruta o descartar el primer nodo distinto y pasar en su lugar todos los registros duplicados a la ruta de datos.
grouping_fields	*lista*	Enumera los campos utilizados para determinar si los registros son idénticos. **Nota:** Esta propiedad está en desuso desde IBM SPSS Modeler 16 y posterior.
composite_value	Intervalo estructurado	Vea el ejemplo que se muestra a continuación.
composite_values	Intervalo estructurado	Vea el ejemplo que se muestra a continuación.
inc_record_count	*flag*	Crea un campo adicional que indica la cantidad de registros de entrada agregados para conformar cada registro agregado.
count_field	*cadena*	Especifica el nombre del campo de recuento de registros.
sort_keys	Intervalo estructurado	**Nota:** Esta propiedad está en desuso desde IBM SPSS Modeler 16 y posterior.
default_ascending	*flag*	
low_distinct_key_count	*tag*	Especifica que sólo tiene un pequeño número de registros y/o un pequeño número de valores exclusivos del campo(s) clave.
keys_pre_sorted	*tag*	Especifica que todos los registros con los mismos valores clave se agrupan en la entrada.
disable_sql_generation	*tag*	

Ejemplo para la propiedad composite_value

La propiedad composite_value tiene el formato general siguiente:

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

FILLOPTION tiene el formato [FillType, Option1, Option2, ...].

Ejemplos:

```
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"])
```

Las opciones personalizadas requieren más de un argumento, añadidos como una lista, por ejemplo:

```
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"])
```

Ejemplo para la propiedad `composite_values`

La propiedad `composite_values` tiene el formato general siguiente:

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

Ejemplo:

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

Propiedades de mergenode

El nodo Fundir toma varios registros de entrada y crea un registro de salida único que contiene todos o algunos de los campos de entrada. Es útil para fusionar datos desde diferentes orígenes, como datos de clientes internos y datos demográficos adquiridos.

Ejemplo

```
node = stream.create("merge", "My node")
# supongamos que customerdata y salesdata son nodos de importación de base de datos configurados
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"]])
```

Tabla 62. Propiedades de mergenode.

Propiedad de **mergenode**	Tipo de datos	Descripción de la propiedad
método	Order Claves Condition Rankedcondition	Especifique si los registros se fusionan en el orden en que aparecen en los archivos de datos, si se utilizarán uno o más campos clave para fusionar los registros que tengan el mismo valor en sus campos clave, si los registros se fusionarán si se satisface una condición especificada, o si se debe fusionar cada emparejamiento de fila de los conjuntos primario y todos los secundarios; utilizando la expresión de clasificación para ordenar coincidencias múltiples de menor a mayor.
condition	*cadena*	Si method se establece en Condition, especifica la condición para incluir o descartar registros.
key_fields	*lista*	
common_keys	*tag*	
join	Interior FullOuter PartialOuter Anti	
outer_join_tag.n	*tag*	En esta propiedad, *n* es el nombre de etiqueta tal y como recoge el cuadro de diálogo Seleccionar conjunto de datos. Tenga en cuenta que es posible que existan varios nombres de etiquetas especificados, ya que pueden ser varios los conjuntos de datos que aporten registros incompletos.
single_large_input	*tag*	Determina si se va a usar la optimización para tener una entrada relativamente grande en comparación con el resto de entradas.
single_large_input_tag	*cadena*	Especifica el nombre de etiqueta tal y como se muestra en el cuadro de diálogo Seleccionar conjunto de datos grande. Tenga en cuenta que el uso de esta propiedad es ligeramente distinto que el de la propiedad outer_join_tag (marca frente a cadena), ya que solamente se puede especificar un único conjunto de datos de entrada.
use_existing_sort_keys	*tag*	Determina si las entradas ya se han ordenado en función de uno o varios campos clave.
existing_sort_keys	[[*'string'*, 'Ascending'] \ [*'string''*, 'Descending']]	Especifica los campos que ya están ordenados y la dirección en que dicho orden se ha establecido.
primary_dataset	*cadena*	Si method es Rankedcondition, seleccione el conjunto de datos primario de la fusión. Esto se puede considerar como el lado izquierdo de una fusión de unión externa.
add_tag_duplicate	*Boolean*	Si method es Rankedcondition, y este valor está establecido en Y, si el conjunto de datos fusionado resultante contiene varios campos con el mismo nombre procedentes de orígenes de datos distintos, se añaden las etiquetas respectivas de los orígenes de datos al comienzo de las cabeceras de columna de campo.

Tabla 62. Propiedades de mergenode (continuación).

Propiedad de mergenode	Tipo de datos	Descripción de la propiedad
merge_condition	*cadena*	
ranking_expression	*cadena*	
Num_matches	*entero*	Número de coincidencias a devolver, en función de merge_condition y ranking_expression. Mínimo 1, máximo 100.

Propiedades rfmaggregatenode

El nodo Adición de RFM (actualidad, frecuencia, monetario) permite tomar datos de transacciones históricas de clientes, deshacerse de los datos no utilizados y combinar todos los datos de transacciones restantes en una única fila que indica cuándo hizo negociaciones con los clientes por última vez, cuántas transacciones hicieron y el valor monetario total de dichas transacciones.

Ejemplo

```
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")
```

Tabla 63. propiedades rfmaggregatenode.

propiedades rfmaggregatenode	Tipo de datos	Descripción de la propiedad
relative_to	Fixed Today	Especifica la fecha a partir de la que se calculará la actualidad de las transacciones.
reference_date	*date*	Sólo está disponible si se selecciona Fixed en relative_to.
contiguous	*tag*	Si los datos se han clasificado previamente de forma que todos los registros con el mismo ID aparecen en la ruta de datos, al seleccionar esta opción acelerará el procesamiento.
id_field	*campo*	Especifica el campo que desea utilizar para identificar el cliente y sus transacciones.
date_field	*campo*	Especifica el campo de fecha que se utilizará para calcular la actualidad.
value_field	*campo*	Especifica el campo que se utilizará para calcular el valor monetario.
extensión	*cadena*	Especifica un prefijo o sufijo para campos agregados duplicados.
add_as	Suffix Prefix	Especifica si la extensión se debe añadir como sufijo o prefijo.
discard_low_value_records	*tag*	Permite utilizar la configuración discard_records_below.

Tabla 63. propiedades rfmaggregatenode (continuación).

propiedades rfmaggregatenode	Tipo de datos	Descripción de la propiedad
discard_records_below	*número*	Especifique un valor mínimo por debajo del cual no se utilice ningún detalle de transacción al calcular los totales de RFM. Las unidades del valor se relacionan con el campo valor seleccionado.
only_recent_transactions	*tag*	Permite utilizar de la configuración specify_transaction_date o transaction_within_last.
specify_transaction_date	*tag*	
transaction_date_after	*date*	Sólo está disponible si selecciona specify_transaction_date. Especifica la fecha de transacción tras la que se incluirán los registros en su análisis.
transaction_within_last	*número*	Sólo está disponible si selecciona transaction_within_last. Especifica el número y tipo de períodos (días, semanas, meses o años) desde la fecha Calcular actualidad relativa a tras la cual se incluirán los registros en su análisis.
transaction_scale	Days Weeks Meses Años	Sólo está disponible si selecciona transaction_within_last. Especifica el número y tipo de períodos (días, semanas, meses o años) desde la fecha Calcular actualidad relativa a tras la cual se incluirán los registros en su análisis.
save_r2	*tag*	Muestra la fecha de la segunda transacción más reciente para cada cliente.
save_r3	*tag*	Sólo está disponible si selecciona save_r2. Muestra la fecha de la tercera transacción más reciente para cada cliente.

Propiedades de Rprocessnode

El nodo Rprocess le permite tomar los datos de una cadena de IBM(r) SPSS(r) Modeler y modificarlos utilizando su propio script R personalizado. Una vez modificados los datos, se devuelven a la cadena.

Ejemplo

```
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")
```

Tabla 64. Propiedades de Rprocessnode.

Propiedades de Rprocessnode	Tipo de datos	Descripción de la propiedad
syntax	*cadena*	

Tabla 64. Propiedades de Rprocessnode (continuación).

Propiedades de Rprocessnode	Tipo de datos	Descripción de la propiedad
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*flag*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*flag*	

Propiedades de samplenode

El nodo Muestrear selecciona un subconjunto de registros. Se admite una variedad de tipos de muestras, entre las que se incluyen las muestras estratificadas, agrupadas en clústeres y no aleatorias (estructuradas). El muestreo puede ser de gran utilidad para mejorar el rendimiento y para seleccionar grupos de registros o transacciones relacionadas para un análisis.

Ejemplo

```
/* 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]])
```

Tabla 65. Propiedades de samplenode.

Propiedad de samplenode	Tipo de datos	Descripción de la propiedad
método	Simple Complex	
mode	Incluir Descartar	Incluye o descarta los registros que reúnan la condición especificada.
sample_type	Primero OneInN RandomPct	Especifica el método de muestreo.
first_n	*entero*	Se incluirán o descartarán los registros hasta el punto de corte especificado.
one_in_n	*número*	Incluye o descarta cada *n* registros.
rand_pct	*número*	Especifica el porcentaje de registros que incluir o descartar.
use_max_size	*tag*	Activa el uso del parámetro maximum_size.

Tabla 65. Propiedades de samplenode (continuación).

Propiedad de **samplenode**	Tipo de datos	Descripción de la propiedad
maximum_size	*entero*	Especifica la muestra más grande que se va a incluir o descartar de la ruta de datos. Esta opción es redundante, por lo que se desactiva cuando se especifican las opciones First e Include.
set_random_seed	*tag*	Activa el uso del parámetro de semillas aleatorias.
random_seed	*entero*	Especifica el valor utilizado como semilla aleatoria.
complex_sample_type	Random Systematic	
sample_units	Proportions Counts	
sample_size_proportions	Fixed Custom Variable	
sample_size_counts	Fixed Custom Variable	
fixed_proportions	*número*	
fixed_counts	*entero*	
variable_proportions	*campo*	
variable_counts	*campo*	
use_min_stratum_size	*tag*	
minimum_stratum_size	*entero*	Esta opción sólo se aplica cuando se toma una muestra compleja con Sample units=Proportions.
use_max_stratum_size	*tag*	
maximum_stratum_size	*entero*	Esta opción sólo se aplica cuando se toma una muestra compleja con Sample units=Proportions.
clusters	*campo*	
stratify_by	*[campo1 ... campoN]*	
specify_input_weight	*tag*	
input_weight	*campo*	
new_output_weight	*cadena*	
sizes_proportions	[[string *valor cadena*][string *valor cadena*]...]	Si sample_units=proportions y sample_size_proportions=Custom, especifica un valor para cada combinación posible de valores de campos de especificación.
default_proportion	*número*	
sizes_counts	[[string *valor cadena*][string *valor cadena*]...]	Especifica un valor para cada combinación de valores posible de campos de estratificación. Se utiliza de forma similar a sizes_proportions pero especificando un entero en lugar de una proporción.
default_count	*número*	

Propiedades de selectnode

El nodo Seleccionar selecciona o descarta un subconjunto de registros de la ruta de datos en función de una condición específica. Por ejemplo, podría seleccionar los registros que pertenezcan a una región de ventas determinada.

Ejemplo

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

Tabla 66. Propiedades de selectnode.

Propiedad de selectnode	Tipo de datos	Descripción de la propiedad
mode	Incluir Descartar	Especifica si incluir o descartar los registros seleccionados.
condition	*cadena*	Condición para incluir o descartar registros.

Propiedades de sortnode

Los nodos Ordenar organizan registros en orden ascendente o descendente atendiendo a los valores de uno o varios campos.

Ejemplo

```
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"]])
```

Tabla 67. Propiedades de sortnode.

Propiedad de sortnode	Tipo de datos	Descripción de la propiedad
keys	*lista*	Especifica los campos que desea ordenar. Si no se especifica ninguna dirección, se utilizará la predeterminada.
default_ascending	*tag*	Especifica el orden de clasificación predeterminado.
use_existing_keys	*tag*	Determina si la clasificación se optimiza usando el orden de clasificación anterior de los campos que ya están ordenados.
existing_keys		Especifica los campos que ya están ordenados y la dirección en que dicho orden se ha establecido. Utiliza el mismo formato que las propiedades keys.

Propiedades streamingts

El nodo Generación de análisis TS construye y puntúa modelos de series temporales en un paso, sin la necesidad de un nodo Intervalos de tiempo.

Ejemplo

```
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")
```

Tabla 68. propiedades streamingts.

Propiedades de streamingts	Tipo de datos	Descripción de la propiedad
custom_fields	*flag*	Si custom_fields=false, se utilizarán los valores de un nodo Tipo situado en un punto anterior de la ruta. Si custom_fields=true, deberán especificarse targets e inputs.
targets	[*campo1...campoN*]	
inputs	[*campo1...campoN*]	
method	ExpertModeler Exsmooth Arima	
calculate_conf	*flag*	
conf_limit_pct	*real*	
use_time_intervals_node	*flag*	Si use_time_intervals_node=true, se utilizarán los valores de un nodo Intervalos de tiempo situado en un punto anterior de la ruta. Si use_time_intervals_node=false, deberán especificarse interval_offset_position, interval_offset e interval_type.
interval_offset_position	LastObservation LastRecord	LastObservation se refiere a la **Última observación válida**. LastRecord se refiere a la **Cuenta hacia atrás a partir del último registro**.
interval_offset	*número*	
interval_type	Períodos Años Trimestres Meses WeeksNonPeriodic DaysNonPeriodic HoursNonPeriodic MinutesNonPeriodic SecondsNonPeriodic	
eventos	*campos*	
expert_modeler_method	AllModels Exsmooth Arima	
consider_seasonal	*flag*	
detect_outliers	*flag*	

Tabla 68. propiedades streamingts (continuación).

Propiedades de streamingts	Tipo de datos	Descripción de la propiedad
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	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	Ninguno SquareRoot NaturalLog	
arima_p	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo
arima_d	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo
arima_q	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo
arima_sp	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo
arima_sd	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo
arima_sq	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo
arima_transformation_type	Ninguno SquareRoot NaturalLog	Misma propiedad que el nodo de modelado Serie de tiempo
arima_include_constant	*flag*	Misma propiedad que el nodo de modelado Serie de tiempo
tf_arima_p.*nombrecampo*	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_d.*nombrecampo*	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_q.*nombrecampo*	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_sp.*nombrecampo*	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_sd.*nombrecampo*	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_sq.*nombrecampo*	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_delay.*nombrecampo*	*entero*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.

Tabla 68. propiedades streamingts (continuación).

Propiedades de **streamingts**	Tipo de datos	Descripción de la propiedad
tf_arima_transformation_type.*nombredecampo*	Ninguno SquareRoot NaturalLog	
arima_detect_outlier_mode	Ninguno 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	Recuento Porcentaje	
deployment_rebuild_count	*número*	
deployment_rebuild_pct	*número*	
deployment_rebuild_field	*<campo>*	

Capítulo 11. Propiedades de nodos de operaciones con campos

Propiedades de anonymizenode

El nodo Anonimizar transforma la manera en que se representan los nombres y los valores de los campos a partir de ese punto de la ruta, lo que permite disfrazar los datos originales. Puede resultar útil si desea permitir que otros usuarios generen modelos utilizando datos confidenciales, como los nombres de los clientes u otros detalles.

Ejemplo

```
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)
# Anonimizar nodo requiere los campos de entrada al establecer los valores
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")
```

Tabla 69. Propiedades de anonymizenode

Propiedades de anonymizenode	Tipo de datos	Descripción de la propiedad
enable_anonymize	*flag*	Cuando se establece en True, activa la anonimización de los valores de los campos (equivale a seleccionar **Sí** para dicho campo en la columna Anonimizar valores).
use_prefix	*flag*	Cuando se establece en True, se utilizará un prefijo personalizado, si es que se ha establecido uno. Se aplica a los campos que se anonimizarán mediante el método Hash y es equivalente a elegir el botón de radio **Personalizado** en el cuadro de diálogo Reemplazar valores correspondiente a dicho campo.
prefix	*cadena*	Equivale a escribir un prefijo en el cuadro de texto del cuadro de diálogo Reemplazar valores. El prefijo predeterminado es el valor predeterminado, si no se ha especificado otra cosa.
transformation	Random Fixed	Determina si los parámetros de transformación que se aplican a un campo anonimizado mediante el método Transformar serán aleatorios o fijos.
set_random_seed	*flag*	Cuando se establece en True, se utilizará el valor inicial especificado (si transformation también se ha establecido en Random).
random_seed	*entero*	Cuando set_random_seed se establece en True, esta es la semilla para el número aleatorio.
scale	*número*	Cuando transformation se establece en Fixed, se utiliza este valor para "scale by" (escalar por). El valor máximo de la escala suele ser 10, pero puede reducirse para evitar desbordamientos.

Tabla 69. Propiedades de anonymizenode (continuación)

Propiedades de anonymizenode	Tipo de datos	Descripción de la propiedad
translate	*número*	Cuando transformation se establece en Fixed, se utiliza este valor para "translate" (convertir). El valor máximo de traslación suele ser 1000, pero puede reducirse para evitar desbordamientos.

Properties autodataprepnode

El nodo de preparación automática de datos (ADP) puede analizar sus datos e identificar los valores fijos, cribar los campos problemáticos o que no serán útiles y derivar nuevos atributos cuando sea necesario y mejorar el rendimiento mediante técnicas de cribado y muestreo inteligente. Puede utilizar el nodo de forma totalmente automática, permitiendo que el nodo seleccione y aplique valores fijos, o bien puede tener una vista previa de los cambios antes de que se apliquen y aceptarlos o rechazarlos.

Ejemplo

```
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")
```

Tabla 70. properties autodataprepnode

properties autodataprepnode	Tipo de datos	Descripción de la propiedad
objective	Balanced Speed Exactitud Personalizado	
custom_fields	*flag*	Si es verdadero, le permite especificar el objetivo, la entrada y otros campos del nodo actual. Si es falso, se utiliza la configuración actual de un nodo Tipo situado en un punto anterior de la ruta.
target	*campo*	Especifica un campo de objetivo único.
inputs	[*field1 ... fieldN*]	Campos de entrada o predictor utilizados por el modelo.
use_frequency	*flag*	
frequency_field	*campo*	
use_weight	*flag*	
weight_field	*campo*	
excluded_fields	Filter Ninguno	
if_fields_do_not_match	StopExecution ClearAnalysis	
prepare_dates_and_times	*flag*	Controla el acceso a todos los campos de fecha y hora
compute_time_until_date	*flag*	

Tabla 70. properties autodataprepnode (continuación)

properties autodataprepnode	Tipo de datos	Descripción de la propiedad
reference_date	Today Fixed	
fixed_date	*fecha*	
units_for_date_durations	Automatic Fixed	
fixed_date_units	Años Meses Days	
compute_time_until_time	*flag*	
reference_time	CurrentTime Fixed	
fixed_time	*hora*	
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	*número*	
exclude_too_many_categories	*flag*	
maximum_number_categories	*número*	
exclude_if_large_category	*flag*	
maximum_percentage_category	*número*	
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*	

Tabla 70. properties autodataprepnode (continuación)

properties autodataprepnode	Tipo de datos	Descripción de la propiedad
replace_missing_ordinal_target	*flag*	
maximum_values_for_ordinal	*número*	
minimum_values_for_continuous	*número*	
outlier_cutoff_value	*número*	
outlier_method	Replace Eliminar	
rescale_continuous_inputs	*flag*	
rescaling_method	MinMax ZScore	
min_max_minimum	*número*	
min_max_maximum	*número*	
z_score_final_mean	*número*	
z_score_final_sd	*número*	
rescale_continuous_target	*flag*	
target_final_mean	*número*	
target_final_sd	*número*	
transform_select_input_fields	*flag*	
maximize_association_with_target	*flag*	
p_value_for_merging	*número*	
merge_ordinal_features	*flag*	
merge_nominal_features	*flag*	
minimum_cases_in_category	*número*	
bin_continuous_fields	*flag*	
p_value_for_binning	*número*	
perform_feature_selection	*flag*	
p_value_for_selection	*número*	
perform_feature_construction	*flag*	
transformed_target_name_extension	*cadena*	
transformed_inputs_name_extension	*cadena*	
constructed_features_root_name	*cadena*	
years_duration_ name_extension	*cadena*	
months_duration_ name_extension	*cadena*	
days_duration_ name_extension	*cadena*	
hours_duration_ name_extension	*cadena*	
minutes_duration_ name_extension	*cadena*	
seconds_duration_name_extension	*cadena*	
year_cyclical_name_extension	*cadena*	
month_cyclical_name_extension	*cadena*	
day_cyclical_name_extension	*cadena*	
hour_cyclical_name_extension	*cadena*	

Tabla 70. properties autodataprepnode (continuación)

properties autodataprepnode	Tipo de datos	Descripción de la propiedad
minute_cyclical_name_extension	*cadena*	
second_cyclical_name_extension	*cadena*	

Propiedades de astimeintervalsnode

El nodo Intervalos de tiempo original no es compatible con el Servidor de análisis (AS). El nodo Intervalos de tiempo AS (nuevo en SPSS Modeler versión 17.0) contiene un subconjunto de las funciones del nodo Intervalos de tiempo existente que se puede utilizar con Servidor de análisis.

Utilice el nodo Intervalos de tiempo AS para especificar intervalos y derivar un campo de tiempo nuevo para la estimación o previsión. Se admite una gama completa de intervalos de tiempo que abarca desde segundos a años.

Tabla 71. Propiedades de astimeintervalsnode

Propiedades de astimeintervalsnode	Tipo de datos	Descripción de la propiedad
time_field	*campo*	Solo puede aceptar un único campo continuo. Ese campo lo utiliza el nodo como la clave de agregación para convertir el intervalo. Si se utiliza aquí un campo de entero, se considera como un índice de tiempo.
dimensiones	*[campo1 campo2 ... campon]*	Estos campos se utilizan para crear series temporales individuales basándose en los valores de campo.
fields_to_aggregate	*[campo1 campo2 ... campon]*	Estos campos se agregan como parte del cambio del período del campo de tiempo. Los campos no incluidos en este selector se filtran y excluyen de los datos que salen del nodo.

Propiedades de binningnode

El nodo Intervalos crea automáticamente nuevos campos nominales (conjunto) en función de los valores de uno o más campos continuos (rango numérico) existentes. Por ejemplo, puede transformar un campo de ingresos continuo en un campo categórico nuevo que contenga grupos de ingresos como desviaciones desde la media. Una vez creados los intervalos para el campo nuevo, puede generar un nodo Derivar en función de los puntos de corte.

Ejemplo

```
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)
```

Tabla 72. Propiedades de binningnode

Propiedad de binningnode	Tipo de datos	Descripción de la propiedad
campos	*[campo1 campo2 ... campon]*	Los campos continuos (rango numérico) pendientes de transformación. Se pueden crear intervalos de varios campos de forma simultánea.
method	FixedWidth EqualCount Rank SDev Optimal	Método utilizado para determinar los puntos de corte de los intervalos de campo nuevos (categorías).
rcalculate_bins	Always IfNecessary	Especifica si se vuelven a calcular los intervalos y los datos se colocan en el intervalo adecuado cada vez que se ejecuta el nodo o si los datos sólo se añaden a los intervalos existentes y cualquier nuevo intervalo que se haya añadido.
fixed_width_name_extension	*cadena*	La extensión predeterminada es *_BIN*.
fixed_width_add_as	Suffix Prefix	Determina si la extensión se debe añadir al principio (prefijo) o al final (sufijo) del nombre de campo. La extensión predeterminada es *income_BIN*.
fixed_bin_method	Width Count	
fixed_bin_count	*entero*	Especifica un número entero para determinar el número de intervalos de anchura fija (categorías) para los nuevos campos.
fixed_bin_width	*real*	Valor (entero o real) para calcular el ancho del intervalo.
equal_count_name_ extensión	*cadena*	La extensión predeterminada es *_TILE*.
equal_count_add_as	Suffix Prefix	Especifica una extensión, sufijo o prefijo, utilizada para el nombre de los campos generados con p-tiles estándar. La extensión predeterminada es *_TILE* más *N*, donde *N* es el número de cuantil.
tile4	*flag*	Genera cuatro intervalos de cuantiles, cada uno con el 25% de los casos.
tile5	*flag*	Genera cinco intervalos de quintiles.
tile10	*flag*	Genera 10 intervalos de deciles.
tile20	*flag*	Genera 20 intervalos de veintiles.
tile100	*flag*	Genera 100 intervalos de percentiles.
use_custom_tile	*flag*	
custom_tile_name_extension	*cadena*	La extensión predeterminada es *_TILEN*.
custom_tile_add_as	Suffix Prefix	
custom_tile	*entero*	

Tabla 72. Propiedades de binningnode (continuación)

Propiedad de binningnode	Tipo de datos	Descripción de la propiedad
equal_count_method	RecordCount ValueSum	El método RecordCount trata de asignar el mismo número de registros a cada intervalo, mientras que ValueSum asigna registros de manera que la suma de los valores de cada intervalo sea la misma.
tied_values_method	Next Actual Random	Especifica en qué intervalo se van a insertar los datos de valor empatado.
rank_order	Ascending Descending	Esta propiedad incluye Ascending (el valor más bajo se marca con 1) o Descending (el valor más alto se marca con 1).
rank_add_as	Suffix Prefix	Esta opción se aplica al rango, rango fraccional y rango como porcentaje.
rango	*flag*	
rank_name_extension	*cadena*	La extensión predeterminada es *_RANK*.
rank_fractional	*flag*	Establece rangos de casos en los que el valor del campo nuevo es igual al rango dividido por la suma de las ponderaciones de los casos que no están perdidos. Los rangos fraccionales están dentro del rango de 0–1.
rank_fractional_name_ extensión	*cadena*	La extensión predeterminada es *_F_RANK*.
rank_pct	*flag*	Cada rango se divide por el número de registros con valores válidos y se multiplica por 100. Los rangos fraccionales de porcentaje están dentro del rango de 1–100.
rank_pct_name_extension	*cadena*	La extensión predeterminada es *_P_RANK*.
sdev_name_extension	*cadena*	
sdev_add_as	Suffix Prefix	
sdev_count	One Two Three	
optimal_name_extension	*cadena*	La extensión predeterminada es *_OPTIMAL*.
optimal_add_as	Suffix Prefix	
optimal_supervisor_field	*campo*	Campo elegido como campo supervisor, con el que se relacionan los campos seleccionados para los intervalos.
optimal_merge_bins	*flag*	Especifica que todos los intervalos con un número pequeño de casos se añadirán a un intervalo vecino de mayor tamaño.
optimal_small_bin_threshold	*entero*	
optimal_pre_bin	*flag*	Indica si debe agruparse previamente en intervalos el conjunto de datos.
optimal_max_bins	*entero*	Especifica un límite superior con el fin de evitar que se genere un número desmesurado de intervalos.

Tabla 72. Propiedades de binningnode (continuación)

Propiedad de binningnode	Tipo de datos	Descripción de la propiedad
optimal_lower_end_point	Inclusive Exclusive	
optimal_first_bin	Unbounded Bounded	
optimal_last_bin	Unbounded Bounded	

Propiedades de derivenode

El nodo Derivar modifica los valores de datos o crea campos nuevos desde uno o más campos existentes. Crea campos del tipo fórmula, marca, nominal, estado, recuento y condicional.

Ejemplo 1

```
# Crear y configurar un nodo de campo Derivar marca
node = 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\"")

# Crear y configurar un nodo de campo Derivar condicional
node = 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\"")
```

Ejemplo 2

Este script presupone que existen dos columnas numéricas denominadas `XPos` e `YPos` que representan las coordenadas X e Y de un punto (por ejemplo, donde se ha producido un evento). El script crea un nodo Derivar que calcula una columna geoespacial de coordenadas X e Y que representan dicho punto en un sistema de coordenadas específico:

```
stream = modeler.script.stream()
# Otro código de configuración de ruta
node = stream.createAt("derive", "Location", 192, 96)
node.setPropertyValue("new_name", "Location")
node.setPropertyValue("formula_expr", "['XPos', 'YPos']")
node.setPropertyValue("formula_type", "Geospatial")
# Ahora que hemos definido el tipo de medición general, defina los
# detalles del objeto geoespacial
node.setPropertyValue("geo_type", "Point")
node.setPropertyValue("has_coordinate_system", True)
node.setPropertyValue("coordinate_system", "ETRS_1989_EPSG_Arctic_zone_5-47")
```

Tabla 73. Propiedades de derivenode

Propiedad de derivenode	Tipo de datos	Descripción de la propiedad
new_name	*cadena*	Nombre del campo nuevo.

Tabla 73. Propiedades de derivenode (continuación)

Propiedad de derivenode	Tipo de datos	Descripción de la propiedad
mode	Single Múltiples	Especifica si los campos son únicos o múltiples.
campos	*lista*	Se utiliza en modo Múltiple solamente para seleccionar varios campos.
name_extension	*cadena*	Especifica la extensión de los nombres de los nuevos campos.
add_as	Suffix Prefix	Añade la extensión como un prefijo (al principio) o como un sufijo (al final) del nombre de los campos.
result_type	Formula Flag Set State Count Conditional	Los seis tipos de campos nuevos que se pueden crear.
formula_expr	*cadena*	Expresión para calcular un nuevo valor de campo en el nodo Derivar.
flag_expr	*cadena*	
flag_true	*cadena*	
flag_false	*cadena*	
set_default	*cadena*	
set_value_cond	*cadena*	Estructurada para proporcionar la condición asociada a un valor dado.
state_on_val	*cadena*	Especifica el valor del campo nuevo cuando se cumple la condición Activado.
state_off_val	*cadena*	Especifica el valor del campo nuevo cuando se cumple la condición Desactivado.
state_on_expression	*cadena*	
state_off_expression	*cadena*	
state_initial	On Off	Asigna a cada registro del nuevo campo un valor inicial activado (On) o desactivado (Off). Este valor puede cambiar a medida que se cumplan las condiciones.
count_initial_val	*cadena*	
count_inc_condition	*cadena*	
count_inc_expression	*cadena*	
count_reset_condition	*cadena*	
cond_if_cond	*cadena*	
cond_then_expr	*cadena*	
cond_else_expr	*cadena*	

Tabla 73. Propiedades de derivenode (continuación)

Propiedad de derivenode	Tipo de datos	Descripción de la propiedad
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	Esta propiedad se puede utilizar para definir la medición asociada con el campo derivado. La función de establecimiento se puede pasar como una cadena o uno de los valores MeasureType. El método de obtención siempre devolverá los valores MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Para campos de recopilación (listas con profundidad 0), esta propiedad define el tipo de medición asociado con los valores subyacentes.
geo_type	Point Multipunto Cadena lineal Cadena multilínea Polígono Multipolígono	En campos geoespaciales, esta propiedad define el tipo del objeto geoespacial representado por este campo. Debería ser coherente con la profundidad de lista de los valores
has_coordinate_system	*booleano*	En campos geoespaciales, esta propiedad define si este campo tiene un sistema de coordenadas
coordinate_system	*cadena*	En campos geoespaciales, esta propiedad define el sistema de coordenadas para este campo

Propiedades de ensemblenode

El nodo Conjunto combina dos o más nugget de modelo para obtener predicciones más precisas que pueden conseguirse de cualquier modelo.

Ejemplo

```
# Crear y configurar un nodo Conjunto
# Utilizar este nodo con los modelos en demos\streams\pm_module\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")
```

Tabla 74. Propiedades de ensemblenode.

Propiedades de ensemblenode	Tipo de datos	Descripción de la propiedad
ensemble_target_field	*campo*	Especifica el campo objetivo de todos los modelos utilizados en el conjunto.
filter_individual_model_output	*tag*	Especifica si los resultados de puntuación de los modelos individuales se deben eliminar.

Tabla 74. Propiedades de ensemblenode (continuación).

Propiedades de ensemblenode	Tipo de datos	Descripción de la propiedad
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting AdjustedPropensityWeightedVoting HighestConfidence AverageRawPropensity AverageAdjustedPropensity	Especifica el método utilizado para determinar la puntuación del conjunto. Este conjunto sólo se aplica si el objetivo seleccionado es un campo de marca.
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	Especifica el método utilizado para determinar la puntuación del conjunto. Este conjunto sólo se aplica si el objetivo seleccionado es un campo nominal.
flag_voting_tie_selection	Random HighestConfidence RawPropensity AdjustedPropensity	Si se selecciona un método de votación, especifica cómo se resolverán los empates. Este conjunto sólo se aplica si el objetivo seleccionado es un campo de marca.
set_voting_tie_selection	Random HighestConfidence	Si se selecciona un método de votación, especifica cómo se resolverán los empates. Este conjunto sólo se aplica si el objetivo seleccionado es un campo nominal.
calculate_standard_error	*tag*	Si el campo objetivo es continuo, se ejecuta un error estándar de forma predeterminada para calcular la diferencia entre los valores medidos o estimados y los valores true; y para mostrar si las estimaciones coinciden.

Propiedades de fillernode

El nodo Rellenar sustituye valores de campos y cambia el almacenamiento. Puede sustituir los valores en función de una condición CLEM, como `@BLANK(@FIELD)`. También puede sustituir todos los espacios vacíos o valores nulos por un valor específico. Un nodo Rellenar suelen utilizarse junto con un nodo Tipo para sustituir valores perdidos.

Ejemplo

```
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\"")
```

Tabla 75. Propiedades de fillernode

Propiedad de fillernode	Tipo de datos	Descripción de la propiedad
campos	*lista*	Campos del conjunto de datos cuyos valores se van a examinar y sustituir.

Tabla 75. Propiedades de fillernode (continuación)

Propiedad de fillernode	Tipo de datos	Descripción de la propiedad
replace_mode	Always Conditional Vacío Null BlankAndNull	Se pueden sustituir todos los valores, valores vacíos o valores nulos, o bien reemplazar aquellos basados en una condición específica.
condition	*cadena*	
replace_with	*cadena*	

Propiedades de filternode

El nodo Filtrar filtra (descarta) campos, vuelve a nombrar campos y correlaciona campos de nodo de origen a otro.

Ejemplo

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

Usando la propiedad default_include. Tenga en cuenta que, si establece el valor de la propiedad default_include, no se incluirán o excluirán automáticamente todos los campos, sino que simplemente se determinará el valor predeterminado de los seleccionados actualmente. Esto equivale funcionalmente a pulsar en el botón **Incluir campos de forma predeterminada** del cuadro de diálogo del nodo Filtrar. Por ejemplo, imagine que ejecuta el siguiente script:

```
node = modeler.script.stream().create("filter", "Filter")
node.setPropertyValue("default_include", False)
# Incluir estos dos campos en la lista
for f in ["Age", "Sex"]:
    node.setKeyedPropertyValue("include", f, True)
```

Esto hará que el nodo pase los campos *Edad* y *Sexo* y descarte el resto. Ahora, imagine que ejecuta de nuevo el mismo script, pero designa dos campos diferentes:

```
node = modeler.script.stream().create("filter", "Filter")
node.setPropertyValue("default_include", False)
# Incluir estos dos campos en la lista
for f in ["BP", "Na"]:
    node.setKeyedPropertyValue("include", f, True)
```

De esta forma, se añadirán dos campos más al filtro, de manera que pasan un total de cuatro campos (*Edad*, *Sexo*, *PS* y *Na*). En otras palabras, al volver a establecer el valor de default_include en False no se restablecen automáticamente todos los campos.

Si lo desea, si ahora cambia default_include a True (ya sea usando un script o en el cuadro de diálogo del nodo Filtrar), cambiará el comportamiento de forma que los cuatro campos enumerados anteriormente no se incluirían, sino que quedarían descartados. Si no está seguro, se recomienda experimentar con los controles del cuadro de diálogo del nodo Filtrar para entender esta interacción.

Tabla 76. Propiedades de filternode

Propiedad de **filternode**	Tipo de datos	Descripción de la propiedad
default_include	*flag*	Propiedad con clave para especificar si el comportamiento predeterminado es para pasar o filtrar los campos: Tenga en cuenta que, si se establece esta propiedad, no se incluirán o excluirán automáticamente todos los campos, sino que simplemente se determinará si los campos seleccionados se incluyen o excluyen de forma predeterminada. Vea el ejemplo que se muestra a continuación para ver más comentarios.
include	*flag*	Propiedad con clave para incluir y eliminar el campo.
new_name	*cadena*	

Propiedades de historynode

El nodo Historial se utiliza para crear campos nuevos que contienen datos de los campos de registros anteriores. Los nodos Historial se suelen utilizar para los datos secuenciales, como los datos de series temporales. Antes de utilizar un nodo Historial, puede desear ordenar los datos utilizando un nodo Ordenar.

Ejemplo

```
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")
```

Tabla 77. Propiedades de historynode

Propiedad de **historynode**	Tipo de datos	Descripción de la propiedad
campos	*lista*	Campos para los que desea un historial.
offset	*número*	Especifica el último registro anterior al registro actual desde el que desea extraer valores del campo histórico.
span	*número*	Especifica el número de registros anteriores de los que desea extraer valores.
unavailable	Descartar Leave Fill	Para tratar registros que no tienen valores de historial, suele hacer referencia a los primeros registros (en la parte superior del conjunto de datos), de los que no hay registros previos que utilizar como historial.
fill_with	Cadena Number	Especifica un valor o cadena que utilizar en el caso de los registros en los que no existen valores de historial disponibles.

Propiedades de partitionnode

El nodo Partición genera un campo de partición, que divide los datos en subconjuntos diferentes para las fases de entrenamiento, comprobación y validación en la generación del modelo.

Ejemplo

```
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")
```

Tabla 78. Propiedades de partitionnode

Propiedad de partitionnode	Tipo de datos	Descripción de la propiedad
new_name	*cadena*	Nombre del campo de partición generado por el nodo.
create_validation	*flag*	Especifica si se debe crear una partición de validación.
training_size	*entero*	Porcentaje de registros (0-100) que se van a asignar a la partición de entrenamiento.
testing_size	*entero*	Porcentaje de registros (0-100) que se van a asignar a la partición de comprobación.
validation_size	*entero*	Porcentaje de registros (0-100) que se van a asignar a la partición de entrenamiento. Se ignora si no se crea una partición de validación.
training_label	*cadena*	Etiqueta para la partición de entrenamiento.
testing_label	*cadena*	Etiqueta para la partición de comprobación.
validation_label	*cadena*	Etiqueta para la partición de validación. Se ignora si no se crea una partición de validación.
value_mode	Sistema SystemAndLabel Label	Especifica los valores utilizados para representar cada partición en los datos. Por ejemplo, el entero del sistema 1, la etiqueta Entrenamiento o una combinación de los dos, 1_Entrenamiento, pueden representar la muestra de entrenamiento.
set_random_seed	*Booleana*	Especifica si se debe utilizar una semilla aleatoria especificada por el usuario.
random_seed	*entero*	Valor de semilla aleatoria especificada por el usuario. Para que se pueda utilizar este valor, set_random_seed se debe establecer en True.
enable_sql_generation	*Booleana*	Especifica si se utiliza la retrotracción SQL para asignar registros a particiones.
unique_field		Especifica el campo de entrada que se utiliza para garantizar que los registros se asignan a particiones de una forma aleatoria pero reproducible. Para que se pueda utilizar este valor, enable_sql_generation se debe establecer en True.

Propiedades de reclassifynode

El nodo Reclasificar transforma un conjunto de valores categóricos en otro. La reclasificación es útil para contraer categorías o reagrupar datos para su análisis.

Ejemplo

```
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"])
```

Tabla 79. Propiedades de reclassifynode

Propiedad de reclassifynode	Tipo de datos	Descripción de la propiedad
mode	Single Múltiples	Single reclasifica las categorías de un campo. Multiple activa las opciones que permiten la transformación de varios campos al mismo tiempo.
replace_field	*flag*	
field	*cadena*	Sólo se utiliza en modo Single.
new_name	*cadena*	Sólo se utiliza en modo Single.
campos	*[field1 field2 ... fieldn]*	Sólo se utiliza en modo Multiple.
name_extension	*cadena*	Sólo se utiliza en modo Multiple.
add_as	Suffix Prefix	Sólo se utiliza en modo Multiple.
reclasificar	*cadena*	Propiedad estructurada para valores de campos.
use_default	*flag*	Utiliza el valor predeterminado.
valor predeterminado	*cadena*	Especifica un valor predeterminado.
pick_list	*[string string ... string]*	Permite al usuario importar una lista de valores nuevos conocidos para rellenar la lista desplegable de la tabla.

Propiedades de reordernode

El nodo Reorg. campos define el orden natural utilizado para mostrar los campos en la parte posterior de la ruta. Este orden afecta a la visualización de los campos en diversas ubicaciones, como las tablas, las listas y el selector de campos. Esta operación resulta útil al trabajar con conjuntos de datos amplios que hacen más visibles los campos de interés.

Ejemplo

```
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"])
```

Tabla 80. Propiedades de reordernode

Propiedad de reordernode	Tipo de datos	Descripción de la propiedad
mode	Custom Automático	Se pueden ordenar los valores de forma automática o especificar un orden personalizado.
sort_by	Name Tipo Storage	
ascending	*flag*	
start_fields	*[field1 field2 ... fieldn]*	Los campos nuevos se han insertado después de estos campos.
end_fields	*[field1 field2 ... fieldn]*	Los campos nuevos se han insertado antes de estos campos.

Propiedades de reprojectnode

En SPSS Modeler, elementos como las funciones espaciales Creador de expresiones, el nodo de Predicción espacio-temporal (STP) y el nodo Visualización de mapas utilizan el sistema de coordenadas proyectado. Utilice el nodo Reproyectar para cambiar el sistema de coordenadas de los datos que importa y que utilizan un sistema de coordenadas geográficas.

Tabla 81. Propiedades de reprojectnode

Propiedades de reprojectnode	Tipo de datos	Descripción de la propiedad
reproject_fields	*[campo1 campo2 ... campon]*	Lista todos los campos que se van a reproyectar.
reproject_type	Streamdefault Especifique	Elija cómo reproyectar los campos.
coordinate_system	*string*	El nombre del sistema de coordenadas que se aplicará a los campos. Ejemplo: `set reprojectnode.coordinate_system = "WGS_1984_World_Mercator"`

Propiedades de restructurenode

El nodo Reestructurar convierte un campo nominal o marca en un grupo de campos que se puede rellenar con los valores todavía de otro campo. Por ejemplo, para un campo determinado llamado *tipo de pago*, con valores de *crédito*, *efectivo*, y *débito*, se crearían tres campos nuevos (*crédito*, *efectivo*, *débito*), que contendría cada uno el valor del pago real realizado.

Ejemplo

```
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"])
```

Tabla 82. Propiedades de restructurenode

Propiedad de restructurenode	Tipo de datos	Descripción de la propiedad
fields_from	[*category category category*] all	
include_field_name	*flag*	Indica si se debe usar el nombre del campo en el nombre de campo reestructurado.
value_mode	OtherFields Flags	Indica el modo de definir los valores para los campos reestructurados. Con OtherFields, debe especificar los campos que se van a usar (consulte a continuación). Con Flags, los valores son marcas numéricas.
value_fields	*lista*	Es necesario en caso de que value_mode sea OtherFields. Especifica los campos que se van a usar como campos de valores.

Propiedades de rfmanalysisnode

El nodo Análisis de RFM (actualidad, frecuencia, monetario) permite determinar cuantitativamente qué clientes son los mejores examinando cuándo ha sido la compra más reciente de un cliente (actualidad), cuántas veces suele comprar (frecuencia) y cuánto gasta el cliente en todas las transacciones (valor monetario).

Ejemplo

```
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])
```

Tabla 83. Propiedades de rfmanalysisnode

Propiedades de rfmanalysisnode	Tipo de datos	Descripción de la propiedad
actualidad	*campo*	Especifica el campo de actualidad. Puede ser una fecha, marca de tiempo o un número simple.
frecuencia	*campo*	Especifica el campo de frecuencia.
monetary	*campo*	Especifica el campo de monetario.
recency_bins	*entero*	Especifica el número de intervalos de actividades recientes que se van a generar.
recency_weight	*número*	Especifica la ponderación que se aplicará a los datos de actividades recientes. El valor predeterminado es 100.
frequency_bins	*entero*	Especifica el número de intervalos de frecuencia que se van a generar.

Tabla 83. Propiedades de rfmanalysisnode (continuación)

Propiedades de rfmanalysisnode	Tipo de datos	Descripción de la propiedad
frequency_weight	*número*	Especifica la ponderación que se aplicará a los datos de frecuencia. El valor por omisión es 10.
monetary_bins	*entero*	Especifica el número de intervalos de monetario que se van a generar.
monetary_weight	*número*	Especifica la ponderación que se aplicará a los datos de monetario. El valor predeterminado es 1.
tied_values_method	Next Actual	Especifica en qué intervalo se van a insertar los datos de valor empatado.
recalculate_bins	Always IfNecessary	
add_outliers	*flag*	Sólo está disponible si recalculate_bins se define como IfNecessary. Si se selecciona, los registros por debajo del intervalo más inferior se añaden al intervalo inferior y los registros por encima, se añaden al intervalo superior.
binned_field	Recency Frequency Monetary	
recency_thresholds	*valor valor*	Sólo está disponible si recalculate_bins se define como Siempre. Especifica los umbrales superior e inferior de los intervalos de actividades recientes. El umbral superior de un intervalo se utiliza como el umbral inferior del siguiente, por ejemplo, [10 30 60] definiría dos intervalos, el primer intervalo con los umbrales superior e inferior de 10 y 30, con los umbrales del segundo intervalo de 30 y 60.
frequency_thresholds	*valor valor*	Sólo está disponible si recalculate_bins se define como Siempre.
monetary_thresholds	*valor valor*	Sólo está disponible si recalculate_bins se define como Siempre.

Propiedades de settoflagnode

El nodo Marcas deriva varios campos de marcas en función de los valores categóricos definidos para uno o más campos nominales.

Ejemplo

```
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"])
```

Tabla 84. Propiedades de settoflagnode

Propiedad de settoflagnode	Tipo de datos	Descripción de la propiedad
fields_from	[*category category category*] all	
true_value	*cadena*	Especifica el valor para verdadero utilizado por el nodo al configurar una marca. El valor predeterminado es T (del inglés 'True').
false_value	*cadena*	Especifica el valor para falso utilizado por el nodo al configurar una marca. El valor predeterminado es F (del inglés 'False').
use_extension	*flag*	Utiliza una extensión como sufijo o prefijo para el nuevo campo de marca.
extensión	*cadena*	
add_as	Suffix Prefix	Especifica si la extensión es un sufijo o un prefijo.
aggregate	*flag*	Agrupa registros en función de campos clave. Si algún registro se establece como verdadero, se activarán todos los campos de marca de un grupo.
keys	*lista*	Campos clave.

Propiedades de statisticstransformnode

El nodo Transformación Statistics ejecuta una selección de comandos de sintaxis de IBM SPSS Statistics en los orígenes de datos de IBM SPSS Modeler. Este nodo requiere una copia de IBM SPSS Statistics con licencia.

Las propiedades de este nodo están descritas en “Propiedades de statisticstransformnode” en la página 301.

Propiedades de timeintervalsnode

El nodo Intervalos de tiempo especifica intervalos y genera etiquetas (si es necesario) para modelar los datos de series temporales. Si los valores no están espaciados de manera uniforme, el nodo puede rellenar o agregar valores según sea necesario para crear un intervalo uniforme entre registros.

Ejemplo

```
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")
```

Tabla 85. Propiedades de timeintervalsnode.

Propiedad de **timeintervalsnode**	Tipo de datos	Descripción de la propiedad
interval_type	Ninguno Períodos CyclicPeriods Años Trimestres Meses DaysPerWeek DaysNonPeriodic HoursPerDay HoursNonPeriodic MinutesPerDay MinutesNonPeriodic SecondsPerDay SecondsNonPeriodic	
mode	Label Create	Determina si desea etiquetar los registros de manera consecutiva o crear la serie según una fecha, marca de tiempo o campo de tiempo concretos.
campo	*campo*	Al crear la serie a partir de los datos, especifica el campo que informa de la fecha u hora de cada registro.
period_start	*entero*	Especifica el intervalo de inicio para períodos o períodos cíclicos.
cycle_start	*entero*	Ciclo de inicio de los períodos cíclicos.
year_start	*entero*	En el caso de los tipos de intervalo que procedan, el año en que el primer intervalo tiene lugar.
quarter_start	*entero*	En el caso de los tipos de intervalo que procedan, el trimestre en que el primer intervalo tiene lugar.
month_start	Enero Febrero Marzo Abril Mayo Junio Julio Agosto Septiembre Octubre Noviembre Diciembre	
day_start	*entero*	
hour_start	*entero*	

Tabla 85. Propiedades de timeintervalsnode (continuación).

Propiedad de **timeintervalsnode**	Tipo de datos	Descripción de la propiedad
minute_start	*entero*	
second_start	*entero*	
periods_per_cycle	*entero*	En el caso de los períodos cíclicos, número en cada ciclo.
fiscal_year_begins	Enero Febrero Marzo Abril Mayo Junio Julio Agosto Septiembre Octubre Noviembre Diciembre	En el caso de los intervalos trimestrales, especifica el mes en el que comienza el año fiscal.
week_begins_on	Sunday Monday Tuesday Wednesday Thursday Friday Saturday Sunday	En el caso de los intervalos periódicos (días a la semana, horas al día, minutos al día y segundos al día), especifica el día en el que comienza la semana.
day_begins_hour	*entero*	En el caso de los intervalos periódicos (horas al día, minutos al día y segundos al día), especifica la hora a la que comienza el día. Se puede usar junto con day_begins_minute y day_begins_second para determinar una hora exacta, como *8:05:01*. Vea el ejemplo de uso que se muestra a continuación.
day_begins_minute	*entero*	En el caso de los intervalos periódicos (horas al día, minutos al día y segundos al día), especifica el minuto en el que comienza el día (por ejemplo, *5* en *8:05*).
day_begins_second	*entero*	En el caso de los intervalos periódicos (horas al día, minutos al día y segundos al día), especifica el segundo en el que comienza el día (por ejemplo, *17* en *8:05:17*).
days_per_week	*entero*	En el caso de los intervalos periódicos (días a la semana, horas al día, minutos al día y segundos al día), especifica el número de días a la semana.
hours_per_day	*entero*	En el caso de los intervalos periódicos (horas al día, minutos al día y segundos al día), especifica el número de horas al día.

Tabla 85. Propiedades de timeintervalsnode (continuación).

Propiedad de **timeintervalsnode**	Tipo de datos	Descripción de la propiedad
interval_increment	1 2 3 4 5 6 10 15 20 30	En el caso de los minutos al día y de los segundos al día, especifica el número de minutos o segundos que se va a aumentar en cada registro.
field_name_extension	*cadena*	
field_name_extension_as_prefix	*tag*	
date_format	"DDMMAA" "MMDDYY" "AAMMDD" "YYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-AAAA" "DD-MES-YY" "DD-MES-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.AAAA" "MM.DD.YYYY" "DD.MES.YY" "DD.MES.YYYY" "DD/MM/YY" "DD/MM/AAAA" "MM/DD/YY" "MM/DD/YYYY" "DD/MES/YY" "DD/MES/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"	

Tabla 85. Propiedades de timeintervalsnode (continuación).

Propiedad de **timeintervalsnode**	Tipo de datos	Descripción de la propiedad
aggregate	Media Sum Mode Mín Máx Primero Last TrueIfAnyTrue	Especifica el método de agregación para un campo.
pad	Vacío MeanOfRecentPoints True False	Especifica el método de relleno para un campo.
agg_mode	Todos Especifica	Determina si se agregan o rellenan todos los campos con las funciones predeterminadas según sea necesario o bien si se especifican los campos y las funciones que deben usarse.
agg_range_default	Media Sum Mode Mín Máx	Especifica la función predeterminada que se va a usar al agregar campos continuos.
agg_set_default	Mode Primero Last	Especifica la función predeterminada que se va a usar al agregar campos nominales.
agg_flag_default	TrueIfAnyTrue Mode Primero Last	
pad_range_default	Vacío MeanOfRecentPoints	Especifica la función predeterminada que se va a usar al agregar campos continuos.
pad_set_default	Vacío MostRecentValue	
pad_flag_default	Vacío True False	
max_records_to_create	*entero*	Especifica el número máximo de registros que se van a crear al rellenar la serie.
estimation_from_beginning	*tag*	
estimation_to_end	*tag*	
estimation_start_offset	*entero*	
estimation_num_holdouts	*entero*	
create_future_records	*tag*	
num_future_records	*entero*	
create_future_field	*tag*	
future_field_name	*cadena*	

Propiedades de transposenode

El nodo Transponer intercambia los datos en filas y columnas de manera que los registros se conviertan en campos y los campos en registros.

Ejemplo

```
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")
```

Tabla 86. Propiedades de transposenode

Propiedad de transposenode	Tipo de datos	Descripción de la propiedad
transposed_names	Prefix Leer	Se pueden generar nuevos nombres de campo automáticamente a partir de un prefijo concreto o bien se pueden leer desde un campo existente en los datos.
prefix	*cadena*	
num_new_fields	*entero*	Al usar un prefijo, especifica el número máximo de campos nuevos que se van a crear.
read_from_field	*campo*	Campo del que se leen los nombres. Debe tratarse de un campo instanciado o, de lo contrario, se producirá un error al ejecutar el nodo.
max_num_fields	*entero*	Al leer nombres de un campo, especifica un límite superior con el fin de evitar que se genere un número desmesurado de campos.
transpose_type	Numérico Cadena Personalizado	De forma predeterminada, solamente los campos continuos (rango numérico) se transponen, si bien se puede elegir un subconjunto personalizado de campos numéricos o, en su lugar, transponer todos los campos de cadena.
transpose_fields	*lista*	Especifica los campos que se van a transponer cuando se usa la opción Custom.
id_field_name	*campo*	

Propiedades de typenode

El nodo Tipo especifica propiedades y metadatos de campo. Por ejemplo, puede especificar un nivel de medición (continuo, nominal, ordinal o marca) para cada campo, establecer las opciones para gestionar valores perdidos y nulos del sistema, establecer el rol de un campo con fines de modelado, especificar las etiquetas de valor y campo y especificar los valores de un campo.

Ejemplo

```
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"]])
```

Observe que en algunos casos puede que sea necesario instanciar totalmente el nodo Tipo para que otros nodos funcionen adecuadamente, como, por ejemplo, la propiedad`fields from` del nodo Marcas. Simplemente conecte un nodo Tabla y ejecútelo para instanciar los campos:

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

Tabla 87. Propiedades de typenode.

Propiedad de typenode	Tipo de datos	Descripción de la propiedad

`dirección`	`Input`
	`Destino`
	`Both`
	`Ninguno`
	`Partition`
	`Split`
	`Frequency`
	`RecordID`
		Propiedad con clave para los roles de los campos.
		Nota: Los valores `In` y `Out` han quedado en desuso. Deberán de ser compatibles en versiones posteriores.

`type`	`Range`
	`Flag`
	`Set`
	`Sin tipo`
	`Discrete`
	`OrderedSet`
	`Predeterminado`
		El nivel de medición del campo (anteriormente denominado el "tipo" de campo). Si se establece `type` a `Default`, se borra `Specify`, se reestablecerá a `Read`.
		Si se establece `value_mode` a `Pass` o `Read`, el establecimiento de `type` no afectará a `value_mode`.
		Nota: Los tipos de datos utilizados internamente difieren de los que son visibles en el nodo tipo. La correspondencia es la siguiente: Range -> Continuous Set - > Nominal OrderedSet -> Ordinal Discrete- > Categorical

`almacenamiento`	`Desconocido`
	`Cadena`
	`Entero`
	`Real`
	`Hora`
	`Fecha`
	`Marca de tiempo`
		Propiedad con clave de solamente lectura para el tipo de almacenamiento de campos.

`check`	`Ninguno`
	`Nullify`
	`Coerce`
	`Descartar`
	`Warn`
	`Abort`
		Propiedad con clave para la comprobación del rango y el tipo de campo.

Tabla 87. Propiedades de typenode (continuación).

Propiedad de typenode	Tipo de datos	Descripción de la propiedad
values	[*value value*]	Para un campo continuos, el primer valor es el mínimo y el último es el máximo. Para campos nominales, especifique todos los valores. Para los campos marca, el primer valor representa *falso* y el último, *verdadero*. La configuración de esta propiedad establece de forma automática la propiedad value_mode en Specify.
value_mode	Leer Pasar Leer+ Actual Especifique	Determina la forma en la que se establecen los valores. Tenga en cuenta que no puede establecer esta propiedad directamente en Specify. Para utilizar valores específicos, establezca la propiedad values.
extend_values	*flag*	Se aplica cuando value_mode se establece en Read. Establézcala en T para añadir nuevos valores de lectura a los valores existentes del campo. Establézcala en F para descartar los valores existentes y favorecer a los nuevos valores de lectura.
enable_missing	*flag*	Cuando está definida como T, activa el seguimiento de los valores perdidos para el campo.
missing_values	[*value value ...*]	Especifica los valores de datos que denotan los datos perdidos.
range_missing	*flag*	Especifica si se ha definido un rango de valores perdidos (vacíos) para un campo.
missing_lower	*string*	Si range_missing es verdadero, especifica el límite inferior del rango de valores perdidos.
missing_upper	*string*	Si range_missing es verdadero, especifica el límite superior del rango de valores perdidos.
null_missing	*flag*	Cuando se establece en T, los valores *nulos* (valores no definidos que se muestran como $null$ en el software) se consideran valores perdidos.
whitespace_ missing	*flag*	Cuando se establece en T, los valores que sólo contienen un espacio en blanco (espacios, tabulaciones y líneas nuevas) se consideran valores perdidos.
description	*string*	Especifica la descripción de un campo.
value_labels	*[[Valor CadenaEtiqueta] [Valor CadenaEtiqueta] ...]*	Se utiliza para especificar etiquetas para los pares de valores.
display_places	*entero*	Establece el número de cifras decimales para el campo cuando se muestra (sólo se aplica a campos con almacenamiento REAL). Un valor de 1 utilizará el valor predeterminado de la ruta.
export_places	*entero*	Establece el número de cifras decimales para el campo cuando se exporta (sólo se aplica a campos con almacenamiento REAL). Un valor de 1 utilizará el valor predeterminado de la ruta.

Tabla 87. Propiedades de typenode (continuación).

Propiedad de typenode	Tipo de datos	Descripción de la propiedad
decimal_separator	DEFAULT PERIOD COMMA	Establece el separador decimal para el campo (sólo se aplica a campos con almacenamiento REAL).
date_format	"DDMMAA" "MMDDYY" "AAMMDD" "YYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-AAAA" "DD-MES-YY" "DD-MES-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.AAAA" "MM.DD.YYYY" "DD.MES.YY" "DD.MES.YYYY" "DD/MM/YY" "DD/MM/AAAA" "MM/DD/YY" "MM/DD/YYYY" "DD/MES/YY" "DD/MES/YYYY" MON YYYY q Q YYYY ww WK YYYY	Establece el formato de fecha para el campo (sólo se aplica a campos con almacenamiento FECHA o MARCADETIEMPO).
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"	Establece el formato de hora para el campo (sólo se aplica a campos con almacenamiento TIME o TIMESTAMP).
number_format	DEFAULT STANDARD SCIENTIFIC CURRENCY	Establece el formato de presentación de los números para el campo.
standard_places	*entero*	Establece el número de cifras decimales para el campo cuando se muestra en formato estándar. Un valor de 1 utilizará el valor predeterminado de la ruta. Tenga en cuenta que el intervalo display_places existente también cambiará esto, pero ahora se ha desaprobado.

Tabla 87. Propiedades de typenode (continuación).

Propiedad de typenode	Tipo de datos	Descripción de la propiedad
scientific_places	*entero*	Establece el número de cifras decimales para el campo cuando se muestra en formato científico. Un valor de 1 utilizará el valor predeterminado de la ruta.
currency_places	*entero*	Establece el número de cifras decimales para el campo cuando se muestra en formato moneda. Un valor de 1 utilizará el valor predeterminado de la ruta.
grouping_symbol	DEFAULT NONE LOCALE PERIOD COMMA SPACE	Establece el símbolo de agrupación para el campo.
column_width	*entero*	Establece el ancho de columna para el campo. Un valor de 1 establecerá el ancho de columna en Auto.
justify	AUTO CENTER LEFT RIGHT	Establece la justificación de columna para el campo.
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	Esta propiedad con clave es similar a type en cuanto a que puede utilizarse para definir la medición asociada al campo. La diferencia es que, en los scripts Python, la función de establecimiento puede pasar también uno de los valores MeasureType, mientras que la función de obtención siempre devolverá los valores MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Para campos de recopilación (listas con profundidad 0), esta propiedad con clave define el tipo de medición asociado con los valores subyacentes.
geo_type	Point Multipunto Cadena lineal Cadena multilínea Polígono Multipolígono	En campos geoespaciales, esta propiedad con clave define el tipo del objeto geoespacial representado por este campo. Debería ser coherente con la profundidad de lista de los valores.
has_coordinate_system	*booleano*	En campos geoespaciales, esta propiedad define si este campo tiene un sistema de coordenadas
coordinate_system	*string*	En campos geoespaciales, esta propiedad con clave define el sistema de coordenadas para este campo.

Tabla 87. Propiedades de typenode (continuación).

Propiedad de typenode	Tipo de datos	Descripción de la propiedad
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	Esta propiedad con clave es similar a custom_storage en cuanto a que puede utilizarse para definir el almacenamiento de alteración temporal para el campo. La diferencia es que, en los scripts Python, la función de establecimiento puede pasar también uno de los valores StorageType, mientras que la función de obtención siempre devolverá los valores StorageType.
custom_list_ storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	Para campos de lista, esta propiedad con clave especifica el tipo de almacenamiento de los valores subyacentes.
custom_list_depth	*entero*	Para campos de lista, esta propiedad con clave especifica la profundidad del campo.

Capítulo 12. Propiedades de nodos Gráfico

Propiedades comunes de nodos Gráfico

Esta sección describe las propiedades disponibles para los nodos de gráficos, incluidas las comunes y aquellas específicas de cada tipo de nodo.

Tabla 88. Propiedades comunes de nodos de gráficos

Propiedades comunes de nodos de gráficos	Tipo de datos	Descripción de la propiedad
título	*string*	Especifica el título. Ejemplo: "Esto es un título".
caption	*string*	Especifica el pie. Por ejemplo: "Esto es un pie".
output_mode	Screen File	Determina si el resultado del nodo de gráficos se muestra o si se guarda en un archivo.
output_format	BMP JPEG PNG HTML output (*.cou*)	Especifica el tipo de resultado. El tipo exacto de resultado permitido para cada nodo varía.
full_filename	*string*	Especifica la ruta de destino y el nombre de archivo del resultado generado por el nodo de gráficos.
use_graph_size	*marca*	Controla si el tamaño del gráfico se ha establecido de manera explícita mediante las propiedades de ancho y altura a continuación. Afecta solamente a los gráficos que tienen salida a pantalla. No disponible para el nodo Distribución.
graph_width	*número*	Cuando use_graph_size es True, establece el ancho del gráfico en píxeles.
graph_height	*número*	Cuando use_graph_size es True, establece la altura del gráfico en píxeles.

Desactivación de los campos opcionales

Los campos opcionales, como un campo de superposición para gráficos, se pueden desactivar estableciendo el valor de la propiedad en "" (cadena vacía), tal y como se muestra en el siguiente ejemplo:

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

Especificación de colores

Los colores de los títulos, pies, fondos y etiquetas se pueden especificar mediante las cadenas hexadecimales que comiencen con el símbolo almohadilla (#). Por ejemplo, para establecer el fondo del gráfico en cielo azul, debe utilizar la siguiente instrucción:

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

Aquí, los dos primeros dígitos, 87, especifican el contenido rojo, los dos del medio, CE, especifican el contenido verde y los dos últimos, EB, el contenido azul. Cada dígito puede tomar un valor del rango 0-9 o A-F. Juntos, estos valores pueden especificar red-green-blue, o RGB o color.

Nota: Al especificar colores en RVA, puede utilizar el selector de campos en la interfaz de usuario para determinar el código de color correcto. Basta con colocarse sobre el color para ver un texto con la información deseada.

Propiedades de collectionnode

El nodo Colección muestra la distribución de valores de un campo numérico relativo a los valores de otro. (Crea gráficos parecidos a los histogramas.) Es útil para ilustrar una variable o un campo cuyos valores cambian con el tiempo. Con los gráficos 3D también puede incluir un eje simbólico que muestra las distribuciones por categoría.

Ejemplo

```
node = stream.create("collection", "My node")
# Pestaña "Gráfico"
node.setPropertyValue("three_D", True)
node.setPropertyValue("collect_field", "Drug")
node.setPropertyValue("over_field", "Age")
node.setPropertyValue("by_field", "BP")
node.setPropertyValue("operation", "Sum")
# Sección "Superponer"
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# pestaña "Opciones"
node.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)
```

Tabla 89. Propiedades de collectionnode

Propiedad de collectionnode	Tipo de datos	Descripción de la propiedad
over_field	*campo*	
over_label_auto	*flag*	
over_label	*cadena*	
collect_field	*campo*	
collect_label_auto	*flag*	
collect_label	*cadena*	
three_D	*flag*	
by_field	*campo*	
by_label_auto	*flag*	
by_label	*cadena*	
operation	Sum Media Mín Máx SDev	
color_field	*cadena*	
panel_field	*cadena*	
animation_field	*cadena*	

Tabla 89. Propiedades de collectionnode (continuación)

Propiedad de collectionnode	Tipo de datos	Descripción de la propiedad
range_mode	Automatic UserDefined	
range_min	*número*	
range_max	*número*	
bins	ByNumber ByWidth	
num_bins	*número*	
bin_width	*número*	
use_grid	*flag*	
graph_background	*color*	Al principio de esta sección se describen los colores de gráficos estándar.
page_background	*color*	Al principio de esta sección se describen los colores de gráficos estándar.

Propiedades de distributionnode

El nodo Distribución muestra las instancias de valores simbólicos (categóricos), como el tipo de hipoteca o el género. Normalmente, podría usar el nodo Distribución para mostrar los desequilibrios de los datos, que pueden rectificarse mediante el nodo Equilibrar antes de crear un modelo.

Ejemplo

```
node = stream.create("distribution", "My node")
# Pestaña "Gráfico"
node.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)
```

Tabla 90. Propiedades de distributionnode

Propiedades de distributionnode	Tipo de datos	Descripción de la propiedad
plot	SelectedFields Flags	
x_field	*campo*	
color_field	*campo*	Campo de superposición.
normalize	*flag*	
sort_mode	ByOccurence Alphabetic	
use_proportional_scale	*flag*	

Propiedades de evaluationnode

El nodo Evaluación ayuda a evaluar y comparar modelos predictivos. El diagrama de evaluación muestra la calidad con que los modelos predicen resultados particulares. Ordena registros en función del valor predicho y la confianza de la predicción. Divide el registro en grupos de igual tamaño (**cuantiles**) y, a continuación, representa el valor del criterio de negocio de cada cuantil de mayor a menor. El gráfico muestra múltiples modelos como líneas independientes.

Ejemplo

```
node = stream.create("evaluation", "My node")
# Pestaña "Gráfico"
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")
```

Tabla 91. Propiedades de evaluationnode.

Propiedad de evaluationnode	Tipo de datos	Descripción de la propiedad
chart_type	Gains Response Lift Profit ROI ROC	
inc_baseline	*tag*	
field_detection_method	Metadata Name	
use_fixed_cost	*tag*	
cost_value	*number*	
cost_field	*cadena*	
use_fixed_revenue	*tag*	
revenue_value	*number*	
revenue_field	*cadena*	
use_fixed_weight	*tag*	
weight_value	*number*	
weight_field	*campo*	

Tabla 91. Propiedades de evaluationnode (continuación).

Propiedad de evaluationnode	Tipo de datos	Descripción de la propiedad
n_tile	Quartiles Quintles Deciles Vingtiles Percentiles 1000-tiles	
cumulative	*tag*	
style	Line Point	
point_type	Rectángulo Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
export_data	*tag*	
data_filename	*cadena*	
delimiter	*cadena*	
new_line	*tag*	
inc_field_names	*tag*	
inc_best_line	*tag*	
inc_business_rule	*tag*	
business_rule_condition	*cadena*	
plot_score_fields	*tag*	
score_fields	*[campo1 ... campoN]*	
target_field	*campo*	
use_hit_condition	*tag*	
hit_condition	*cadena*	
use_score_expression	*tag*	
score_expression	*cadena*	
caption_auto	*tag*	

Propiedades de graphboardnode

El nodo Tablero ofrece muchos tipos diferentes de gráficos en un único nodo. Con este nodo puede seleccionar los campos de datos que desee explorar y seleccionar un gráfico de los disponibles para los datos seleccionados. El nodo filtra automáticamente cualquier tipo de gráfico que no sea compatible con las selecciones de campo.

Nota: Si establece una propiedad que no es válida para el tipo de gráfico (por ejemplo, si especifica `y_field` para un histograma), se ignora dicha propiedad.

Nota: En la IU, en la pestaña Detallado de muchos tipos de gráfico distintos, hay un campo **Resumen**; los scripts no soportan dicho campo en la actualidad.

Ejemplo

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

Tabla 92. Propiedades de graphboardnode

Propiedades de **graphboard**	Tipo de datos	Descripción de la propiedad
graph_type	2DDotplot 3DArea 3DBar 3DDensity 3DHistogram 3DPie 3DScatterplot Área ArrowMap Bar BarCounts BarCountsMap BarMap BinnedScatter Diagramas de caja Bubble ChoroplethMeans ChoroplethMedians ChoroplethSums ChoroplethValues ChoroplethCounts CoordinateMap CoordinateChoroplethMeans CoordinateChoroplethMedians CoordinateChoroplethSums CoordinateChoroplethValues CoordinateChoroplethCounts Dotplot Heatmap HexBinScatter Histograma Line LineChartMap LineOverlayMap Parallel Path Pie PieCountMap PieCounts PieMap PointOverlayMap PolygonOverlayMap Ribbon Scatterplot SPLOM Surface	Identifica el tipo de gráfico.
x_field	*campo*	Especifica una etiqueta personalizada para el eje *x*. Disponible solamente para etiquetas.
y_field	*campo*	Especifica una etiqueta personalizada para el eje *y*. Disponible solamente para etiquetas.
z_field	*campo*	Se utiliza en algunos gráficos 3D.
color_field	*campo*	Se utiliza en mapas de calor.

Tabla 92. Propiedades de graphboardnode (continuación)

Propiedades de graphboard	Tipo de datos	Descripción de la propiedad
size_field	*campo*	Se utiliza en gráficos de burbujas.
categories_field	*campo*	
values_field	*campo*	
rows_field	*campo*	
columns_field	*campo*	
campos	*campo*	
start_longitude_field	*campo*	Se utiliza con flechas en un mapa de referencia.
end_longitude_field	*campo*	
start_latitude_field	*campo*	
end_latitude_field	*campo*	
data_key_field	*campo*	Se utiliza en diversos mapas.
panelrow_field	*cadena*	
panelcol_field	*cadena*	
animation_field	*cadena*	
longitude_field	*campo*	Se utiliza en mapas de coordenadas.
latitude_field	*campo*	
map_color_field	*campo*	

Propiedades de histogramnode

El nodo Histograma muestra las instancias de valores de los campos numéricos. Se suele utilizar para explorar los datos antes de las manipulaciones y la generación de modelos. Al igual que con el nodo Distribución, con frecuencia el nodo Histograma detecta desequilibrios en los datos.

Ejemplo

```
node = stream.create("histogram", "My node")
# Pestaña "Gráfico"
node.setPropertyValue("field", "Drug")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# pestaña "Opciones"
node.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)
```

Tabla 93. Propiedades de histogramnode

Propiedad de histogramnode	Tipo de datos	Descripción de la propiedad
field	*campo*	
color_field	*campo*	

Tabla 93. Propiedades de histogramnode (continuación)

Propiedad de histogramnode	Tipo de datos	Descripción de la propiedad
panel_field	*campo*	
animation_field	*campo*	
range_mode	Automatic UserDefined	
range_min	*número*	
range_max	*número*	
bins	ByNumber ByWidth	
num_bins	*número*	
bin_width	*número*	
normalize	*flag*	
separate_bands	*flag*	
x_label_auto	*flag*	
x_label	*cadena*	
y_label_auto	*flag*	
y_label	*cadena*	
use_grid	*flag*	
graph_background	*color*	Al principio de esta sección se describen los colores de gráficos estándar.
page_background	*color*	Al principio de esta sección se describen los colores de gráficos estándar.
normal_curve	*flag*	Indica si se debe mostrar la curva de distribución normal en los resultados.

Propiedades de multiplotnode

El nodo G. múltiple crea un gráfico que muestra varios campos *Y* sobre un campo *X* único. Los campos *Y* están representados como líneas coloreadas; cada uno equivale a un nodo Gráfico con el estilo establecido en **Línea** y el Modo para X establecido en **Ordenar**. Los gráficos múltiples son útiles cuando quiere explorar la fluctuación de varias variables a través del tiempo.

Ejemplo

```
node = stream.create("multiplot", "My node")
# Pestaña "Gráfico"
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("y_fields", ["Drug", "BP"])
node.setPropertyValue("panel_field", "Sex")
# Sección "Superponer"
node.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")
```

Tabla 94. Propiedades de multiplotnode

Propiedad de multiplotnode	Tipo de datos	Descripción de la propiedad
x_field	*campo*	
y_fields	*lista*	
panel_field	*campo*	
animation_field	*campo*	
normalize	*flag*	
use_overlay_expr	*flag*	
overlay_expression	*cadena*	
records_limit	*número*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*flag*	
x_label	*cadena*	
y_label_auto	*flag*	
y_label	*cadena*	
use_grid	*flag*	
graph_background	*color*	Al principio de esta sección se describen los colores de gráficos estándar.
page_background	*color*	Al principio de esta sección se describen los colores de gráficos estándar.

Propiedades de plotnode

El nodo Gráfico muestra la relación entre los campos numéricos. Puede crear un gráfico mediante puntos (un diagrama de dispersión) o líneas.

Ejemplo

```
node = stream.create("plot", "My node")
# Pestaña "Gráfico"
node.setPropertyValue("three_D", True)
node.setPropertyValue("x_field", "BP")
node.setPropertyValue("y_field", "Cholesterol")
node.setPropertyValue("z_field", "Drug")
# Sección "Superponer"
node.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")
# Pestaña "Resultados"
node.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "JPEG")
node.setPropertyValue("full_filename", "C:/temp/graph_output/plot_output.jpeg")
```

Tabla 95. Propiedades de plotnode.

Propiedad de plotnode	Tipo de datos	Descripción de la propiedad
x_field	*campo*	Especifica una etiqueta personalizada para el eje *x*. Disponible solamente para etiquetas.
y_field	*campo*	Especifica una etiqueta personalizada para el eje *y*. Disponible solamente para etiquetas.
three_D	*tag*	Especifica una etiqueta personalizada para el eje *y*. Disponible sólo para etiquetas en gráficos 3D.
z_field	*campo*	
color_field	*campo*	Campo de superposición.
size_field	*campo*	
shape_field	*campo*	
panel_field	*campo*	Especifica un campo de marcas o nominal para crear un gráfico independiente para cada categoría. Los gráficos aparecerán juntos en una ventana de resultados.
animation_field	*campo*	Especifica un campo de marcas o nominal para ilustrar las categorías de los valores de datos creando una serie de gráficos secuenciados mediante la animación.
transp_field	*campo*	Especifica un campo para ilustrar las categorías de los valores de datos utilizando un nivel de transparencia distinto para cada categoría. No disponible para gráficos de líneas.
overlay_type	Ninguno Smoother Función	Determina si se muestra una función superpuesta o suavizamiento LOESS.
overlay_expression	*cadena*	Especifica la expresión utilizada cuando overlay_type se establece en Function.
style	Point Line	
point_type	Rectángulo Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
x_mode	Ordenar Overlay AsRead	

Tabla 95. Propiedades de plotnode (continuación).

Propiedad de plotnode	Tipo de datos	Descripción de la propiedad
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*	
z_range_max	*number*	
jitter	*tag*	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*tag*	
x_label	*cadena*	
y_label_auto	*tag*	
y_label	*cadena*	
z_label_auto	*tag*	
z_label	*cadena*	
use_grid	*tag*	
graph_background	*color*	Al principio de esta sección se describen los colores de gráficos estándar.
page_background	*color*	Al principio de esta sección se describen los colores de gráficos estándar.
use_overlay_expr	*tag*	Desaprobado en favor de overlay_type.

Propiedades de timeplotnode

El nodo Gráfico de tiempo muestra uno o más conjuntos de datos de series temporales. Normalmente, primero se utilizaría un nodo Intervalos de tiempo para crear un campo *EtiquetaTiempo*, que se utilizaría para etiquetar el eje *x*.

Ejemplo

```
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 settings
node.setPropertyValue("symbol_size", 2.0)
```

Tabla 96. Propiedades de timeplotnode.

Propiedad de **timeplotnode**	Tipo de datos	Descripción de la propiedad
plot_series	Series Models	
use_custom_x_field	*tag*	
x_field	*campo*	
y_fields	*lista*	
panel	*tag*	
normalize	*tag*	
line	*tag*	
points	*tag*	
point_type	Rectángulo Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
suavizamiento	*tag*	Puede añadir suavizamientos al gráfico únicamente si establece panel en True.
use_records_limit	*tag*	
records_limit	*entero*	
symbol_size	*number*	Especifica el tamaño del símbolo.
panel_layout	Horizontal Vertical	

Propiedades de webnode

El nodo Malla muestra la fuerza de las relaciones entre los valores de dos o más campos simbólicos (categóricos). El gráfico utiliza líneas de varios anchos para indicar la fuerza de la conexión. Podría utilizar un nodo Malla, por ejemplo, para explorar las relaciones existentes entre la compra de un conjunto de elementos en un sitio de comercio electrónico.

Ejemplo

```
node = stream.create("web", node "My ") # Pestaña "Gráfico"
node.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)
# pestaña "Opciones"
node.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")
```

Tabla 97. Propiedades de webnode

Propiedad de **webnode**	Tipo de datos	Descripción de la propiedad
use_directed_web	*flag*	
campos	*lista*	
to_field	*campo*	
from_fields	*lista*	
true_flags_only	*flag*	
line_values	Absolute OverallPct PctLarger PctSmaller	
strong_links_heavier	*flag*	
num_links	ShowMaximum ShowLinksAbove ShowAll	
max_num_links	*número*	
links_above	*número*	
discard_links_min	*flag*	
links_min_records	*número*	
discard_links_max	*flag*	
links_max_records	*número*	
weak_below	*número*	
strong_above	*número*	
link_size_continuous	*flag*	
web_display	Circular Red Directed cuadrícula	
graph_background	*color*	Al principio de esta sección se describen los colores de gráficos estándar.
symbol_size	*número*	Especifica el tamaño del símbolo.

Capítulo 13. Propiedades de nodos de modelado

Propiedades comunes de nodos de modelado

Las siguientes propiedades son comunes a algunos o todos los nodos de modelado. Las excepciones se indican en la documentación de los nodos de modelado individuales según sea adecuado.

Tabla 98. Propiedades comunes de nodos de modelado

Propiedad	Valores	Descripción de la propiedad
`custom_fields`	*flag*	Si es verdadero, le permite especificar el objetivo, la entrada y otros campos del nodo actual. Si es falso, se utiliza la configuración actual de un nodo Tipo situado en un punto anterior de la ruta.

`objetivo`
o
`targets`
	campo
	
	o
	[*field1 ... fieldN*]
		Especifica un único campo objetivo o varios campos objetivo dependiendo del tipo de modelo.

`inputs`	[*field1 ... fieldN*]	Campos de entrada o predictor utilizados por el modelo.
`partición`	*campo*	
`use_partitioned_data`	*flag*	Si se ha definido un campo de partición, esta opción garantiza que sólo se utilizarán los datos de la partición de entrenamiento para la generación del modelo.
`use_split_data`	*flag*	
`splits`	*[campo1 ... campoN]*	Especifica el campo o campos para utilizar en el modelado de divisiones. Sólo funciona si `use_split_data` está establecido como `True`.
`use_frequency`	*flag*	Los modelos específicos utilizan campos de ponderación y frecuencia como se indica en cada tipo de modelo.
`frequency_field`	*campo*	
`use_weight`	*flag*	
`weight_field`	*campo*	
`use_model_name`	*flag*	
`model_name`	*cadena*	Nombre personalizado para nuevo modelo.
`mode`	`Simple` `Valores avanzados`	

Propiedades de anomalydetectionnode

El nodo Detección de anomalías identifica casos extraños, o valores atípicos, que no se ajustan a patrones de datos "normales". Con este nodo, es posible identificar valores atípicos aunque no se ajusten a ningún patrón previamente conocido o no se realice una búsqueda exacta.

Ejemplo

```
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)
```

Tabla 99. Propiedades de anomalydetectionnode

Propiedades de **anomalydetectionnode**	Valores	Descripción de la propiedad
inputs	*[campo1 ... campoN]*	Los modelos de detección de anomalías criban registros basándose en los campos de entrada especificados. No utilizan un campo objetivo. Los campos de ponderación y frecuencia tampoco se usan. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
mode	Expert Simple	
anomaly_method	IndexLevel PerRecords NumRecords	Especifica el método utilizado para determinar el valor de corte para marcar los registros como anómalos.
index_level	*número*	Especifica el valor de corte mínimo con el que se van a marcar anomalías.
percent_records	*número*	Establece el umbral para marcar registros de acuerdo al porcentaje de registros en los datos de entrenamiento.
num_records	*número*	Establece el umbral para marcar registros de acuerdo al número de registros en los datos de entrenamiento.
num_fields	*entero*	El número de campos de los que se va a informar por cada registro anómalo.
impute_missing_values	*flag*	
adjustment_coeff	*número*	Valor que se usa para equilibrar la ponderación relativa asignada a los campos categóricos y continuos al calcular la distancia.
peer_group_num_auto	*flag*	Calcula automáticamente el número de grupos de homólogos.
min_num_peer_groups	*entero*	Especifica el número mínimo de grupos de homólogos empleado cuando peer_group_num_auto se establece en True.
max_num_per_groups	*entero*	Especifica el número máximo de grupos de homólogos.
num_peer_groups	*entero*	Especifica el número de grupos de homólogos empleado cuando peer_group_num_auto se establece en False.

Tabla 99. Propiedades de anomalydetectionnode (continuación)

Propiedades de anomalydetectionnode	Valores	Descripción de la propiedad
noise_level	*número*	Determina el modo en que se tratan los valores atípicos durante el clúster. Especifique un valor entre 0 y 0,5.
noise_ratio	*número*	Especifica la parte de memoria asignada al componente que debe usarse para el almacenamiento en búfer de ruido. Especifique un valor entre 0 y 0,5.

Propiedades de apriorinode

El nodo Apriori extrae un conjunto de reglas de los datos y destaca aquellas reglas con un mayor contenido de información. Apriori ofrece cinco métodos diferentes para la selección de reglas y utiliza un sofisticado esquema de indización para procesar eficientemente grandes conjuntos de datos. En los problemas de mucho volumen, Apriori se entrena más rápidamente, no tiene un límite arbitrario para el número de reglas que puede retener y puede gestionar reglas que tengan hasta 32 precondiciones. Apriori requiere que todos los campos de entrada y salida sean categóricos, pero ofrece un mejor rendimiento ya que está optimizado para este tipo de datos.

Ejemplo

```
node = stream.create("apriori", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("partition", "Test")
# Para no transacionales
node.setPropertyValue("use_transactional_data", False)
node.setPropertyValue("consequents", ["Age"])
node.setPropertyValue("antecedents", ["BP", "Cholesterol", "Drug"])
# Para transacionales
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("content_field", "Drug")
# "Model" tab
node.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" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("evaluation", "ConfidenceRatio")
node.setPropertyValue("lower_bound", 7)
```

Tabla 100. Propiedades de apriorinode

Propiedades de apriorinode	Valores	Descripción de la propiedad
consequents	*campo*	Los modelos Apriori utilizan Consecuentes y Antecedentes en lugar de los campos estándar objetivo y de entrada. Los campos de ponderación y frecuencia no se usan. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
antecedents	[*field1 ... fieldN*]	
min_supp	*número*	
min_conf	*número*	
max_antecedents	*número*	
true_flags	*flag*	
optimize	Speed Memory	
use_transactional_data	*flag*	
contiguous	*flag*	
id_field	*cadena*	
content_field	*cadena*	
mode	Simple Valores avanzados	
evaluación	RuleConfidence DifferenceToPrior ConfidenceRatio InformationDifference NormalizedChiSquare	
lower_bound	*número*	
optimize	Speed Memory	Se utiliza para especificar si la generación del modelo se debe optimizar para la velocidad o la memoria.

Propiedades associationrulesnode

El nodo Reglas de asociación es similar al nodo Apriori; sin embargo, a diferencia de Apriori, el nodo Reglas de asociación puede procesar datos de lista. Además, el nodo Reglas de asociación puede utilizarse con IBM SPSS Analytic Server para procesar datos de gran tamaño y aprovechar el proceso en paralelo de mayor velocidad.

Tabla 101. Propiedades associationrulesnode

Propiedades associationrulesnode	Tipo de datos	Descripción de la propiedad
predicciones	*campo*	Los campos en esta lista sólo pueden aparecer como un predictor de una regla
condiciones	*[campo1...campoN]*	Los campos en esta lista sólo pueden aparecer como una condición de una regla
max_rule_conditions	*entero*	El número máximo de condiciones que pueden incluirse en una sola regla. Mínimo 1, máximo 9.

Tabla 101. Propiedades associationrulesnode (continuación)

Propiedades **associationrulesnode**	Tipo de datos	Descripción de la propiedad
max_rule_predictions	*entero*	El número máximo de predicciones que pueden incluirse en una sola regla. Mínimo 1, máximo 5.
max_num_rules	*entero*	El número máximo de reglas que pueden considerarse como parte de la generación de regla. Mínimo 1, máximo 10.000.
rule_criterion_top_n	Confianza Rulesupport Lift Conditionsupport Capacidad de despliegue	El criterio de reglas que determina el valor por el cual se eligen las reglas "N" superiores en el modelo.
true_flags	*Boolean*	Establecer en *Y* determina que sólo se considerarán los valores verdaderos para campos de distintivo durante la generación de la regla.
rule_criterion	*Boolean*	Establecer en *Y* determina que los valores del criterio de regla se utilizan para excluir reglas durante la generación de modelos.
min_confidence	*número*	0,1 a 100 - el valor de porcentaje para el nivel de confianza mínimo necesario para una regla producida por el modelo. Si el modelo genera una regla con un nivel de confianza inferior al valor especificado aquí, la regla se descarta.
min_rule_support	*número*	0,1 a 100 - el valor de porcentaje para el soporte de regla mínimo necesario para una regla producida por el modelo. Si el modelo genera una regla con un nivel de soporte de regla inferior al valor especificado, la regla se descarta.
min_condition_support	*número*	0,1 a 100 - el valor de porcentaje para el soporte de condición mínima necesaria para una regla producida por el modelo. Si el modelo genera una regla con un nivel de soporte de condición inferior al valor especificado, la regla se descarta.
min_lift	*entero*	1 a 10 – representa la elevación mínima necesaria para una regla producida por el modelo. Si el modelo genera una regla con un nivel de elevación inferior al valor especificado, la regla se descarta.
exclude_rules	*Boolean*	Se utiliza para seleccionar una lista de campos relacionados a partir de los cuales no desea que el modelo cree reglas. Ejemplo: set :gsarsnode.exclude_rules = [[[campo1,campo2, campo3]],[[campo4, campo5]]] - donde cada lista de campos separados por [] es una fila en la tabla.
num_bins	*entero*	Establezca el número de intervalos automáticos en los que se agrupan los campos continuos. Mínimo 2, máximo 10.

Tabla 101. Propiedades associationrulesnode (continuación)

Propiedades `associationrulesnode`	**Tipo de datos**	**Descripción de la propiedad**
`max_list_length`	*entero*	Se aplica a cualquier campo de lista del que no se conoce la longitud máxima. Los elementos de la lista hasta el número especificado aquí se incluyen en la generación de modelos; los elementos adicionales se descartan. Mínimo 1, máximo 100.
`output_confidence`	*Boolean*	
`output_rule_support`	*Boolean*	
`output_lift`	*Boolean*	
`output_condition_support`	*Boolean*	
`output_deployability`	*Boolean*	
`rules_to_display`	`upto` `all`	El número máximo de reglas a visualizar en las tablas de salida.
`display_upto`	*entero*	Si `upto` se establece en `rules_to_display`, establezca el número de reglas a visualizar en las tablas de salida. El mínimo es 1.
`field_transformations`	*Boolean*	
`records_summary`	*Boolean*	
`rule_statistics`	*Boolean*	
`most_frequent_values`	*Boolean*	
`most_frequent_fields`	*Boolean*	
`word_cloud`	*Boolean*	

`word_cloud_sort`	`Confianza`
	`Rulesupport`
	`Lift`
	`Conditionsupport`
	`Capacidad de despliegue`	

`word_cloud_display`	*entero*	Mínimo 1, máximo 20.
`max_predictions`	*entero*	El número máximo de reglas que se pueden aplicar a cada entrada de la puntuación.

`criterio`	`Confianza`
	`Rulesupport`
	`Lift`
	`Conditionsupport`
	`Capacidad de despliegue`
		Seleccione la medida utilizada para determinar la fuerza de las reglas.

`allow_repeats`	*Boolean*	Determine si las reglas con la misma predicción se incluyen en la puntuación.

`check_input`	`NoPredictions`
	`Predictions`
	`NoCheck`	

Propiedades de autoclassifiernode

El nodo Clasificador automático crea y compara varios modelos diferentes para obtener resultados binarios (sí o no, abandono o no de clientes, etc.), lo que le permite seleccionar el mejor enfoque para un análisis determinado. Son compatibles varios algoritmos de modelado, por lo que es posible seleccionar los métodos que desee utilizar, las opciones específicas para cada uno y los criterios para comparar los resultados. El nodo genera un conjunto de modelos basado en las opciones especificadas y clasifica los mejores candidatos en función de los criterios que especifique.

Ejemplo

```
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)
```

Tabla 102. Propiedades de autoclassifiernode.

Propiedades de autoclassifiernode	Valores	Descripción de la propiedad
objetivo	*campo*	En objetivos de marca, el nodo Clasificador binario requiere un único campo objetivo y uno o más campos de entrada. También se puede especificar campos de ponderación y frecuencia. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
ranking_measure	Exactitud Area_under_curve Profit Lift Num_variables	
ranking_dataset	Training Test	
number_of_models	*entero*	Número de modelos que se incluirán en el nugget de modelo. Especifique un entero entre 1 y 100.
calculate_variable_importance	*tag*	
enable_accuracy_limit	*tag*	
accuracy_limit	*entero*	Entero entre 0 y 100.
enable_ area_under_curve _limit	*tag*	
area_under_curve_limit	*número*	Número real entre 0,0 y 1,0.
enable_profit_limit	*tag*	
profit_limit	*número*	Entero mayor que 0.
enable_lift_limit	*tag*	
lift_limit	*número*	Número real mayor que 1,0.
enable_number_of_variables_limit	*tag*	

Tabla 102. Propiedades de autoclassifiernode (continuación).

Propiedades de autoclassifiernode	Valores	Descripción de la propiedad
number_of_variables_limit	*número*	Entero mayor que 0.
use_fixed_cost	*tag*	
fixed_cost	*número*	Número real mayor que 0.0.
variable_cost	*campo*	
use_fixed_revenue	*tag*	
fixed_revenue	*número*	Número real mayor que 0.0.
variable_revenue	*campo*	
use_fixed_weight	*tag*	
fixed_weight	*número*	Número real mayor que 0,0
variable_weight	*campo*	
lift_percentile	*número*	Entero entre 0 y 100.
enable_model_build_time_limit	*tag*	
model_build_time_limit	*número*	Entero que indica el número máximo de minutos que se puede tardar en generar cada uno de los modelos.
enable_stop_after_time_limit	*tag*	
stop_after_time_limit	*número*	Número real que indica el número máximo de horas que puede tardar una ejecución del clasificador automático.
enable_stop_after_valid_model_produced	*tag*	
use_costs	*tag*	
<algorithm>	*tag*	Activa o desactiva el uso de un determinado algoritmo.
<algorithm>.<property>	*cadena*	Define un valor de propiedad para un algoritmo específico. Consulte el tema "Propiedades de ajustes de algoritmo" para obtener más información.

Propiedades de ajustes de algoritmo

En el caso de los nodos Clasificador automático, Autonumérico y Agrupación en clústeres automática, las propiedades de determinados algoritmos utilizados por el nodo se pueden establecer utilizando el formato general:

```
autonode.setKeyedPropertyValue(<algoritmo>, <propiedad>, <valor>)
```

Por ejemplo:

```
node.setKeyedPropertyValue("neuralnetwork", "method", "MultilayerPerceptron")
```

Los nombres de algoritmos del nodo Clasificador automático son `cart`, `chaid`, `quest`, `c50`, `logreg`, `decisionlist`, `bayesnet`, `discriminant`, `svm` y `knn`.

Los nombres de algoritmos del nodo Autonumérico son `cart`, `chaid`, `neuralnetwork`, `genlin`, `svm`, `regression`, `linear` y `knn`.

Los nombres de algoritmos del nodo Autoclúster son `twostep`, `k-means` y `kohonen`.

Los nombres de las propiedades son los nombres estándar, según se han documentado para cada nodo de algoritmo.

Las propiedades de algoritmos que contienen puntos u otros signos de puntuación deben encerrarse entre comillas simples. Por ejemplo:

```
node.setKeyedPropertyValue("logreg", "tolerance", "1.0E-5")
```

También es posible asignar varios valores a una propiedad. Por ejemplo:

```
node.setKeyedPropertyValue("decisionlist", "search_direction", ["Up", "Down"])
```

Para activar o desactivar el uso de un determinado algoritmo:

```
node.setPropertyValue("chaid", True)
```

Nota: En los casos en los que determinadas opciones de algoritmos no están disponibles en el nodo Clasificador automático o cuando sólo se puede especificar un único valor, en lugar de un intervalo de valores, se aplican los mismos límites que tienen los scripts cuando se accede al nodo de la manera estándar.

Propiedades de nodo de agrupación en clústeres automática

El nodo Agrupación en clústeres automática calcula y compara los modelos de agrupación en clústeres que identifican grupos de registros con características similares. El nodo funciona de la misma manera que otros nodos de modelado automático, permitiéndole experimentar con múltiples combinaciones de opciones en una única pasada de modelado. Los modelos se pueden comparar utilizando medidas básicas con las que se intenta filtrar y definir la utilidad de los modelos de clúster y proporcionar una medida según la importancia de campos concretos.

Ejemplo

```
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)
```

Tabla 103. Propiedades de nodo de agrupación en clústeres automática

Propiedades de autoclusternode	Valores	Descripción de la propiedad
evaluación	*campo*	**Nota:** Solo nodo Agrupación en clústeres automática. Identifica el campo cuyo valor de importancia se calculará. Como alternativa, se puede utilizar para identificar la calidad con la que el clúster diferencia el valor de este campo y, por lo tanto; la calidad con la que el modelo predecirá este campo.
ranking_measure	Silhouette Num_clusters Size_smallest_cluster Size_largest_cluster Smallest_to_largest Importance	
ranking_dataset	Entrenamiento Prueba	

Tabla 103. Propiedades de nodo de agrupación en clústeres automática (continuación)

Propiedades de autoclusternode	Valores	Descripción de la propiedad
summary_limit	*entero*	Número de modelos que se incluirán en el informe. Especifique un entero entre 1 y 100.
enable_silhouette_limit	*flag*	
silhouette_limit	*entero*	Entero entre 0 y 100.
enable_number_less_limit	*flag*	
number_less_limit	*número*	Número real entre 0,0 y 1,0.
enable_number_greater_limit	*flag*	
number_greater_limit	*número*	Entero mayor que 0.
enable_smallest_cluster_limit	*flag*	
smallest_cluster_units	Percentage Counts	
smallest_cluster_limit_percentage	*número*	
smallest_cluster_limit_count	*entero*	Entero mayor que 0.
enable_largest_cluster_limit	*flag*	
largest_cluster_units	Percentage Counts	
largest_cluster_limit_percentage	*número*	
largest_cluster_limit_count	*entero*	
enable_smallest_largest_limit	*flag*	
smallest_largest_limit	*número*	
enable_importance_limit	*flag*	
importance_limit_condition	Greater_than Less_than	
importance_limit_greater_than	*número*	Entero entre 0 y 100.
importance_limit_less_than	*número*	Entero entre 0 y 100.
<algorithm>	*flag*	Activa o desactiva el uso de un determinado algoritmo.
<algorithm>.<property>	*cadena*	Define un valor de propiedad para un algoritmo específico. Consulte el tema "Propiedades de ajustes de algoritmo" en la página 168 para obtener más información.

Propiedades de autonumericnode

El nodo Autonumérico calcula y compara modelos para resultados de rango numérico continuo utilizando cierto número de métodos diferentes. El nodo funciona de la misma manera que el nodo Clasificador automático, lo que le permite seleccionar los algoritmos que desee utilizar y experimentar con varias combinaciones de opciones en una única pasada de modelado. Los algoritmos admitidos incluyen redes neuronales, C&RT, CHAID, regresión lineal, regresión lineal generalizada y máquinas de vectores de soporte (SVM). Los modelos se pueden comparar basándose en la correlación, el error relativo o el número de variables utilizado.

Ejemplo

```
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)
```

Tabla 104. Propiedades de autonumericnode

Propiedades de autonumericnode	**Valores**	**Descripción de la propiedad**
custom_fields	*flag*	Si es verdadero, se utilizará la configuración de campos personalizada en lugar de la configuración del nodo Tipo.
target	*campo*	El nodo Autonumérico requiere un único campo objetivo y uno o más campos de entrada. También se puede especificar campos de ponderación y frecuencia. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
inputs	*[campo1 ... campo2]*	
partición	*campo*	
use_frequency	*flag*	
frequency_field	*campo*	
use_weight	*flag*	
weight_field	*campo*	
use_partitioned_data	*flag*	Si se ha definido un campo de partición, sólo se utilizarán los datos de entrenamiento para la generación del modelo.
ranking_measure	Correlation NumberOfFields	
ranking_dataset	Test Entrenamiento	
number_of_models	*entero*	Número de modelos que se incluirán en el nugget de modelo. Especifique un entero entre 1 y 100.
calculate_variable_importance	*flag*	
enable_correlation_limit	*flag*	
correlation_limit	*entero*	
enable_number_of_fields_limit	*flag*	
number_of_fields_limit	*entero*	
enable_relative_error_limit	*flag*	
relative_error_limit	*entero*	
enable_model_build_time_limit	*flag*	
model_build_time_limit	*entero*	
enable_stop_after_time_limit	*flag*	
stop_after_time_limit	*entero*	

Tabla 104. Propiedades de autonumericnode (continuación)

Propiedades de autonumericnode	Valores	Descripción de la propiedad
stop_if_valid_model	*flag*	
<algorithm>	*flag*	Activa o desactiva el uso de un determinado algoritmo.
<algorithm>.<property>	*cadena*	Define un valor de propiedad para un algoritmo específico. Consulte el tema "Propiedades de ajustes de algoritmo" en la página 168 para obtener más información.

Propiedades de bayesnetnode

El nodo Red bayesiana le permite crear un modelo de probabilidad combinando pruebas observadas y registradas con conocimiento del mundo real para establecer la probabilidad de instancias. El nodo se centra en las redes Naïve Bayes aumentado a árbol (TAN) y de manto de Markov que se utilizan principalmente para la clasificación.

Ejemplo

```
node = stream.create("bayesnet", "My node")
node.setPropertyValue("continue_training_existing_model", True)
node.setPropertyValue("structure_type", "MarkovBlanket")
node.setPropertyValue("use_feature_selection", True)
# pestaña Experto
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("independence", "Pearson")
```

Tabla 105. Propiedades de bayesnetnode

Propiedades de bayesnetnode	Valores	Descripción de la propiedad
inputs	*[campo1 ... campoN]*	Los modelos de red bayesiana utilizan un único campo objetivo y uno o más campos de entrada. Los campos continuos se establecen en intervalos automáticamente. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
continue_training_existing_model	*flag*	
structure_type	TAN MarkovBlanket	Seleccione la estructura que desea utilizar cuando cree la red bayesiana.
use_feature_selection	*flag*	
parameter_learning_method	Likelihood Bayes	Especifica el método utilizado para calcular las tablas de probabilidad condicional entre nodos donde se conocen los valores de los elementos padre.
mode	Expert Simple	
missing_values	*flag*	
all_probabilities	*flag*	

Tabla 105. Propiedades de bayesnetnode (continuación)

Propiedades de bayesnetnode	Valores	Descripción de la propiedad
independence	Likelihood Pearson	Especifica el método utilizado para determinar si las observaciones relacionadas de dos variables son independientes entre sí.
significance_level	*número*	Especifica el valor de corte para determinar la independencia.
maximal_conditioning_set	*número*	Establece el número máximo de variables de condición que se utilizarán para la comprobación de la independencia.
inputs_always_selected	*[campo1 ... campoN]*	Especifica qué campos del conjunto de datos se deben utilizar siempre al generar la red bayesiana. **Nota:** el campo objetivo siempre está seleccionado.
maximum_number_inputs	*número*	Especifica el número máximo de campos de entrada que se deben utilizar al generar la red bayesiana.
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propiedades de buildr

El nodo Crear R le permite especificar script R personalizado para realizar la creación de modelos y la puntuación de modelos desplegados en IBM SPSS Modeler.

Ejemplo

```
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)""")
```

Tabla 106. Propiedades de buildr.

Propiedades de buildr	Valores	Descripción de la propiedad
build_syntax	*cadena*	Sintaxis de scripts R para la creación del modelo.
score_syntax	*cadena*	Sintaxis de scripts R para la puntuación del modelo.
convert_flags	StringsAndDoubles LogicalValues	Opción para convertir campos de distintivos.

Tabla 106. Propiedades de buildr (continuación).

Propiedades de buildr	Valores	Descripción de la propiedad
convert_datetime	*flag*	Opción para convertir las variables con los formatos de fecha o de fecha y hora para formatos de fecha/hora R.
convert_datetime_class	POSIXct POSIXlt	Opciones para especificar a qué formato se convierten las variables con los formatos de fecha o de fecha y hora.
convert_missing	*tag*	Opción para convertir los valores que faltan al valor R NA.
output_html	*tag*	Opción para visualizar gráficos en una pestaña en el nugget de modelo R.
output_text	*flag*	Opción para escribir salida de texto de la consola R en una pestaña del modelo R.

Propiedades de c50node

El nodo C5.0 genera un árbol de decisión o un conjunto de reglas. El modelo divide la muestra basándose en el campo que ofrece la máxima ganancia de información en cada nivel. El campo objetivo debe ser categórico. Se permiten varias divisiones en más de dos subgrupos.

Ejemplo

```
node = stream.create("c50", "My node")
# "Model" tab
node.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)
# pestaña "Costes"
node.setPropertyValue("use_costs", True)
node.setPropertyValue("costs", [["drugA", "drugX", 2]])
```

Tabla 107. Propiedades de c50node

Propiedad de c50node	Valores	Descripción de la propiedad
target	*campo*	Los modelos C50 utilizan un único campo objetivo y uno o más campos de entrada. También se puede especificar un campo de ponderación. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
output_type	DecisionTree RuleSet	
group_symbolics	*flag*	
use_boost	*flag*	
boost_num_trials	*número*	

Tabla 107. Propiedades de c50node (continuación)

Propiedad de c50node	Valores	Descripción de la propiedad
use_xval	*flag*	
xval_num_folds	*número*	
mode	Simple Valores avanzados	
favor	Exactitud Generality	Generalización o precisión de favor.
expected_noise	*número*	
min_child_records	*número*	
pruning_severity	*número*	
use_costs	*flag*	
costes	*structured*	Ésta es una propiedad estructurada.
use_winnowing	*flag*	
use_global_pruning	*flag*	Activado (True) de forma predeterminada.
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propiedades de carmanode

El modelo CARMA extrae un conjunto de reglas de los datos sin necesidad de especificar campos de entrada ni de objetivo. A diferencia de Apriori el nodo CARMA ofrece configuraciones de generación basadas en el soporte de las reglas (soporte tanto para el antecedente como el consecuente) en lugar de hacerlo sólo respecto al soporte del antecedente. Esto significa que las reglas generadas se pueden utilizar en una gama de aplicaciones más amplia, por ejemplo, para buscar una lista de productos o servicios (antecedentes) cuyo consecuente es el elemento que se desea promocionar durante esta temporada de vacaciones.

Ejemplo

```
node = stream.create("carma", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Drug"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.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)
# Opciones de experto
node.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)
```

Tabla 108. Propiedades de carmanode

Propiedades de carmanode	Valores	Descripción de la propiedad
inputs	*[field1 ... fieldn]*	Los modelos CARMA utilizan una lista de campos de entrada, pero no de campos objetivo. Los campos de ponderación y frecuencia no se usan. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
id_field	*campo*	Campo utilizado como el campo de ID para la generación del modelo.
contiguous	*flag*	Se utiliza para especificar si los ID del campo de ID son contiguos.
use_transactional_data	*flag*	
content_field	*campo*	
min_supp	*número (porcentaje)*	Está relacionado con el soporte de regla en lugar de con el soporte de antecedentes. El valor por omisión es 20%.
min_conf	*número (porcentaje)*	El valor por omisión es 20%.
max_size	*número*	El valor por omisión es 10.
mode	Simple Valores avanzados	El valor predeterminado es Simple.
exclude_multiple	*flag*	Excluye las reglas con varios consecuentes. El valor predeterminado es False.
use_pruning	*flag*	El valor predeterminado es False.
pruning_value	*número*	El valor predeterminado es 500.
vary_support	*flag*	
estimated_transactions	*entero*	
rules_without_antecedents	*flag*	

Propiedades de cartnode

El nodo de árbol de clasificación y regresión (C&R) genera un árbol de decisión que permite predecir o clasificar observaciones futuras. El método utiliza la partición reiterada para dividir los registros de entrenamiento en segmentos minimizando las impurezas en cada paso, donde un nodo se considera "puro" si el 100% de los casos del nodo corresponden a una categoría específica del campo objetivo. Los campos de entrada y objetivo pueden ser continuos (rango numérico) o categóricos (nominal, ordinal o marca). Todas las divisiones son binarias (sólo se crean dos subgrupos).

Ejemplo

```
node = stream.createAt("cart", "My node", 200, 100)
# "Fields" tab
node.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")
# Pestaña "Opciones de modelo"
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Cart_Drug")
```

Tabla 109. Propiedades de cartnode

Propiedades de cartnode	**Valores**	**Descripción de la propiedad**
target	*campo*	Los modelos de árbol C&R requieren un único campo objetivo y uno o más campos de entrada. También se puede especificar un campo de frecuencia. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
continue_training_existing_model	*flag*	
objective	Standard Aumento Agregación autodocimante psm	psm se utiliza para conjuntos de datos de grandes dimensiones y requiere una conexión al Servidor.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*flag*	
tree_directives	*cadena*	Especifique directivas para desarrollar el árbol. Las directivas se pueden escribir entre comillas triples para evitar comillas o saltos de línea no deseados. Recuerde que las directivas pueden ser muy sensibles a las pequeñas modificaciones de las opciones de modelado o los datos y es posible que no se puedan generalizar para otros conjuntos de datos.
use_max_depth	Predeterminado Personalizado	
max_depth	*entero*	Máxima profundidad del árbol, desde 0 a 1000. Sólo se utiliza si use_max_depth = Custom.
prune_tree	*flag*	Poda del árbol para evitar sobreajustes.
use_std_err	*flag*	Use la diferencia máxima en riesgos (en errores estándar).

Tabla 109. Propiedades de cartnode (continuación)

Propiedades de cartnode	Valores	Descripción de la propiedad
std_err_multiplier	*número*	Diferencia máxima.
max_surrogates	*número*	Número máximo de sustitutos.
use_percentage	*flag*	
min_parent_records_pc	*número*	
min_child_records_pc	*número*	
min_parent_records_abs	*número*	
min_child_records_abs	*número*	
use_costs	*flag*	
costes	*structured*	Propiedad estructurada.
priors	Datos Equal Personalizado	
custom_priors	*structured*	Propiedad estructurada.
adjust_priors	*flag*	
trails	*número*	Número de modelos de componente para un aumento o agregación autodocimante.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Regla de combinación predeterminada para objetivos categóricos.
range_ensemble_method	Media Mediana	Regla de combinación predeterminada para objetivos continuos.
large_boost	*flag*	Aplicar aumento a conjunto de datos muy grandes.
min_impurity	*número*	
impurity_measure	Gini Twoing Ordered	
train_pct	*número*	Conjunto de prevención sobreajustado.
set_random_seed	*flag*	Opción replicar resultados.
seed	*número*	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propiedades de chaidnode

El nodo CHAID genera árboles de decisión utilizando estadísticos de chi-cuadrado para identificar las divisiones óptimas. A diferencia de los nodos C&RT y Árbol y QUEST, CHAID puede generar árboles no binarios, lo que significa que algunas divisiones generarán más de dos ramas. Los campos de entrada y objetivo pueden ser continuos (rango numérico) o categóricos. CHAID exhaustivo es una modificación de CHAID que examina con mayor precisión todas las divisiones posibles, aunque necesita más tiempo para realizar los cálculos.

Ejemplo

```
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)
```

Tabla 110. Propiedades de chaidnode

Propiedad de chaidnode	Valores	Descripción de la propiedad
target	*campo*	Los modelos CHAID requieren un único campo objetivo y uno o más campos de entrada. También se puede especificar un campo de frecuencia. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
continue_training_existing_model	*flag*	
objective	Standard Aumento Agregación autodocimante psm	psm se utiliza para conjuntos de datos de grandes dimensiones y requiere una conexión al Servidor.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*flag*	
tree_directives	*cadena*	
method	Chaid ExhaustiveChaid	
use_max_depth	Predeterminado Personalizado	
max_depth	*entero*	Máxima profundidad del árbol, desde 0 a 1000. Sólo se utiliza si use_max_depth = Custom.
use_percentage	*flag*	
min_parent_records_pc	*número*	
min_child_records_pc	*número*	

Tabla 110. Propiedades de chaidnode (continuación)

Propiedad de chaidnode	Valores	Descripción de la propiedad
min_parent_records_abs	*número*	
min_child_records_abs	*número*	
use_costs	*flag*	
costes	*structured*	Propiedad estructurada.
trails	*número*	Número de modelos de componente para un aumento o agregación autodocimante.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Regla de combinación predeterminada para objetivos categóricos.
range_ensemble_method	Media Mediana	Regla de combinación predeterminada para objetivos continuos.
large_boost	*flag*	Aplicar aumento a conjunto de datos muy grandes.
split_alpha	*número*	Nivel de significancia para división.
merge_alpha	*número*	Nivel de significancia para fusión.
bonferroni_adjustment	*flag*	Los valores de significancia de ajuste utilizando el método de Bonferroni.
split_merged_categories	*flag*	Permitir segunda división de categorías fusionadas.
chi_square	Pearson LR	Método usado para calcular la estadística de chi cuadrado: Pearson o Razón de verosimilitud
epsilon	*número*	Cambio mínimo en frecuencias de casillas esperadas.
max_iterations	*número*	Número máximo de iteraciones para la convergencia.
set_random_seed	*entero*	
seed	*número*	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	
maximum_number_of_models	*entero*	

Propiedades de coxregnode

El nodo Regresión de Cox le permite crear un modelo de supervivencia para datos de tiempo hasta el evento en presencia de registros censurados. El modelo produce una función de supervivencia que predice la probabilidad de que el evento de interés se haya producido en el momento dado (*t*) para valores determinados de las variables de entrada.

Ejemplo

```
node = stream.create("coxreg", "My node")
node.setPropertyValue("survival_time", "tenure")
node.setPropertyValue("method", "BackwardsStepwise")
# pestaña Experto
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("removal_criterion", "Conditional")
node.setPropertyValue("survival", True)
```

Tabla 111. Propiedades de coxregnode

Propiedades de coxregnode	**Valores**	**Descripción de la propiedad**
survival_time	*campo*	Los modelos de regresión de Cox requieren un único campo con los tiempos de supervivencia.
target	*campo*	Los modelos de regresión de Cox requieren un único campo objetivo y uno o más campos de entrada. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
method	Intro Stepwise BackwardsStepwise	
grupos	*campo*	
model_type	MainEffects Personalizado	
custom_terms	[*"BP*Sexo" "BP*Edad"*]	
mode	Expert Simple	
max_iterations	*número*	
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	*número*	
probability_removal	*número*	

Tabla 111. Propiedades de coxregnode (continuación)

Propiedades de coxregnode	Valores	Descripción de la propiedad
output_display	EachStep LastStep	
ci_enable	*flag*	
ci_value	90 95 99	
correlación	*flag*	
display_baseline	*flag*	
survival	*flag*	
hazard	*flag*	
log_minus_log	*flag*	
one_minus_survival	*flag*	
separate_line	*campo*	
value	*número* o *cadena*	Si no se especifica ningún valor para un campo, se utilizará la opción predeterminada "Mean" para dicho campo.

Propiedades de decisionlistnode

El nodo Lista de decisiones identifica subgrupos, o segmentos, que muestran una mayor o menor posibilidad de proporcionar un resultado binario relacionado con la población global. Por ejemplo, puede buscar clientes que tengan menos posibilidades de abandonar o más posibilidades de responder favorablemente a una campaña. Puede incorporar su conocimiento empresarial al modelo añadiendo sus propios segmentos personalizados y previsualizando modelos alternativos uno junto a otro para comparar los resultados. Los modelos de listas de decisiones constan de una lista de reglas en las que cada regla tiene una condición y un resultado. Las reglas se aplican en orden, y la primera regla que coincide determina el resultado.

Ejemplo

```
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)
```

Tabla 112. Propiedades de decisionlistnode

Propiedades de decisionlistnode	Valores	Descripción de la propiedad
target	*campo*	Los modelos de listas de decisiones utilizan un único campo objetivo y uno o más campos de entrada. También se puede especificar un campo de frecuencia. Consulte el tema “Propiedades comunes de nodos de modelado” en la página 161 para obtener más información.
model_output_type	Modelo InteractiveBuilder	

Tabla 112. Propiedades de decisionlistnode (continuación)

Propiedades de decisionlistnode	Valores	Descripción de la propiedad
search_direction	Up Bajar	Hace referencia a la localización de segmentos, donde Up es el equivalente a Alta probabilidad y Down es el equivalente a Baja probabilidad.
target_value	*cadena*	Si no se especifica, se supondrá el valor true para las marcas.
max_rules	*entero*	Número máximo de segmentos sin incluir el resto.
min_group_size	*entero*	Tamaño mínimo del segmento.
min_group_size_pct	*número*	Tamaño mínimo del segmento como porcentaje.
confidence_level	*número*	Umbral mínimo que un campo de entrada tiene que mejorar la probabilidad de la respuesta (aumentar la elevación) para que merezca la pena añadirlo a la definición de un segmento.
max_segments_per_rule	*entero*	
mode	Simple Valores avanzados	
bin_method	EqualWidth EqualCount	
bin_count	*número*	
max_models_per_cycle	*entero*	Amplitud de búsqueda de las listas.
max_rules_per_cycle	*entero*	Amplitud de búsqueda de las reglas de segmentación.
segment_growth	*número*	
include_missing	*flag*	
final_results_only	*flag*	
reuse_fields	*flag*	Permite la reutilización de los atributos (los campos de entrada que aparecen en las reglas).
max_alternatives	*entero*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propiedades de discriminantnode

El análisis discriminante realiza más supuestos rigurosos que regresiones logísticas pero puede ser una alternativa o un suplemento valioso al análisis de regresión logística si se cumplen dichos supuestos.

Ejemplo

```
node = stream.create("discriminant", "My node")
node.setPropertyValue("target", "custcat")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "Stepwise")
```

Tabla 113. Propiedades de discriminantnode

Propiedades de discriminantnode	**Valores**	**Descripción de la propiedad**
target	*campo*	Los modelos discriminantes requieren un único campo objetivo y uno o más campos de entrada. Los campos de ponderación y frecuencia no se usan. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
method	Intro Por pasos	
mode	Simple Valores avanzados	
prior_probabilities	AllEqual ComputeFromSizes	
covariance_matrix	WithinGroups SeparateGroups	
medias	*flag*	Opciones de estadísticos del cuadro de diálogo Salida avanzada.
univariate_anovas	*flag*	
box_m	*flag*	
within_group_covariance	*flag*	
within_groups_correlation	*flag*	
separate_groups_covariance	*flag*	
total_covariance	*flag*	
fishers	*flag*	
unstandardized	*flag*	
casewise_results	*flag*	Opciones de clasificación del cuadro de diálogo Salida avanzada.
limit_to_first	*número*	El valor predeterminado es 10.
summary_table	*flag*	
leave_one_classification	*flag*	
combined_groups	*flag*	
separate_groups_covariance	*flag*	Opción de matrices **Covarianza de grupos separados**.
territorial_map	*flag*	
combined_groups	*flag*	Opción de gráfico **Grupos combinados**.
separate_groups	*flag*	Opción de gráfico **Grupos separados**.
summary_of_steps	*flag*	
F_pairwise	*flag*	

Tabla 113. Propiedades de discriminantnode (continuación)

Propiedades de discriminantnode	Valores	Descripción de la propiedad
stepwise_method	WilksLambda UnexplainedVariance MahalanobisDistance SmallestF RaosV	
V_to_enter	*número*	
criterios	UseValue UseProbability	
F_value_entry	*número*	El valor predeterminado es 3.84.
F_value_removal	*número*	El valor predeterminado es 2.71.
probability_entry	*número*	El valor predeterminado es 0.05.
probability_removal	*número*	El valor predeterminado es 0,10.
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propiedades de factornode

El nodo PCA/Factorial proporciona técnicas eficaces de reducción de datos para reducir la complejidad de los datos. Análisis de componentes principales (PCA) busca combinaciones lineales de los campos de entrada que realizan el mejor trabajo a la hora de capturar la varianza en todo el conjunto de campos, en el que los componentes son ortogonales (perpendiculares) entre ellos. Análisis factorial intenta identificar factores subyacentes que expliquen el patrón de correlaciones dentro de un conjunto de campos observados. Para los dos métodos, el objetivo es encontrar un número pequeño de campos derivados que resuma de forma eficaz la información del conjunto original de campos.

Ejemplo

```
node = stream.create("factor", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["BP", "Na", "K"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Factor_Age")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "GLS")
# Opciones de experto
node.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)
# Sección "Rotación"
node.setPropertyValue("rotation", "DirectOblimin")
node.setPropertyValue("delta", 0.3)
node.setPropertyValue("kappa", 7.0)
```

Tabla 114. Propiedades de factornode

Propiedad de factornode	Valores	Descripción de la propiedad
inputs	[*field1 ... fieldN*]	Los modelos PCA/Factorial utilizan una lista de campos de entrada, pero no de campos objetivo. Los campos de ponderación y frecuencia no se usan. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
method	PC ULS GLS ML PAF Alpha Image	
mode	Simple Valores avanzados	
max_iterations	*número*	
complete_records	*flag*	
matrix	Correlation Covariance	
extract_factors	ByEigenvalues ByFactors	
min_eigenvalue	*número*	
max_factor	*número*	
rotación	Ninguno Varimax DirectOblimin Equamax Quartimax Promax	
delta	*número*	Si selecciona DirectOblimin como el tipo de datos de rotación, podrá especificar un valor para delta. Si no especifica ningún valor, se utilizará el valor predeterminado para delta.
kappa	*número*	Si selecciona Promax como el tipo de datos de rotación, podrá especificar un valor para kappa. Si no especifica ningún valor, se utilizará el valor predeterminado para kappa.
sort_values	*flag*	
hide_values	*flag*	
hide_below	*número*	

Propiedades de featureselectionnode

El nodo Selección de características filtra los campos de entrada para su eliminación en función de un conjunto de criterios (como el porcentaje de valores perdidos); a continuación, clasifica el grado de importancia del resto de entradas de acuerdo con un objetivo específico. Por ejemplo, a partir de un conjunto de datos dado con cientos de entradas potenciales, ¿cuáles tienen mayor probabilidad de ser útiles para el modelado de resultados de pacientes?

Ejemplo

```
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)
```

Para obtener un ejemplo más detallado que cree y aplique un modelo de Selección de características, consulte en.

Tabla 115. Propiedades de featureselectionnode

Propiedad de **featureselectionnode**	Valores	Descripción de la propiedad
target	*campo*	Los modelos de selección de características ordenan predictores por rangos con respecto a su objetivo específico. Los campos de ponderación y frecuencia no se usan. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
screen_single_category	*flag*	En caso de establecer True, filtra campos que tienen demasiados registros dentro de la misma categoría respecto al número total de registros.
max_single_category	*número*	Especifica el umbral que se utiliza cuando screen_single_category es True.
screen_missing_values	*flag*	Si se establece en True, filtra campos con demasiados valores perdidos, expresados como un porcentaje del número total de registros.
max_missing_values	*número*	
screen_num_categories	*flag*	Si se establece en True, filtra campos con demasiadas categorías respecto al número total de registros.
max_num_categories	*número*	
screen_std_dev	*flag*	Si se establece en True, filtra campos con una desviación estándar menor o igual que el mínimo especificado.
min_std_dev	*número*	

Tabla 115. Propiedades de featureselectionnode (continuación)

Propiedad de **featureselectionnode**	Valores	Descripción de la propiedad
screen_coeff_of_var	*flag*	Si se establece en True, filtra campos con un coeficiente de varianza menor o igual que el mínimo especificado.
min_coeff_of_var	*número*	
criterios	Pearson Likelihood CramersV Lambda	Al clasificar los predictores categóricos en función de un objetivo categórico, especifica la medida en la que se basa el valor de importancia.
unimportant_below	*número*	Especifica los valores *p* de umbral utilizados para clasificar las variables como importantes, marginales o sin importancia. Acepta valores de 0,0 a 1,0.
important_above	*número*	Acepta valores de 0,0 a 1,0.
unimportant_label	*cadena*	Especifica la etiqueta para la clasificación como 'Sin importancia'.
marginal_label	*cadena*	
important_label	*cadena*	
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*flag*	Si selection_mode se establece en ImportanceLevel, determina si se seleccionan los campos importantes.
select_marginal	*flag*	Si selection_mode se establece en ImportanceLevel, determina si se seleccionan los campos marginales.
select_unimportant	*flag*	Si selection_mode se establece en ImportanceLevel, determina si se seleccionan los campos sin importancia.
importance_value	*número*	Si selection_mode se establece en ImportanceValue, determina el valor de corte que se va a usar. Acepta valores de 0 a 100.
top_n	*entero*	Si selection_mode se establece en TopN, determina el valor de corte que se va a usar. Acepta valores de 0 a 1000.

Propiedades de genlinnode

El modelo lineal generalizado amplía el modelo lineal general, de manera que la variable dependiente está relacionada linealmente con los factores y las covariables mediante una determinada función de enlace. Además, el modelo permite que la variable dependiente tenga una distribución no normal. Cubre la funcionalidad de un amplio número de modelo estadísticos, incluyendo regresión lineal, regresión logística, modelos log lineales para recuento de datos y modelos de supervivencia censurados por intervalos.

Ejemplo

```
node = stream.create("genlin", "My node")
node.setPropertyValue("model_type", "MainAndAllTwoWayEffects")
node.setPropertyValue("offset_type", "Variable")
node.setPropertyValue("offset_field", "Claimant")
```

Tabla 116. Propiedades de genlinnode

Propiedades de genlinnode	Valores	Descripción de la propiedad
target	*campo*	Los modelos lineales generalizados requieren un único campo objetivo, que debe ser un campo nominal o marca, y uno o más campos de entrada. También se puede especificar un campo de ponderación. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
use_weight	*flag*	
weight_field	*campo*	El tipo de campo es únicamente continuo.
target_represents_trials	*flag*	
trials_type	Variable FixedValue	
trials_field	*campo*	El tipo de campo es continuo, marca u ordinal.
trials_number	*número*	El valor predeterminado es 10.
model_type	MainEffects MainAndAllTwoWayEffects	
offset_type	Variable FixedValue	
offset_field	*campo*	El tipo de campo es únicamente continuo.
offset_value	*número*	Debe ser un número real.
base_category	Last Primero	
include_intercept	*flag*	
mode	Simple Valores avanzados	
distribution	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS: De Gauss inversa. NEGBIN: Binomial negativa.
negbin_para_type	Especifica Estimate	
negbin_parameter	*número*	El valor predeterminado es 1. Debe contener un número real no negativo.
tweedie_parameter	*número*	

Tabla 116. Propiedades de genlinnode (continuación)

Propiedades de genlinnode	Valores	Descripción de la propiedad
link_function	IDENTITY CLOGLOG LOG LOGC LOGIT NEGBIN NLOGLOG ODDSPOWER PROBIT POWER CUMCAUCHIT CUMCLOGLOG CUMLOGIT CUMNLOGLOG CUMPROBIT	CLOGLOG: log-log complementario. LOGC: complemento log. NEGBIN: Negative binomial. NLOGLOG: Log-log negativo. CUMCAUCHIT: Cauchit acumulada. CUMCLOGLOG: Log-log complementario acumulado. CUMLOGIT: Logit acumulado. CUMNLOGLOG: Log-log negativo acumulado. CUMPROBIT: Probit acumulado.
potencia	*número*	El valor debe ser real y distinto de cero.
method	Hybrid Fisher NewtonRaphson	
max_fisher_iterations	*número*	El valor predeterminado es 1; sólo se admiten enteros positivos.
scale_method	MaxLikelihoodEstimate Deviance PearsonChiSquare FixedValue	
scale_value	*número*	El valor predeterminado es 1; debe ser mayor que 0.
covariance_matrix	ModelEstimator RobustEstimator	
max_iterations	*número*	El valor predeterminado es 100; sólo enteros no negativos.
max_step_halving	*número*	El valor predeterminado es 5; sólo enteros positivos.
check_separation	*flag*	
start_iteration	*número*	El valor predeterminado es 20; sólo se admiten enteros positivos.
estimates_change	*flag*	
estimates_change_min	*número*	El valor predeterminado es 1E-006; sólo se admiten números positivos.
estimates_change_type	Absolute Relative	
loglikelihood_change	*flag*	
loglikelihood_change_min	*número*	Sólo se admiten números positivos.
loglikelihood_change_type	Absolute Relative	
hessian_convergence	*flag*	
hessian_convergence_min	*número*	Sólo se admiten números positivos.
hessian_convergence_type	Absolute Relative	

Tabla 116. Propiedades de genlinnode (continuación)

Propiedades de genlinnode	Valores	Descripción de la propiedad
case_summary	*flag*	
contrast_matrices	*flag*	
descriptive_statistics	*flag*	
estimable_functions	*flag*	
model_info	*flag*	
iteration_history	*flag*	
goodness_of_fit	*flag*	
print_interval	*número*	El valor predeterminado es 1; debe ser un entero positivo.
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	
citype	Wald Profile	
tolerancelevel	*número*	El valor predeterminado es 0.0001.
confidence_interval	*número*	El valor predeterminado es 95.
loglikelihood_function	Completa Kernel	
singularity_tolerance	1E-007 1E-008 1E-009 1E-010 1E-011 1E-012	
value_order	Ascending Descending DataOrder	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propiedades de glmmnode

Un modelo lineal mixto generalizado (GLMM) amplía el modelo lineal de modo que el objetivo pueda tener una distribución no normal, esté linealmente relacionado con los factores y covariables mediante una función de enlace especificada y las observaciones se puedan correlacionar. Los modelos lineales mixtos generalizados cubren una amplia variedad de modelos, desde modelos de regresión lineal simple hasta modelos multinivel complejos para datos longitudinales no normales.

Tabla 117. Propiedades de glmmnode.

Propiedades de glmmnode	Valores	Descripción de la propiedad
residual_subject_spec	*estructurado*	La combinación de valores de los campos categóricos especificados que definen de forma exclusiva los sujetos del conjunto de datos.
repeated_measures	*estructurado*	Campos utilizados para identificar observaciones repetidas.
residual_group_spec	[*field1 ... fieldN*]	Campos que definen conjuntos independientes de parámetros de covarianza de efectos repetidos.
residual_covariance_type	Diagonal AR1 ARMA11 COMPOUND_SYMMETRY IDENTITY TOEPLITZ UNSTRUCTURED VARIANCE_COMPONENTS	Especifica la estructura de covarianza de residuos.
custom_target	*flag*	Indica si se puede utilizar un objetivo definido en el nodo anterior (false) o un objetivo personalizado especificado por target_field (true).
target_field	*campo*	Campo a utilizar como objetivo si custom_target es true.
use_trials	*flag*	Indica si hay que utilizar el campo adicional o el valor que especifica el número de ensayos cuando la respuesta objetivo es un número de eventos que tienen lugar en un conjunto de ensayos. El valor predeterminado es false.
use_field_or_value	Campo Value	Indica si se utiliza el campo (valor predeterminado) o valor para especificar el número de ensayos.
trials_field	*campo*	Campo a utilizar para especificar el número de ensayos.
trials_value	*entero*	Valor a utilizar para especificar el número de ensayos. Si se especifica, el valor mínimo es 1.
use_custom_target_reference	*flag*	Indica si hay que utilizar la categoría de referencia personalizada para un objetivo categórico. El valor predeterminado es false.

Tabla 117. Propiedades de glmmnode (continuación).

Propiedades de glmmnode	Valores	Descripción de la propiedad
target_reference_value	*cadena*	Categoría de referencia a utilizar si use_custom_target_reference es true.
dist_link_combination	Nominal Logit GammaLog BinomialLogit PoissonLog BinomialProbit NegbinLog BinomialLogC Custom	Modelos comunes para la distribución de valores de objetivo. Seleccione Custom para especificar una distribución de la lista proporcionada por target_distribution.
target_distribution	Normal Binomial Multinomial Gamma Inverso NegativeBinomial Poisson	La distribución de valores de objetivo cuando dist_link_combination es Custom.
link_function_type	Identity LogC Log CLOGLOG Logit NLOGLOG PROBIT POWER CAUCHIT	Función de enlace para relacionar valores de objetivo a los predictores. Si target_distribution es Binomial podrá utilizarse cualquiera de las funciones de enlace listadas. Si target_distribution es Multinomial podrán utilizarse CLOGLOG, CAUCHIT, LOGIT, NLOGLOG o PROBIT. Si target_distribution es cualquier cosa distinta de Binomial o Multinomial podrán utilizarse IDENTITY, LOG o POWER.
link_function_param	*número*	Valor del parámetro de función de enlace que hay que utilizar. Sólo es aplicable si normal_link_function o link_function_type es POWER.
use_predefined_inputs	*flag*	Indica si los campos de efectos fijos deben ser aquellos definidos anteriormente como campos de entrada (true) o han de ser los campos fixed_effects_list (false). El valor predeterminado es false.
fixed_effects_list	*estructurado*	Si use_predefined_inputs es falso, especifica los campos de entrada que se han de utilizar como campos de efectos fijos.
use_intercept	*flag*	Si es true, el valor predeterminado. incluye la interceptación en el modelo.
random_effects_list	*estructurado*	Lista de campos para especificar como efectos aleatorios.
regression_weight_field	*campo*	Campo a utilizar como campo de ponderación de análisis.
use_offset	Ninguno offset_value offset_field	Indica cómo se especifica la compensación. El valor None significa que no se ha utilizado compensación.

Tabla 117. Propiedades de glmmnode (continuación).

Propiedades de glmmnode	Valores	Descripción de la propiedad
offset_value	*número*	El valor que se ha de utilizar para desplazamiento si use_offset se establece en offset_value.
offset_field	*campo*	El valor que se ha de utilizar para desplazamiento si use_offset se establece en offset_field.
target_category_order	Ascending Descending Datos	Orden de clasificación para objetivos categóricos. El valor Data especifica que se utiliza el orden de clasificación de los datos. El valor predeterminado es Ascending.
inputs_category_order	Ascending Descending Datos	Orden de clasificación para predictores categóricos. El valor Data especifica que se utiliza el orden de clasificación de los datos. El valor predeterminado es Ascending.
max_iterations	*entero*	Número máximo de iteraciones que ejecutará el algoritmo. Un número entero no negativo; el valor predeterminado es 100.
confidence_level	*entero*	Nivel de confianza utilizado para calcular estimaciones de intervalo de los coeficientes del modelo. Un número entero no negativo; el valor máximo es 100, el valor predeterminado es 95.
degrees_of_freedom_method	Fixed Varied	Especifica cómo se calculan los grados de libertad para la prueba de significación.
test_fixed_effects_coeffecients	Modelo Robust	Método para calcular la matriz de covarianza de las estimaciones de los parámetros.
use_p_converge	*marca*	Opción para la convergencia de parámetros.
p_converge	*número*	Blanco, o cualquier valor positivo.
p_converge_type	Absolute Relative	
use_l_converge	*marca*	Opción para la convergencia log-likelihood.
l_converge	*número*	Blanco, o cualquier valor positivo.
l_converge_type	Absolute Relative	
use_h_converge	*marca*	Opción para la convergencia hessiana.
h_converge	*número*	Blanco, o cualquier valor positivo.
h_converge_type	Absolute Relative	
max_fisher_steps	*entero*	
singularity_tolerance	*número*	
use_model_name	*flag*	Indica si hay que especificar un nombre personalizado para el modelo (true) o si se ha de utilizar el nombre generado por el sistema false). El valor predeterminado es false.

Tabla 117. Propiedades de glmmnode (continuación).

Propiedades de glmmnode	Valores	Descripción de la propiedad
model_name	*cadena*	Si use_model_name es true, especifica el nombre de modelo que se va a utilizar.
confidence	onProbability onIncrease	Base para calcular el valor de confianza de la puntuación: probabilidad más alta predicha, o la diferencia entre la probabilidad más alta predicha y la segunda probabilidad más alta.
score_category_probabilities	*flag*	Si es true, genera las probabilidades predichas para objetivos categóricos. El valor predeterminado es false.
max_categories	*entero*	Si score_category_probabilities es true, especifica el número máximo de categorías que se han de guardar.
score_propensity	*flag*	Si es true, produce puntuaciones de propensión para campos de objetivo de marca que indican la probabilidad del resultado "true" para el campo.
emeans	*estructura*	Para cada campo categórico de la lista de efectos fijos, especifica si hay que producir medias marginales estimadas.
covariance_list	*estructura*	Para cada campo continuo de la lista de efectos fijos, especifica si hay que usar la media o un valor personalizado al calcular medias marginales estimadas.
mean_scale	Original Transformed	Especifica si las medias marginales estimadas se calculan basándose en la escala original del objetivo (valor predeterminado) o en la transformación de la función de enlace.
comparison_adjustment_method	DMS SEQBONFERRONI SEQSIDAK	Método de ajuste que hay que utilizar al realizar pruebas de hipótesis con varios contrastes.

Propiedades de kmeansnode

El nodo K-medias agrupa conjuntos de datos en grupos distintos (o clústeres). El método define un número fijo de clústeres, de forma iterativa asigna registros a los clústeres y ajusta los centros de los clústeres hasta que no se pueda mejorar el modelo. En lugar de intentar predecir un resultado, los modelos de *k*-medias utilizan un proceso conocido como aprendizaje no supervisado para revelar los patrones del conjunto de campos de entrada.

Ejemplo

```
node = stream.create("kmeans", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Cholesterol", "BP", "Drug", "Na", "K", "Age"])
# "Model" tab
node.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" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("stop_on", "Custom")
node.setPropertyValue("max_iterations", 10)
node.setPropertyValue("tolerance", 3.0)
node.setPropertyValue("encoding_value", 0.3)
```

Tabla 118. Propiedades de kmeansnode

Propiedad de kmeansnode	Valores	Descripción de la propiedad
inputs	[*field1 ... fieldN*]	Los modelos de K-Medias realizan un análisis de clústeres en un conjunto de campos de entrada pero no utilizan ningún campo objetivo. Los campos de ponderación y frecuencia no se usan. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
num_clusters	*número*	
gen_distance	*flag*	
cluster_label	Cadena Number	
label_prefix	*cadena*	
mode	Simple Valores avanzados	
stop_on	Predeterminado Personalizado	
max_iterations	*número*	
tolerance	*número*	
encoding_value	*número*	
optimize	Speed Memory	Se utiliza para especificar si la generación del modelo se debe optimizar para la velocidad o la memoria.

Propiedades de knnnode

El nodo *k* de modelado de vecino (KNN) asocia el nuevo caso con la categoría o valor de los objetos *k* junto a él en el espacio de predictores, donde *k* es un entero. Los casos parecidos están próximos y los que no lo son están alejados entre sí.

Ejemplo

```
node = stream.create("knn", "My node")
# Objectives tab
node.setPropertyValue("objective", "Custom")
# Settings tab - Neighbors panel
node.setPropertyValue("automatic_k_selection", False)
```

```
node.setPropertyValue("fixed_k", 2)
node.setPropertyValue("weight_by_importance", True)
# Settings tab - Analyze panel
node.setPropertyValue("save_distances", True)
```

Tabla 119. Propiedades de knnnode

Propiedades de knnnode	Valores	Descripción de la propiedad
de errores	PredictTarget IdentifyNeighbors	
objective	Balance Speed Exactitud Personalizado	
normalize_ranges	*flag*	
use_case_labels	*flag*	Seleccione esta casilla de verificación para activar la siguiente opción.
case_labels_field	*campo*	
identify_focal_cases	*flag*	Seleccione esta casilla de verificación para activar la siguiente opción.
focal_cases_field	*campo*	
automatic_k_selection	*flag*	
fixed_k	*entero*	Se activa únicamente si el valor de automatic_k_selection es False.
minimum_k	*entero*	Se activa únicamente si el valor de automatic_k_selection es True.
maximum_k	*entero*	
distance_computation	Euclidean CityBlock	
weight_by_importance	*flag*	
range_predictions	Media Mediana	
perform_feature_selection	*flag*	
forced_entry_inputs	[*field1 ... fieldN*]	
stop_on_error_ratio	*flag*	
number_to_select	*entero*	
minimum_change	*número*	
validation_fold_assign_by_field	*flag*	
number_of_folds	*entero*	Sólo se activa si el valor de validation_fold_assign_by_field es False
set_random_seed	*flag*	
random_seed	*número*	
folds_field	*campo*	Sólo se activa si el valor de validation_fold_assign_by_field es True
all_probabilities	*flag*	
save_distances	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

Tabla 119. Propiedades de knnnode (continuación)

Propiedades de knnnode	Valores	Descripción de la propiedad
adjusted_propensity_partition	Test Validation	

Propiedades de kohonennode

El nodo Kohonen genera un tipo de red neuronal que se puede usar para agrupar un conjunto de datos en grupos distintos. Cuando la red se termina de entrenar, los registros que son similares se deberían cerrar juntos en el mapa de resultados, mientras que los registros que son diferentes aparecerían aparte. Puede observar el número de observaciones capturadas por cada unidad en el nugget de modelo para identificar unidades fuertes. Esto le proporcionará una idea del número apropiado de clústeres.

Ejemplo

```
node = stream.create("kohonen", "My node")
# "Model" tab
node.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" tab
node.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)
```

Tabla 120. Propiedades de kohonennode

Propiedad de kohonennode	Valores	Descripción de la propiedad
inputs	[*field1 ... fieldN*]	Los modelos Kohonen utilizan una lista de campos de entrada, pero no de campos objetivo. Los campos de frecuencia y ponderación no se usan. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
continue	*flag*	
show_feedback	*flag*	
stop_on	Predeterminado Hora	
time	*número*	
optimize	Speed Memory	Se utiliza para especificar si la generación del modelo se debe optimizar para la velocidad o la memoria.

Tabla 120. Propiedades de kohonennode (continuación)

Propiedad de kohonennode	Valores	Descripción de la propiedad
cluster_label	*flag*	
mode	Simple Valores avanzados	
width	*número*	
longitud	*número*	
decay_style	Lineal Exponential	
phase1_neighborhood	*número*	
phase1_eta	*número*	
phase1_cycles	*número*	
phase2_neighborhood	*número*	
phase2_eta	*número*	
phase2_cycles	*número*	

Propiedades de linearnode

Los modelos de regresión lineal predicen un objetivo continuo tomando como base las relaciones lineales entre el destino y uno o más predictores.

Ejemplo

```
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")
```

Tabla 121. Propiedades de linearnode.

Propiedades de linearnode	Valores	Descripción de la propiedad
objetivo	*campo*	Especifica un campo de objetivo único.
inputs	[*field1 ... fieldN*]	Campos de predictor utilizados por el modelo.
continue_training_existing_model	*tag*	
objective	Standard Agregación autodocimante Aumento psm	psm se utiliza para conjuntos de datos de grandes dimensiones y requiere una conexión al Servidor.
use_auto_data_preparation	*tag*	
confidence_level	*number*	

Tabla 121. Propiedades de linearnode (continuación).

Propiedades de linearnode	Valores	Descripción de la propiedad
model_selection	ForwardStepwise BestSubsets Ninguno	
criteria_forward_stepwise	AICC Estadísticas F R cuadrado corregido ASE	
probability_entry	*number*	
probability_removal	*number*	
use_max_effects	*tag*	
max_effects	*number*	
use_max_steps	*tag*	
max_steps	*number*	
criteria_best_subsets	AICC R cuadrado corregido ASE	
combining_rule_continuous	Media Mediana	
component_models_n	*number*	
use_random_seed	*tag*	
random_seed	*number*	
use_custom_model_name	*tag*	
custom_model_name	*cadena*	
use_custom_name	*tag*	
custom_name	*cadena*	
tooltip	*cadena*	
palabras clave	*cadena*	
annotation	*cadena*	

Propiedades de linearasnode

Los modelos de regresión lineal predicen un objetivo continuo tomando como base las relaciones lineales entre el destino y uno o más predictores.

Tabla 122. Propiedades de linearasnode

Propiedades de linearasnode	Valores	Descripción de la propiedad
objetivo	*campo*	Especifica un campo de objetivo único.
inputs	[*field1 ... fieldN*]	Campos de predictor utilizados por el modelo.
weight_field	*campo*	Campo de análisis usado por el modelo.

Tabla 122. Propiedades de linearasnode (continuación)

Propiedades de linearasnode	Valores	Descripción de la propiedad
custom_fields	*distintivo*	El valor predeterminado es TRUE.
intercept	*distintivo*	El valor predeterminado es TRUE.
detect_2way_interaction	*distintivo*	Si se considera o no una interacción de dos sentidos. El valor predeterminado es TRUE.
cin	*número*	Intervalo de confianza usado para calcular las estimaciones de los coeficientes del modelo. Especifique un valor mayor que 0 y menor que 100. El valor predeterminado es 95.
factor_order	ascending descending	Orden de clasificación de los predictores categóricos. El valor predeterminado es ascending.
var_select_method	ForwardStepwise BestSubsets none	El método de selección de modelo que se va a usar. El valor predeterminado es ForwardStepwise.
criteria_for_forward_stepwise	AICC Estadísticas F R cuadrado corregido ASE	Esta estadística se usa para determinar si debe añadirse o eliminarse un efecto del modelo. El valor predeterminado es AdjustedRSquare.
pin	*número*	Se añade al modelo el efecto que tiene el menor valor p que es menor que este umbral pin especificado. El valor predeterminado es 0,05.
pout	*número*	Se eliminan los efectos del modelo que tienen un valor p mayor que este umbral pout especificado. El valor predeterminado es 0,10.
use_custom_max_effects	*distintivo*	Si se usa un número máximo de efectos en el modelo final. El valor predeterminado es FALSE.
max_effects	*número*	Número máximo de efectos por usar en el modelo final. El valor predeterminado es 1.
use_custom_max_steps	*distintivo*	Si se usa el número máximo de pasos. El valor predeterminado es FALSE.
max_steps	*número*	El número máximo de pasos antes de que se detenga el algoritmo escalonado. El valor predeterminado es 1.
criteria_for_best_subsets	AICC R cuadrado corregido ASE	El modo de criterios por usar. El valor predeterminado es AdjustedRSquare.

Propiedades de logregnode

La regresión logística es una técnica de estadístico para clasificar los registros en función los valores de los campos de entrada. Es análoga a la regresión lineal pero toma un campo objetivo categórico en lugar de uno numérico.

Ejemplo multinomial

```
node = stream.create("logreg", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.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" tab
node.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" options
node.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")
```

Ejemplo binomial

```
node = stream.create("logreg", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Cholesterol")
node.setPropertyValue("inputs", ["BP", "Drug", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.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" tab
node.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" options
node.setPropertyValue("binomial_removal_criterion", "LR")
node.setPropertyValue("binomial_probability_removal", 0.2)
```

Tabla 123. Propiedades de logregnode.

Propiedades de logregnode	Valores	Descripción de la propiedad
objetivo	*campo*	Los modelos de regresión logística requieren un único campo objetivo y uno o más campos de entrada. Los campos de frecuencia y ponderación no se usan. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
logistic_procedure	Binomial Multinomial	
include_constant	*tag*	
mode	Simple Expert	
método	Intro Stepwise Forwards Backwards BackwardsStepwise	
binomial_method	Intro Forwards Backwards	
model_type	MainEffects FullFactorial Custom	Si FullFactorial se especifica como el tipo de modelo, no se ejecutarán los métodos por pasos, aunque así se indique. En su lugar, el método utilizado será Enter. Si el tipo de modelo se establece en Custom pero no se ha especificado ningún campo personalizado, se generará un modelo de efectos principales.

Tabla 123. Propiedades de logregnode (continuación).

Propiedades de logregnode	Valores	Descripción de la propiedad
custom_terms	[*[BP Sexo][BP][Edad]*]	
multinomial_base_category	*cadena*	Especifica cómo se determina la categoría de referencia.
binomial_categorical_input	*cadena*	
binomial_input_contrast	Indicator Simple Diferencia Helmert Repeated Polinómico Desviación	Propiedad con clave para la entrada categórica que especifica cómo se determina el contraste.
binomial_input_category	Primero Last	Propiedad con clave para la entrada categórica que especifica cómo se determina la categoría de referencia.
scale	Ninguno UserDefined Pearson Deviance	
scale_value	*number*	
all_probabilities	*tag*	
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	*tag*	
max_terms	*number*	
entry_criterion	Puntuación LR	
removal_criterion	LR Wald	
probability_entry	*number*	
probability_removal	*number*	
binomial_probability_entry	*number*	
binomial_probability_removal	*number*	
requirements	HierarchyDiscrete HierarchyAll Containment Ninguno	
max_iterations	*number*	
max_steps	*number*	

Tabla 123. Propiedades de logregnode (continuación).

Propiedades de logregnode	Valores	Descripción de la propiedad
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	*tag*	
history_steps	*number*	
summary	*tag*	
likelihood_ratio	*tag*	
asymptotic_correlation	*tag*	
goodness_fit	*tag*	
parameters	*tag*	
confidence_interval	*number*	
asymptotic_covariance	*tag*	
classification_table	*tag*	
stepwise_summary	*tag*	
info_criteria	*tag*	
monotonicity_measures	*tag*	
binomial_output_display	at_each_step at_last_step	
binomial_goodness_of_fit	*tag*	
binomial_parameters	*tag*	
binomial_iteration_history	*tag*	
binomial_classification_plots	*tag*	
binomial_ci_enable	*tag*	
binomial_ci	*number*	
binomial_residual	valores atípicos all	
binomial_residual_enable	*tag*	
binomial_outlier_threshold	*number*	
binomial_classification_cutoff	*number*	
binomial_removal_criterion	LR Wald Conditional	
calculate_variable_importance	*tag*	
calculate_raw_propensities	*tag*	

Propiedades de neuralnetnode

Precaución: Una versión más reciente del nodo de modelado Red neural, con características mejoradas, está disponible en esta versión y se describe en la sección siguiente (*neuralnetwork*). Aunque aún puede generar y puntuar un modelo con la versión anterior, recomendamos que actualice sus scripts para que se use la nueva versión. Los detalles de la versión anterior se conservan aquí como referencia.

Ejemplo

```
node = stream.create("neuralnet", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("targets", ["Drug"])
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
# "Model" tab
node.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")
# sección "Opciones de experto para método múltiple"
node.setPropertyValue("m_topologies", "5 30 5; 2 20 3, 1 10 1")
node.setPropertyValue("m_non_pyramids", False)
node.setPropertyValue("m_persistence", 100)
```

Tabla 124. Propiedades de neuralnetnode

Propiedad de neuralnetnode	Valores	Descripción de la propiedad
targets	[*field1 ... fieldN*]	El nodo Red neuronal espera uno o varios campos objetivo y uno o más campos de entrada. Los campos de frecuencia y ponderación se omiten. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
method	Quick Dynamic Multiple Prune ExhaustivePrune RBFN	
prevent_overtrain	*flag*	
train_pct	*número*	
set_random_seed	*flag*	
random_seed	*número*	
mode	Simple Valores avanzados	
stop_on	Predeterminado Exactitud Cycles Hora	Modo de parada.
exactitud	*número*	Precisión de parada.

Tabla 124. Propiedades de neuralnetnode (continuación)

Propiedad de neuralnetnode	Valores	Descripción de la propiedad
cycles	*número*	Ciclos para entrenar.
time	*número*	Tiempo para entrenar (minutos).
continue	*flag*	
show_feedback	*flag*	
binary_encode	*flag*	
use_last_model	*flag*	
gen_logfile	*flag*	
logfile_name	*cadena*	
alpha	*número*	
initial_eta	*número*	
high_eta	*número*	
low_eta	*número*	
eta_decay_cycles	*número*	
hid_layers	One Two Three	
hl_units_one	*número*	
hl_units_two	*número*	
hl_units_three	*número*	
persistence	*número*	
m_topologies	*cadena*	
m_non_pyramids	*flag*	
m_persistence	*número*	
p_hid_layers	One Two Three	
p_hl_units_one	*número*	
p_hl_units_two	*número*	
p_hl_units_three	*número*	
p_persistence	*número*	
p_hid_rate	*número*	
p_hid_pers	*número*	
p_inp_rate	*número*	
p_inp_pers	*número*	
p_overall_pers	*número*	
r_persistence	*número*	
r_num_clusters	*número*	
r_eta_auto	*flag*	
r_alpha	*número*	
r_eta	*número*	

Tabla 124. Propiedades de neuralnetnode (continuación)

Propiedad de neuralnetnode	Valores	Descripción de la propiedad
optimize	Speed Memory	Se utiliza para especificar si la generación del modelo se debe optimizar para la velocidad o la memoria.
calculate_variable_importance	*flag*	Nota: La propiedad sensitivity_analysis utilizada en versiones anteriores se ha desaprobado en favor de esta propiedad. La propiedad anterior se sigue admitiendo, pero se recomienda calculate_variable_importance.
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propiedades de neuralnetworknode

El nodo Red neuronal utiliza un modelo simplificado que emula el modo en que el cerebro humano procesa la información: Funciona simultaneando un número elevado de unidades simples de procesamiento interconectadas que parecen versiones abstractas de neuronas. Las redes neuronales son dispositivos eficaces de cálculo de funciones generales y requieren un conocimiento matemático o estadístico mínimo para entrenarlas o aplicarlas.

Ejemplo

```
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")
```

Tabla 125. Propiedades de neuralnetwork

Propiedades de neuralnetworknode	Valores	Descripción de la propiedad
targets	[*field1 ... fieldN*]	Especifica campos objetivo.
inputs	[*field1 ... fieldN*]	Campos de predictor utilizados por el modelo.
splits	*[field1 ... fieldN*	Especifica el campo o campos para utilizar en el modelado de divisiones.
use_partition	*flag*	Si se ha definido un campo de partición, esta opción garantiza que sólo se utilizarán los datos de la partición de entrenamiento para la generación del modelo.
continue	*flag*	Continuar entrenando modelo existente.
objective	Standard Agregación autodocimante Aumento psm	psm se utiliza para conjuntos de datos de grandes dimensiones y requiere una conexión al Servidor.
method	MultilayerPerceptron RadialBasisFunction	
use_custom_layers	*flag*	

Tabla 125. Propiedades de neuralnetwork (continuación)

Propiedades de neuralnetworknode	Valores	Descripción de la propiedad
first_layer_units	*número*	
second_layer_units	*número*	
use_max_time	*flag*	
max_time	*número*	
use_max_cycles	*flag*	
max_cycles	*número*	
use_min_accuracy	*flag*	
min_accuracy	*número*	
combining_rule_categorical	Voting HighestProbability HighestMeanProbability	
combining_rule_continuous	Media Mediana	
component_models_n	*número*	
overfit_prevention_pct	*número*	
use_random_seed	*flag*	
random_seed	*número*	
missing_values	listwiseDeletion missingValueImputation	
use_model_name	*booleano*	
model_name	*cadena*	
seguridad	onProbability onIncrease	
score_category_probabilities	*flag*	
max_categories	*número*	
score_propensity	*flag*	
use_custom_name	*flag*	
custom_name	*cadena*	
tooltip	*cadena*	
palabras clave	*cadena*	
annotation	*cadena*	

Propiedades de questnode

El nodo QUEST proporciona un método de clasificación binario para generar árboles de decisión; está diseñado para reducir el tiempo de procesamiento necesario para realizar los análisis de C&RT y reducir la tendencia de los métodos de clasificación de árboles para favorecer a las entradas que permitan realizar más divisiones. Los campos de entrada pueden ser continuos (rango numérico), sin embargo el campo objetivo debe ser categórico. Todas las divisiones son binarias.

Ejemplo

```
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)
```

Tabla 126. Propiedades de questnode

Propiedad de questnode	Valores	Descripción de la propiedad
target	*campo*	Los modelos QUEST requieren un único campo objetivo y uno o más campos de entrada. También se puede especificar un campo de frecuencia. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
continue_training_existing_model	*flag*	
objective	Standard Aumento Agregación autodocimante psm	psm se utiliza para conjuntos de datos de grandes dimensiones y requiere una conexión al Servidor.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*flag*	
tree_directives	*cadena*	
use_max_depth	Predeterminado Personalizado	
max_depth	*entero*	Máxima profundidad del árbol, desde 0 a 1000. Sólo se utiliza si use_max_depth = Custom.
prune_tree	*flag*	Poda del árbol para evitar sobreajustes.
use_std_err	*flag*	Use la diferencia máxima en riesgos (en errores estándar).
std_err_multiplier	*número*	Diferencia máxima.
max_surrogates	*número*	Número máximo de sustitutos.
use_percentage	*flag*	
min_parent_records_pc	*número*	
min_child_records_pc	*número*	
min_parent_records_abs	*número*	
min_child_records_abs	*número*	
use_costs	*flag*	
costes	*structured*	Propiedad estructurada.

Tabla 126. Propiedades de questnode (continuación)

Propiedad de questnode	Valores	Descripción de la propiedad
priors	Datos Equal Personalizado	
custom_priors	*structured*	Propiedad estructurada.
adjust_priors	*flag*	
trails	*número*	Número de modelos de componente para un aumento o agregación autodocimante.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Regla de combinación predeterminada para objetivos categóricos.
range_ensemble_method	Media Mediana	Regla de combinación predeterminada para objetivos continuos.
large_boost	*flag*	Aplicar aumento a conjunto de datos muy grandes.
split_alpha	*número*	Nivel de significancia para división.
train_pct	*número*	Conjunto de prevención sobreajustado.
set_random_seed	*flag*	Opción replicar resultados.
seed	*número*	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propiedades de regressionnode

La regresión lineal es una técnica de estadístico común utilizada para resumir datos y realizar predicciones ajustando una superficie o línea recta que minimice las discrepancias existentes entre los valores de salida reales y los predichos.

Nota: El nodo Lineal reemplazará al nodo Regresión en una versión futura. Recomendamos que a partir de ahora utilice modelos lineales para la regresión lineal.

Ejemplo

```
node = stream.create("regression", "My node")
# "Fields" tab
node.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" tab
node.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" tab
node.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)
```

Tabla 127. Propiedades de regressionnode

Propiedad de regressionnode	Valores	Descripción de la propiedad
target	*campo*	Los modelos de regresión requieren un único campo objetivo y uno o más campos de entrada. También se puede especificar un campo de ponderación. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
method	Intro Stepwise Backwards Adelante	
include_constant	*flag*	
use_weight	*flag*	
weight_field	*campo*	
mode	Simple Valores avanzados	
complete_records	*flag*	
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	Utilice comillas dobles para los argumentos.
stepping_method	useP useF	useP : utilizar la probabilidad de F useF: utilizar el valor F
probability_entry	*número*	

Tabla 127. Propiedades de regressionnode (continuación)

Propiedad de regressionnode	Valores	Descripción de la propiedad
probability_removal	*número*	
F_value_entry	*número*	
F_value_removal	*número*	
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*	

Propiedades de sequencenode

El nodo Secuencia encuentra reglas de asociación en datos secuenciales o en datos ordenados en el tiempo. Una secuencia es una lista de conjuntos de elementos que tiende a producirse en un orden previsible. Por ejemplo, si un cliente compra una cuchilla y una loción para después del afeitado, probablemente comprará crema para afeitar la próxima vez que vaya a comprar. El nodo Secuencia se basa en el algoritmo de reglas de asociación de CARMA, que utiliza un método de dos pasos para encontrar las secuencias.

Ejemplo

```
node = stream.create("sequence", "My node")
# "Fields" tab
node.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" tab
node.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" tab
node.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)
```

Tabla 128. Propiedades de sequencenode

Propiedad de sequencenode	Valores	Descripción de la propiedad
id_field	*campo*	Para crear un modelo de secuencias, es necesario especificar un campo de ID, un campo de tiempo opcional y uno o varios campos de contenido. Los campos de ponderación y frecuencia no se usan. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
time_field	*campo*	
use_time_field	*flag*	
content_fields	[*field1 ... fieldn*]	
contiguous	*flag*	
min_supp	*número*	
min_conf	*número*	
max_size	*número*	
max_predictions	*número*	
mode	Simple Valores avanzados	
use_max_duration	*flag*	
max_duration	*número*	
use_gaps	*flag*	
min_item_gap	*número*	
max_item_gap	*número*	
use_pruning	*flag*	
pruning_value	*número*	
set_mem_sequences	*flag*	
mem_sequences	*entero*	

Propiedades de slrmnode

El nodo Modelo de respuesta de autoaprendizaje (SLRM) permite crear un modelo en el que un solo caso nuevo o un pequeño número de casos nuevos se pueden utilizar para volver a calcular el modelo sin tener que entrenar de nuevo el modelo utilizando todos los datos.

Ejemplo

```
node = stream.create("slrm", "My node")
node.setPropertyValue("target", "Offer")
node.setPropertyValue("target_response", "Response")
node.setPropertyValue("inputs", ["Cust_ID", "Age", "Ave_Bal"])
```

Tabla 129. Propiedades de slrmnode

Propiedades de slrmnode	Valores	Descripción de la propiedad
target	*campo*	El campo objetivo debe ser un campo nominal o marca. También se puede especificar un campo de frecuencia. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
target_response	*campo*	El tipo debe ser marca.
continue_training_existing_model	*flag*	
target_field_values	*flag*	Utilizar todos: Usar todos los valores del origen. Especifique: Son necesarios determinados valores.
target_field_values_specify	*[campo1 ... campoN]*	
include_model_assessment	*flag*	
model_assessment_random_seed	*número*	Debe ser un número real.
model_assessment_sample_size	*número*	Debe ser un número real.
model_assessment_iterations	*número*	Número de iteraciones.
display_model_evaluation	*flag*	
max_predictions	*número*	
randomization	*número*	
scoring_random_seed	*número*	
sort	Ascending Descending	Especifica si se mostrarán primero las ofertas con las puntuaciones más altas o más bajas.
model_reliability	*flag*	
calculate_variable_importance	*flag*	

Propiedades de statisticsmodelnode

El nodo Modelo Statistics permite analizar y trabajar con sus datos ejecutando los procedimientos de IBM SPSS Statistics que producen PMML. Este nodo requiere una copia de IBM SPSS Statistics con licencia.

Las propiedades de este nodo están descritas en "Propiedades de statisticsmodelnode" en la página 302.

Propiedades de stpnode

El nodo Predicción espacio-temporal (STP) utiliza datos que contienen datos de ubicación, campos de entrada para la predicción (predictores), un campo de hora y un campo de objetivo. Cada ubicación tiene muchas filas en los datos que representan los valores de cada predictor en cada tiempo de medición. Después de analizar los datos, se puede utilizar para predecir los valores de objetivo en cualquier ubicación dentro de los datos shape que se utilizan en el análisis.

Tabla 130. Propiedades de stpnode

Propiedades de stpnode	Tipo de datos	Descripción de la propiedad
Pestaña **Campos**		
target	*campo*	Este es el campo de destino.
location	*campo*	Campo de ubicación del modelo. Sólo se permiten campos geoespaciales.
location_label	*campo*	Campo categórico a utilizar en la salida para etiquetar las ubicaciones elegidas en location
time_field	*campo*	Campo de hora del modelo. Sólo se permiten campos con medición continua, y el tipo de almacenamiento debe ser hora, fecha, indicación de fecha y hora o entero.
inputs	*[field1 ... campoN]*	Lista de campos de entrada.
Pestaña **Intervalos de tiempo**		
interval_type_timestamp	Años Trimestres Meses Weeks Days Hours Minutes Seconds	
interval_type_date	Años Trimestres Meses Weeks Days	
interval_type_time	Hours Minutes Seconds	Limita el número de días por semana que se tienen en cuenta al crear el índice de hora que utiliza STP para el cálculo
interval_type_integer	Períodos (Sólo campos de índice de hora, almacenamiento Entero)	Intervalo en el que se convertirá el conjunto de datos. La selección disponible depende del tipo de almacenamiento del campo elegido como time_field para el modelo.
period_start	*entero*	
start_month	Enero Febrero Marzo Abril Mayo Junio Julio Agosto Septiembre Octubre Noviembre Diciembre	El mes desde el que el modelo empezará a indexar (por ejemplo, si se establece en March pero el primer registro del conjunto de datos es January, el modelo omitirá los primeros dos registros y comenzará a indexar en marzo.

Tabla 130. Propiedades de stpnode (continuación)

Propiedades de stpnode	Tipo de datos	Descripción de la propiedad
week_begins_on	Sunday Monday Tuesday Wednesday Jueves Friday Saturday	Punto de partida para el índice temporal creado por STP a partir de los datos
days_per_week	*entero*	Mínimo 1, máximo 7, con incrementos de 1
hours_per_day	*entero*	El número de horas que el modelo cuenta en un día. Si se establece en 10, el modelo empezará a indexar en la hora day_begins_at y continuará indexando durante 10 horas, y luego saltará al siguiente valor que coincida con el valor day_begins_at, etc.
day_begins_at	00:00 01:00 02:00 03:00 ... 23:00	Establece el valor de hora desde el que el modelo inicia la indexación.
interval_increment	1 2 3 4 5 6 10 12 15 20 30	Este valor de incremento es de minutos o segundos. Determina dónde el modelo crea índices a partir de los datos. Así que con un incremento de 30 y el intervalo de tipo seconds, el modelo crea un índice a partir de los datos cada 30 segundos.
data_matches_interval	*Boolean*	Si se establece en N, la conversión de los datos al interval_type normal se produce antes de que se construya el modelo. Si los datos ya tiene el formato correcto, e interval_type y sus valores asociados coinciden con sus datos, establézcalo en Y para evitar la conversión o la agregación de los datos. Si lo establece en Y, se inhabilitan todos los controles de agregación.
agg_range_default	Sum Media Mín Máx Median 1stQuartile 3rdQuartile	Determina el método de agregación predeterminado que se utiliza para los campos continuos. Los campos continuos que no estén incluidos específicamente en la agregación predeterminada se agregarán usando el método aquí indicado.

Tabla 130. Propiedades de stpnode (continuación)

Propiedades de stpnode	Tipo de datos	Descripción de la propiedad
custom_agg	[[campo, método de agregación],[]..] Demo: [['x5' 'FirstQuartile']['x4' 'Sum']]	Propiedad estructurada: Parámetro de script: custom_agg Por ejemplo: set :stpnode.custom_agg = [[campo1 función] [campo2 función]] Donde función es la función de agregación a utilizar con dicho campo.
Pestaña **Procedimientos básicos**		
include_intercept	*marca*	
max_autoregressive_lag	*entero*	Mínimo 1, máximo 5, en incrementos de 1. Es el número de registros previos necesarios para una predicción. Por lo tanto, si se establece en 5, por ejemplo, entonces los 5 registros se utilizan para crear un nuevo pronóstico. El número de registros especificado aquí a partir de los datos de construcción se incorporan en el modelo y, por lo tanto, el usuario no necesita proporcionar los datos de nuevo al puntuar el modelo.
estimation_method	Parametric Nonparametric	Método para modelar la matriz de covarianzas espacial
parametric_model	Gaussian Exponential PoweredExponential	Parámetro de orden para el modelo de covarianza espacial de tipo Parametric
exponential_power	*número*	Nivel alimentación para el modelo PoweredExponential. Mínimo 1, máximo 2.
Pestaña **Avanzada**		
max_missing_values	*entero*	Porcentaje máximo de registros con valores faltantes que se permite en el modelo.
significación	*número*	Nivel de significación para pruebas de hipótesis en la construcción del modelo. Especifica el valor de significación para todas las pruebas en la estimación del modelo STP, incluidas dos pruebas de Bondad de ajuste, pruebas F de efectos y pruebas T de coeficiente.
Pestaña **Salida**		
model_specifications	*marca*	
temporal_summary	*marca*	

Tabla 130. Propiedades de stpnode (continuación)

Propiedades de stpnode	Tipo de datos	Descripción de la propiedad
location_summary	*marca*	Determina si la tabla Resumen de ubicación se incluye en la salida del modelo.
model_quality	*marca*	
test_mean_structure	*marca*	
mean_structure_coefficients	*marca*	
autoregressive_coefficients	*marca*	
test_decay_space	*marca*	
parametric_spatial_covariance	*marca*	
correlations_heat_map	*marca*	
correlations_map	*marca*	
location_clusters	*marca*	
similarity_threshold	*número*	Umbral en el cual los clústeres de salida se consideran lo suficientemente parecidos para que se fusionen en un único clúster.
max_number_clusters	*entero*	Límite superior para el número de clústeres que se pueden incluir en la salida del modelo.
Pestaña **Opciones de modelo**		
use_model_name	*marca*	
model_name	*cadena*	
uncertainty_factor	*número*	Mínimo 0, máximo 100. Determina el aumento de la incertidumbre (error) aplicado a las predicciones en el futuro. Es el límite superior e inferior para las predicciones.

Propiedades de svmnode

El nodo Máquina de vectores de soporte (SVM) le permite clasificar datos en uno o dos grupos sin que haya un ajuste por exceso. SVM funciona bien con conjuntos de datos grandes, como aquellos con un gran número de campos de entrada.

Ejemplo

```
node = stream.create("svm", "My node")
# pestaña Experto
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("kernel", "Polynomial")
node.setPropertyValue("gamma", 1.5)
```

Tabla 131. Propiedades de svmnode.

Propiedades de svmnode	Valores	Descripción de la propiedad
all_probabilities	*tag*	

Tabla 131. Propiedades de svmnode (continuación).

Propiedades de svmnode	Valores	Descripción de la propiedad
stopping_criteria	1.0E-1 1.0E-2 1.0E-3 (valor predeterminado) 1.0E-4 1.0E-5 1.0E-6	Determina cuándo detener el algoritmo de optimización.
regularization	*number*	También se conoce como el parámetro C.
precision	*number*	Sólo se utiliza si el nivel de medición del campo objetivo es Continuo.
kernel	RBF (valor predeterminado) Polinómico Sigmoide Lineal	Tipo de función kernel utilizada para la transformación.
rbf_gamma	*number*	Sólo se utiliza si kernel es RBF.
gamma	*number*	Sólo se utiliza si kernel es Polinómico o Sigmoide.
bias	*number*	
grado	*number*	Sólo se utiliza si kernel es Polinómico.
calculate_variable_importance	*tag*	
calculate_raw_propensities	*tag*	
calculate_adjusted_ propensities	*tag*	
adjusted_propensity_partition	Test Validation	

Propiedades de tcmnode

El modelado causal temporal intenta descubrir relaciones causales clave en datos de series temporales. En el modelado causal temporal, especifique un conjunto de series de objetivos y un conjunto de entradas candidato para estos objetivos. El procedimiento crea un modelo de serie temporal autorregresivo para cada objetivo e incluye solo estas entradas que tienen la relación causal más significativa con el objetivo.

Tabla 132. Propiedades de tcmnode

Propiedades de tcmnode	Valores	Descripción de la propiedad
custom_fields	*Boolean*	
dimensionlist	[*dimension1 ... dimensionN*]	
data_struct	Multiple Única	
metric_fields	*campos*	
both_target_and_input	[*f1 ... fN*]	
targets	[*f1 ... fN*]	
candidate_inputs	[*f1 ... fN*]	
forced_inputs	[*f1 ... fN*]	

Tabla 132. Propiedades de tcmnode (continuación)

Propiedades de tcmnode	Valores	Descripción de la propiedad
use_timestamp	Marca de tiempo Period	
input_interval	Ninguno Desconocido Año Trimestre Mes Semana Día Hora Hour_nonperiod Minuto Minute_nonperiod Segundo Second_nonperiod	
period_field	*string*	
period_start_value	*entero*	
num_days_per_week	*entero*	
start_day_of_week	Sunday Monday Martes Miércoles Jueves Viernes Sábado	
num_hours_per_day	*entero*	
start_hour_of_day	*entero*	
timestamp_increments	*entero*	
cyclic_increments	*entero*	
cyclic_periods	*lista*	
output_interval	Ninguno Año Trimestre Mes Semana Día Hora Minuto Segundo	
is_same_interval	El mismo Notsame	
cross_hour	*Boolean*	
aggregate_and_distribute	*lista*	
aggregate_default	Media Sum Modo Mín Máx	
distribute_default	Media Sum	

Tabla 132. Propiedades de tcmnode (continuación)

Propiedades de tcmnode	Valores	Descripción de la propiedad
group_default	Media Sum Modo Mín Máx	
missing_imput	Linear_interp Series_mean K_mean K_meridian Linear_trend Ninguno	
k_mean_param	*entero*	
k_median_param	*entero*	
missing_value_threshold	*entero*	
conf_level	*entero*	
max_num_predictor	*entero*	
max_lag	*entero*	
epsilon	*número*	
threshold	*entero*	
is_re_est	*Boolean*	
num_targets	*entero*	
percent_targets	*entero*	
fields_display	*lista*	
series_display	*lista*	
network_graph_for_target	*Boolean*	
sign_level_for_target	*número*	
fit_and_outlier_for_target	*Boolean*	
sum_and_para_for_target	*Boolean*	
impact_diag_for_target	*Boolean*	
impact_diag_type_for_target	Efecto Cause Both	
impact_diag_level_for_target	*entero*	
series_plot_for_target	*Boolean*	
res_plot_for_target	*Boolean*	
top_input_for_target	*Boolean*	
forecast_table_for_target	*Boolean*	
same_as_for_target	*Boolean*	
network_graph_for_series	*Boolean*	
sign_level_for_series	*número*	
fit_and_outlier_for_series	*Boolean*	
sum_and_para_for_series	*Boolean*	
impact_diagram_for_series	*Boolean*	

Tabla 132. Propiedades de tcmnode (continuación)

Propiedades de tcmnode	Valores	Descripción de la propiedad
impact_diagram_type_for_series	Efecto Cause Both	
impact_diagram_level_for_series	*entero*	
series_plot_for_series	*Boolean*	
residual_plot_for_series	*Boolean*	
forecast_table_for_series	*Boolean*	
outlier_root_cause_analysis	*Boolean*	
causal_levels	*entero*	
outlier_table	Interactive Lista dinámica Both	
rmsp_error	*Boolean*	
bic	*Boolean*	
r_square	*Boolean*	
outliers_over_time	*Boolean*	
series_transormation	*Boolean*	
use_estimation_period	*Boolean*	
estimation_period	Times Observación	
observations	*lista*	
observations_type	Latest Más antiguo	
observations_num	*entero*	
observations_exclude	*entero*	
extend_records_into_future	*Boolean*	
forecastperiods	*entero*	
max_num_distinct_values	*entero*	
display_targets	FIXEDNUMBER PERCENTAGE	
goodness_fit_measure	ROOTMEAN BIC RSQUARE	
top_input_for_series	*Boolean*	
aic	*Boolean*	
rmse	*Boolean*	

Propiedades de timeseriesnode

El nodo Serie temporal estima modelos de suavizado exponencial, modelos autorregresivos integrados de media móvil (ARIMA) univariados y modelos ARIMA (o de función de transferencia) multivariados para series temporales y genera datos de previsiones. Un nodo Serie temporal debe ir siempre precedido por un nodo Intervalos de tiempo.

Ejemplo

```
node = stream.create("timeseries", "My node")
node.setPropertyValue("method", "Exsmooth")
node.setPropertyValue("exsmooth_model_type", "HoltsLinearTrend")
node.setPropertyValue("exsmooth_transformation_type", "None")
```

Tabla 133. Propiedades de timeseriesnode

Propiedades de timeseriesnode	**Valores**	**Descripción de la propiedad**
targets	*campo*	El nodo Serie temporal prevé uno o más objetivos, utilizando opcionalmente uno o más campos de entrada como predictores. Los campos de frecuencia y ponderación no se usan. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
continue	*flag*	
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	Ninguno SquareRoot NaturalLog	
arima_p	*entero*	
arima_d	*entero*	
arima_q	*entero*	
arima_sp	*entero*	

Tabla 133. Propiedades de timeseriesnode (continuación)

Propiedades de **timeseriesnode**	Valores	Descripción de la propiedad
arima_sd	*entero*	
arima_sq	*entero*	
arima_transformation_type	Ninguno SquareRoot NaturalLog	
arima_include_constant	*flag*	
tf_arima_p. *nombredecampo*	*entero*	Para funciones de transferencia.
tf_arima_d. *nombredecampo*	*entero*	Para funciones de transferencia.
tf_arima_q. *nombredecampo*	*entero*	Para funciones de transferencia.
tf_arima_sp. *nombredecampo*	*entero*	Para funciones de transferencia.
tf_arima_sd. *nombredecampo*	*entero*	Para funciones de transferencia.
tf_arima_sq. *nombredecampo*	*entero*	Para funciones de transferencia.
tf_arima_delay. *nombredecampo*	*entero*	Para funciones de transferencia.
tf_arima_transformation_type. *nombredecampo*	Ninguno SquareRoot NaturalLog	Para funciones de transferencia.
arima_detect_outlier_mode	Ninguno 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	*entero*	
eventos	*campos*	
scoring_model_only	*flag*	Se utiliza para modelos con cifras muy grandes (cientos de miles) o series temporales.

Propiedades de treeasnode

El nodo Tree-AS solo está disponible si se dispone de una conexión a IBM SPSS Analytic Server. Este nodo es similar al nodo CHAID existente; no obstante, el nodo Tree-AS está diseñado para procesar datos masivos para crear un único árbol y muestra el modelo resultante en el visor de salidas añadido en SPSS Modeler versión 17. El nodo genera un árbol de decisiones usando un estadístico de chi-cuadrado (CHAID) para identificar las divisiones óptimas. Este uso de CHAID puede generar árboles no binarios, lo que significa que algunas divisiones tienen más de dos ramas. Los campos de entrada y objetivo pueden ser continuos (rango numérico) o categóricos. CHAID exhaustivo es una modificación de CHAID que examina con mayor precisión todas las divisiones posibles, aunque necesita más tiempo para realizar los cálculos.

Tabla 134. Propiedades de treeasnode

Propiedades **treeasnode**	Valores	Descripción de la propiedad
`target`	*campo*	En el nodo Tree-AS, los modelos CHAID requieren un único objetivo y uno o más campos de entrada. También se puede especificar un campo de frecuencia. Consulte el tema "Propiedades comunes de nodos de modelado" en la página 161 para obtener más información.
`method`	`chaid` `exhaustive_chaid`	
`max_depth`	*entero*	Profundidad máxima del árbol, de 0 a 20. El valor predeterminado es 5.
`num_bins`	*entero*	Solo se usa si los datos constan de entradas continuas. Establece el número de intervalos de frecuencia iguales que se van a usar en las entradas; las opciones son: 2, 4, 5, 10, 20, 25, 50 o 100.
`record_threshold`	*entero*	El número de registros en los que el modelo pasa de usar valores p a tamaños del efecto mientras se construye el árbol. El valor predeterminado es de 1.000.000; se incrementa o decrementa de 10.000 en 10.000.
`split_alpha`	*número*	Nivel de significancia para división.El valor debe estar comprendido entre 0,05 y 0,95.
`merge_alpha`	*número*	Nivel de significancia para fusión.El valor debe estar comprendido entre 0,05 y 0,95.
`bonferroni_adjustment`	*distintivo*	Los valores de significancia de ajuste utilizando el método de Bonferroni.
`effect_size_threshold_cont`	*número*	Establece el umbral del tamaño del efecto cuando se dividen los nodos y fusionan las categorías al usar un objetivo continuo. El valor debe estar comprendido entre 0,01 y 0,99.
`effect_size_threshold_cat`	*número*	Establece el umbral del tamaño del efecto cuando se dividen los nodos y fusionan las categorías al usar un objetivo categórico. El valor debe estar comprendido entre 0,01 y 0,99.

Tabla 134. Propiedades de treeasnode (continuación)

Propiedades treeasnode	Valores	Descripción de la propiedad
split_merged_categories	*distintivo*	Permitir segunda división de categorías fusionadas.
grouping_sig_level	*número*	Se usa para determinar cómo se forman los grupos de nodos o cómo se identifican los nodos inusuales.
chi_square	pearson likelihood_ratio	Método usado para calcular la estadística de chi cuadrado: Pearson o Razón de verosimilitud
minimum_record_use	use_percentage use_absolute	
min_parent_records_pc	*número*	El valor predeterminado es de 2. El mínimo es 1 y el máximo 100, en incrementos de 1. El valor de la rama padre debe ser superior que el de la rama hija.
min_child_records_pc	*número*	El valor predeterminado es de 1. El mínimo es 1 y el máximo 100, en incrementos de 1.
min_parent_records_abs	*número*	El valor predeterminado es 100. El mínimo es 1 y el máximo 100, en incrementos de 1. El valor de la rama padre debe ser superior que el de la rama hija.
min_child_records_abs	*número*	El valor predeterminado es 50. Mínimo 1, máximo 100, con incrementos de 1.
epsilon	*número*	Cambio mínimo en frecuencias de casillas esperadas.
max_iterations	*número*	Número máximo de iteraciones para la convergencia.
use_costs	*distintivo*	
costes	*structured*	Propiedad estructurada. El formato es una lista de 3 valores: el valor real, el valor pronosticado y el coste de una predicción errónea. Por ejemplo: tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])
default_cost_increase	ninguno lineal cuadrado personalizada	**Nota:** Solo se habilita en el caso de objetivos ordinales. Establece los valores predeterminados de la matriz de costes.
calculate_conf	*distintivo*	
display_rule_id	*distintivo*	Añade un campo en el resultado de puntuación que indica el ID para el nodo terminal al que se asigna cada registro.

Propiedades de twostepnode

El nodo Bietápico es un método de agrupación en clústeres de dos pasos. El primer paso es hacer una única pasada por los datos para comprimir los datos de entrada de la fila en un conjunto de subclústeres administrable. El segundo paso utiliza un método de agrupación en clústeres jerárquica para fundir progresivamente los subclústeres en clústeres cada vez más grandes. El bietápico tiene la ventaja de estimar automáticamente el número óptimo de clústeres para los datos de entrenamiento. Puede gestionar tipos de campos mixtos y grandes conjuntos de datos eficazmente.

Ejemplo

```
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)
```

Tabla 135. Propiedades de twostepnode

Propiedad de twostepnode	Valores	Descripción de la propiedad
inputs	[*field1 ... fieldN*]	Los modelos bietápicos utilizan una lista de campos de entrada, pero no de campos objetivo. Los campos de ponderación y frecuencia no se reconocen. Consulte el tema “Propiedades comunes de nodos de modelado” en la página 161 para obtener más información.
standardize	*flag*	
exclude_outliers	*flag*	
percentage	*número*	
cluster_num_auto	*flag*	
min_num_clusters	*número*	
max_num_clusters	*número*	
num_clusters	*número*	
cluster_label	Cadena Number	
label_prefix	*cadena*	
distance_measure	Euclidean Loglikelihood	
clustering_criterion	AIC BIC	

Propiedades de twostepAS

El clúster Bietápico es una herramienta de exploración diseñada para revelar agrupaciones naturales (o clústeres) dentro de un conjunto de datos que de otro modo no serían evidentes. El algoritmo que emplea este procedimiento incluye varias atractivas características que lo hacen diferente de las técnicas de agrupación en clústeres tradicionales, como el tratamiento de variables categóricas y continuas, la selección automática de número de clústeres y la escalabilidad.

Tabla 136. Propiedades de twostepAS

Propiedades de twostepAS	Valores	Descripción de la propiedad
inputs	[*f1 ... fN*]	Los modelos bietápicos utilizan una lista de campos de entrada, pero no de objetivos. Los campos de ponderación y frecuencia no se reconocen.
use_predefined_roles	Booleano	Default=True
use_custom_field_assignments	Booleano	Default=False
cluster_num_auto	Booleano	Default=True
min_num_clusters	número entero	Default=2
max_num_clusters	número entero	Default=15
num_clusters	número entero	Default=5
clustering_criterion	AIC BIC	
automatic_clustering_method	use_clustering_criterion_setting Distance_jump Mínimo Máximo	
feature_importance_method	use_clustering_criterion_setting effect_size	
use_random_seed	Booleano	
random_seed	número entero	
distance_measure	Euclidean Loglikelihood	
include_outlier_clusters	Booleano	Default=True
num_cases_in_feature_tree_leaf_ is_less_than	número entero	Default=10
top_perc_outliers	número entero	Default=5
initial_dist_change_threshold	número entero	Default=0
leaf_node_maximum_branches	número entero	Default=8
non_leaf_node_maximum_branches	número entero	Default=8
max_tree_depth	número entero	Default=3
adjustment_weight_on_measurement_ level	número entero	Default=6
memory_allocation_mb	número	Default=512
delayed_split	Booleano	Default=True

Tabla 136. Propiedades de twostepAS (continuación)

Propiedades de twostepAS	**Valores**	**Descripción de la propiedad**
fields_to_standardize	[*f1 ... fN*]	
adaptive_feature_selection	Booleano	Default=True
featureMisPercent	número entero	Default=70
coefRange	número	Default=0,05
percCasesSingleCategory	número entero	Default=95
numCases	número entero	Default=24
include_model_specifications	Booleano	Default=True
include_record_summary	Booleano	Default=True
include_field_transformations	Booleano	Default=True
excluded_inputs	Booleano	Default=True
evaluate_model_quality	Booleano	Default=True
show_feature_importance bar chart	Booleano	Default=True
show_feature_importance_ word_cloud	Booleano	Default=True
show_outlier_clusters interactive_table_and_chart	Booleano	Default=True
show_outlier_clusters_pivot_table	Booleano	Default=True
across_cluster_feature_importance	Booleano	Default=True
across_cluster_profiles_pivot_ table	Booleano	Default=True
withinprofiles	Booleano	Default=True
cluster_distances	Booleano	Default=True
cluster_label	Cadena Número	
label_prefix	Serie de caracteres	

Capítulo 14. Propiedades de nodos de nugget de modelo

Los nodos de nugget de modelo comparten las mismas propiedades comunes que los otros nodos. Consulte el tema “Propiedades de nodos comunes” en la página 69 para obtener más información.

Propiedades de applyanomalydetectionnode

Los nodos de modelado Detección de anomalías pueden utilizarse para generar un nugget de modelo Detección de anomalías. El nombre de script de este nugget de modelo es *applyanomalydetectionnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte “Propiedades de anomalydetectionnode” en la página 161

Tabla 137. Propiedades de applyanomalydetectionnode.

Propiedades de applyanomalydetectionnode	Valores	Descripción de la propiedad
anomaly_score_method	FlagAndScore FlagOnly ScoreOnly	Determina que resultados se crean para puntuación.
num_fields	*entero*	Campos para informar.
discard_records	*flag*	Indica si los registros se descartan del resultado o no.
discard_anomalous_records	*flag*	Indicador de cuando descartar los registros anómalos o *no* anómalos. El valor predeterminado es off, que significa que se descartan los registros *no* anómalos. En caso contrario, si es on, se descartan los registros anómalos. Esta propiedad se activa sólo si la propiedad discard_records se activa.

Propiedades de applyapriorinode

Los nodos de modelado Apriori pueden utilizarse para generar un nugget de modelo Apriori. El nombre de script de este nugget de modelo es *applyapriorinode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte “Propiedades de apriorinode” en la página 163

Tabla 138. Propiedades de applyapriorinode.

Propiedades de applyapriorinode	Valores	Descripción de la propiedad
max_predictions	*número (entero)*	
ignore_unmatached	*flag*	
allow_repeats	*flag*	
check_basket	NoPredictions Predictions NoCheck	
criterio	Confianza Support RuleSupport Lift Capacidad de despliegue	

Propiedades de applyassociationrulesnode

El nodo de modelado de reglas de asociación se puede utilizar para generar un nugget de modelo de reglas de asociación. El nombre de script de este nugget de modelo es *applyassociationrulesnode*. Para obtener más información sobre los scripts para propio nodo de modelado, consulte "Propiedades associationrulesnode" en la página 164.

Tabla 139. Propiedades de applyassociationrulesnode

Propiedades de applyassociationrulesnode	Tipo de datos	Descripción de la propiedad
max_predictions	*entero*	Número máximo de reglas que se pueden aplicar a cada entrada de la puntuación.
criterio	Confianza Rulesupport Lift Conditionsupport Capacidad de despliegue	Seleccione la medida utilizada para determinar la fuerza de las reglas.
allow_repeats	*Boolean*	Determina si se incluyen las reglas con la misma predicción en la puntuación.
check_input	NoPredictions Predictions NoCheck	

Propiedades de applyautoclassifiernode

Los nodos de modelado de clasificador automático se pueden utilizar para crear un nugget de modelo Clasificador automático. El nombre de script de este nugget de modelo es *applyautoclassifiernode*. Para obtener más información sobre los scripts para el propio nodo de modelado, "Propiedades de autoclassifiernode" en la página 167

Tabla 140. Propiedades de applyautoclassifiernode.

Propiedades de applyautoclassifiernode	Valores	Descripción de la propiedad
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting HighestConfidence AverageRawPropensity	Especifica el método utilizado para determinar la puntuación del conjunto. Este conjunto sólo se aplica si el objetivo seleccionado es un campo de marca.
flag_voting_tie_selection	Random HighestConfidence RawPropensity	Si se selecciona un método de votación, especifica cómo se resolverán los empates. Este conjunto sólo se aplica si el objetivo seleccionado es un campo de marca.
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	Especifica el método utilizado para determinar la puntuación del conjunto. Este conjunto sólo se aplica si el objetivo seleccionado es un campo de conjunto.
set_voting_tie_selection	Random HighestConfidence	Si se selecciona un método de votación, especifica cómo se resolverán los empates. Este conjunto sólo se aplica si el objetivo seleccionado es un campo nominal.

Propiedades de applyautoclusternode

Los nodos de modelado de Clúster automático se pueden utilizar para crear un nugget de modelo Clúster automático. El nombre de script de este nugget de modelo es *applyautoclusternode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de nodo de agrupación en clústeres automática" en la página 169

Propiedades de applyautonumericnode

Los nodos de modelado autonumérico se pueden utilizar para crear un nugget de modelo Autonumérico. El nombre de script de este nugget de modelo es *applyautonumericnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, "Propiedades de autonumericnode" en la página 170

Tabla 141. Propiedades de applyautonumericnode.

Propiedades de applyautonumericnode	Valores	Descripción de la propiedad
calculate_standard_error	*flag*	

Propiedades de applybayesnetnode

Los nodos de modelado de red bayesiana pueden utilizarse para generar un nugget de modelo de red bayesiana. El nombre de script de este nugget de modelo es *applybayesnetnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de bayesnetnode" en la página 172.

Tabla 142. Propiedades de applybayesnetnode.

Propiedades de applybayesnetnode	Valores	Descripción de la propiedad
all_probabilities	*flag*	
raw_propensity	*flag*	
adjusted_propensity	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

Propiedades de applyc50node

Los nodos de modelado C5.0 pueden utilizarse para generar un nugget de modelo C5.0. El nombre de script de este nugget de modelo es *applyc50node*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de c50node" en la página 174.

Tabla 143. Propiedades de applyc50node.

Propiedades de applyc50node	Valores	Descripción de la propiedad
sql_generate	Nunca NoMissingValues	Se utiliza para establecer las opciones de generación de SQL durante la ejecución del conjunto de reglas.
calculate_conf	*flag*	Disponible cuando la generación de SQL está activada. Esta propiedad incluye los cálculos de confianza en el árbol generado.
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

Propiedades de applycarmanode

Los nodos de modelado CARMA pueden utilizarse para generar un nugget de modelo CARMA. El nombre de script de este nugget de modelo es *applycarmanode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de carmanode" en la página 175.

Propiedades de applycartnode

Se pueden utilizar los nodos de modelado C&RT para generar un nugget de modelo C&RT. El nombre de script de este nugget de modelo es *applycartnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de cartnode" en la página 176.

Tabla 144. Propiedades de applycartnode.

Propiedades de applycartnode	Valores	Descripción de la propiedad
sql_generate	Nunca MissingValues NoMissingValues	Se utiliza para establecer las opciones de generación de SQL durante la ejecución del conjunto de reglas.
calculate_conf	*flag*	Disponible cuando la generación de SQL está activada. Esta propiedad incluye los cálculos de confianza en el árbol generado.
display_rule_id	*flag*	Añade un campo en el resultado de puntuación que indica el ID para el nodo terminal al que se asigna cada registro.
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

Propiedades de applychaidnode

Los nodos de modelado CHAID pueden utilizarse para generar un nugget de modelo CHAID. El nombre de script de este nugget de modelo es *applychaidnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de chaidnode" en la página 178.

Tabla 145. Propiedades de applychaidnode.

Propiedades de applychaidnode	Valores	Descripción de la propiedad
sql_generate	Nunca MissingValues	
calculate_conf	*flag*	
display_rule_id	*flag*	Añade un campo en el resultado de puntuación que indica el ID para el nodo terminal al que se asigna cada registro.
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

Propiedades de applycoxregnode

Los nodos de modelado Cox pueden utilizarse para generar un nugget de modelo Cox. El nombre de script de este nugget de modelo es *applycoxregnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de coxregnode" en la página 180.

Tabla 146. Propiedades de applycoxregnode.

Propiedades de applycoxregnode	Valores	Descripción de la propiedad
future_time_as	Intervalos Campos	
time_interval	*number*	
num_future_times	*entero*	
time_field	*campo*	
past_survival_time	*campo*	
all_probabilities	*flag*	
cumulative_hazard	*flag*	

Propiedades de applydecisionlistnode

Los nodos de modelado Lista de decisiones pueden utilizarse para generar un nugget de modelo Lista de decisiones. El nombre de script de este nugget de modelo es *applydecisionlistnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de decisionlistnode" en la página 182.

Tabla 147. Propiedades de applydecisionlistnode.

Propiedades de applydecisionlistnode	Valores	Descripción de la propiedad
enable_sql_generation	*flag*	Cuando se establece en true, IBM SPSS Modeler intenta enviar el modelo Lista de decisiones a SQL.
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

Propiedades de applydiscriminantnode

Los nodos de modelado Discriminante pueden utilizarse para generar un nugget de modelo Discriminante. El nombre de script de este nugget de modelo es *applydiscriminantnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de discriminantnode" en la página 183.

Tabla 148. Propiedades de applydiscriminantnode.

Propiedades de applydiscriminantnode	Valores	Descripción de la propiedad
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

Propiedades de applyfactornode

Los nodos de modelado PCA/Factorial pueden utilizarse para generar un nugget de modelo PCA/Factorial. El nombre de script de este nugget de modelo es *applyfactornode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de factornode" en la página 185.

Propiedades de applyfeatureselectionnode

Los nodos de modelado Selección de características pueden utilizarse para generar un nugget de modelo Selección de características. El nombre de script de este nugget de modelo es *applyfeatureselectionnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de featureselectionnode" en la página 187.

Tabla 149. Propiedades de applyfeatureselectionnode.

Propiedades de applyfeatureselectionnode	Valores	Descripción de la propiedad
selected_ranked_fields		especifica qué campos clasificados se comprueban en el explorador de modelos.
selected_screened_fields		Especifica qué campos filtrados se comprueban en el explorador de modelos.

Propiedades de applygeneralizedlinearnode

Los nodos de modelado lineal generalizado (genlin) pueden utilizarse para generar un nugget de modelo lineal generalizado. El nombre de script de este nugget de modelo es *applygeneralizedlinearnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de genlinnode" en la página 188.

Tabla 150. Propiedades de applygeneralizedlinearnode.

Propiedades de applygeneralizedlinearnode	Valores	Descripción de la propiedad
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

Propiedades de applyglmmnode

Los nodos de modelado GLMM pueden utilizarse para generar un nugget de modelo GLMM. El nombre de script de este nugget de modelo es *applyglmmnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de glmmnode" en la página 192.

Tabla 151. Propiedades de applyglmmnode.

Propiedades de applyglmmnode	Valores	Descripción de la propiedad
confidence	onProbability onIncrease	Base para calcular el valor de confianza de la puntuación: probabilidad más alta predicha, o la diferencia entre la probabilidad más alta predicha y la segunda probabilidad más alta.
score_category_probabilities	*flag*	Si es True, genera las probabilidades predichas para objetivos categóricos. Se crea un campo para cada categoría. El valor predeterminado es False.

Tabla 151. Propiedades de applyglmmnode (continuación).

Propiedades de applyglmmnode	Valores	Descripción de la propiedad
max_categories	*entero*	Número máximo de categorías para el que se van a predecir las probabilidades. Sólo se utiliza si score_category_probabilities es True.
score_propensity	*flag*	Si se establece en True, genera puntuaciones de propensión en bruto (probabilidad de resultado "True") para modelos con objetivos de marca. Si las particiones están en vigor, también genera puntuaciones de propensión ajustadas en función de la partición de prueba. El valor predeterminado es False.

Propiedades de applykmeansnode

Los nodos de modelado K-medias pueden utilizarse para generar un nugget de modelo K-medias. El nombre de script de este nugget de modelo es *applykmeansnode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de kmeansnode" en la página 195.

Propiedades de applyknnnode

Los nodos de modelado KNN pueden utilizarse para generar un nugget de modelo KNN. El nombre de script de este nugget de modelo es *applyknnnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de knnnode" en la página 196.

Tabla 152. Propiedades de applyknnnode.

Propiedades de applyknnnode	Valores	Descripción de la propiedad
all_probabilities	*flag*	
save_distances	*flag*	

Propiedades de applykohonennode

Los nodos de modelado Kohonen pueden utilizarse para generar un nugget de modelo Kohonen. El nombre de script de este nugget de modelo es *applykohonennode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de c50node" en la página 174.

Propiedades de applylinearnode

Los nodos de modelado lineal pueden utilizarse para generar un nugget de modelo lineal. El nombre de script de este nugget de modelo es *applylinearnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de linearnode" en la página 199.

Tabla 153. Propiedades de applylinearnode.

Propiedades de linear	Valores	Descripción de la propiedad
use_custom_name	*flag*	
custom_name	*cadena*	
enable_sql_generation	*flag*	

Propiedades de applylinearasnode

Los nodos de modelado Linear-AS pueden utilizarse para generar un nugget de modelo Linear-AS. El nombre de script de este nugget de modelo es *applylinearasnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de linearasnode" en la página 200.

Tabla 154. Propiedades de applylinearasnode

Propiedad applylinearasnode	Valores	Descripción de la propiedad
enable_sql_generation	udf native	El valor predeterminado es udf.

Propiedades de applylogregnode

Los nodos de modelado Regresión logística pueden utilizarse para generar un nugget de modelo Regresión logística. El nombre de script de este nugget de modelo es *applylogregnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de logregnode" en la página 201.

Tabla 155. Propiedades de applylogregnode.

Propiedades de applylogregnode	Valores	Descripción de la propiedad
calculate_raw_propensities	*flag*	
calculate_conf	*marca*	
enable_sql_generation	*marca*	

Propiedades de applyneuralnetnode

Los nodos de modelado Red neuronal pueden utilizarse para generar un nugget de modelo Red neuronal. El nombre de script de este nugget de modelo es *applyneuralnetnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de neuralnetnode" en la página 206.

Precaución: Una versión más reciente del nodo de modelado Red neural, con características mejoradas, está disponible en esta versión y se describe en la sección siguiente (*applyneuralnetwork*). Aunque la versión anterior sigue estando disponible, le recomendamos actualizar sus scripts para que se usen la nueva versión. En este documento se incluyen detalles de la versión anterior como referencia, pero en versiones futuras dejará de ser compatible.

Tabla 156. Propiedades de applyneuralnetnode.

Propiedades de applyneuralnetnode	Valores	Descripción de la propiedad
calculate_conf	*flag*	Disponible cuando la generación de SQL está activada. Esta propiedad incluye los cálculos de confianza en el árbol generado.
enable_sql_generation	*flag*	
nn_score_method	Diferencia SoftMax	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

Propiedades de applyneuralnetworknode

Los nodos de modelado Red neuronal pueden utilizarse para generar un nugget de modelo Red neuronal. El nombre de script de este nugget de modelo es *applyneuralnetworknode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de neuralnetworknode" en la página 208.

Tabla 157. Propiedades de applyneuralnetworknode

Propiedades de applyneuralnetworknode	Valores	Descripción de la propiedad
use_custom_name	*marca*	
custom_name	*string*	
confidence	onProbability onIncrease	
score_category_probabilities	*marca*	
max_categories	*número*	
score_propensity	*marca*	

Propiedades de applyquestnode

Los nodos de modelado QUEST pueden utilizarse para generar un nugget de modelo QUEST. El nombre de script de este nugget de modelo es *applyquestnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de questnode" en la página 209.

Tabla 158. Propiedades de applyquestnode.

Propiedades de applyquestnode	Valores	Descripción de la propiedad
sql_generate	Nunca MissingValues NoMissingValues	
calculate_conf	*flag*	
display_rule_id	*flag*	Añade un campo en el resultado de puntuación que indica el ID para el nodo terminal al que se asigna cada registro.
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

Propiedades de applyr

Los nodos de modelado R pueden utilizarse para generar un nugget de modelo R. El nombre de script de este nugget de modelo es *applyr*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de buildr" en la página 173.

Tabla 159. Propiedades de applyr

Propiedades de applyr	Valores	Descripción de la propiedad
score_syntax	*cadena*	Sintaxis de scripts R para la puntuación del modelo.
convert_flags	StringsAndDoubles LogicalValues	Opción para convertir campos de distintivos.

Tabla 159. Propiedades de applyr (continuación)

Propiedades de applyr	Valores	Descripción de la propiedad
convert_datetime	*flag*	Opción para convertir las variables con los formatos de fecha o de fecha y hora para formatos de fecha/hora R.
convert_datetime_class	POSIXct POSIXlt	Opciones para especificar a qué formato se convierten las variables con los formatos de fecha o de fecha y hora.
convert_missing	*flag*	Opción para convertir los valores que faltan al valor R NA.

Propiedades de applyregressionnode

Los nodos de modelado Regresión lineal pueden utilizarse para generar un nugget de modelo Regresión lineal. El nombre de script de este nugget de modelo es *applyregressionnode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de regressionnode" en la página 211.

Propiedades de applyselflearningnode

Los nodos de modelado de modelo de respuesta de autoaprendizaje (SLRM) pueden utilizarse para generar un nugget de modelo SLRM. El nombre de script de este nugget de modelo es *applyselflearningnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de slrmnode" en la página 214.

Tabla 160. Propiedades de applyselflearningnode.

Propiedades de applyselflearningnode	Valores	Descripción de la propiedad
max_predictions	*number*	
randomization	*number*	
scoring_random_seed	*number*	
sort	ascending descending	Especifica si se mostrarán primero las ofertas con las puntuaciones más altas o más bajas.
model_reliability	*flag*	Tiene en cuenta la opción de fiabilidad del modelo de la pestaña Configuración.

Propiedades de applysequencenode

Los nodos de modelado Secuencia pueden utilizarse para generar un nugget de modelo Secuencia. El nombre de script de este nugget de modelo es *applysequencenode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de sequencenode" en la página 213.

Propiedades de applysvmnode

Los nodos de modelado SVM pueden utilizarse para generar un nugget de modelo SVM. El nombre de script de este nugget de modelo es *applysvmnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de svmnode" en la página 219.

Tabla 161. Propiedades de applysvmnode.

Propiedades de applysvmnode	Valores	Descripción de la propiedad
all_probabilities	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

Propiedades de applystpnode

El nodo de modelado STP puede utilizarse para generar un nugget de modelo asociado, que muestra la salida del modelo en el Visor de salida. El nombre de script de este nugget de modelo es *applystpnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de stpnode" en la página 215.

Tabla 162. Propiedades de applystpnode

Propiedades de applystpnode	Tipo de datos	Descripción de la propiedad
uncertainty_factor	*Boolean*	Mínimo 0, máximo 100.

Propiedades de applytcmnode

Los nodos de modelado temporal causal (TCM) pueden usarse para generar un nugget de modelo TCM. El nombre de script de este nugget de modelo es *applytcmnode*. Para obtener más información sobre los scripts del propio nodo de modelado, consulte "Propiedades de tcmnode" en la página 220.

Tabla 163. Propiedades de applytcmnode

Propiedades de applytcmnode	Valores	Descripción de la propiedad
ext_future	*booleano*	
ext_future_num	*entero*	
noise_res	*booleano*	
conf_limits	*booleano*	
target_fields	*lista*	
target_series	*lista*	

Propiedades de applytimeseriesnode

Los nodos de modelado Serie temporal pueden utilizarse para generar un nugget de modelo Serie temporal. El nombre de script de este nugget de modelo es *applytimeseriesnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de timeseriesnode" en la página 223.

Tabla 164. Propiedades de applytimeseriesnode.

Propiedades de applytimeseriesnode	Valores	Descripción de la propiedad
calculate_conf	*flag*	
calculate_residuals	*flag*	

Propiedades de applytreeasnode

Los nodos de modelado Tree-AS pueden utilizarse para generar un nugget de modelo Tree-AS. El nombre de script de este nugget de modelo es *applytreenode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de treeasnode" en la página 226.

Tabla 165. Propiedades de applytreeasnode

Propiedades de applytreeasnode	Valores	Descripción de la propiedad
calculate_conf	*distintivo*	Esta propiedad incluye cálculos de confianza en el árbol generado.
display_rule_id	*distintivo*	Añade un campo en el resultado de puntuación que indica el ID para el nodo terminal al que se asigna cada registro.
sql_generate	udf native	Se usa para definir las opciones de generación SQL durante la ejecución de secuencia. Elija entre retrotraer a la base de datos y puntuar usando un adaptador de puntuaciones de SPSS Modeler Server (si está conectado a una base de datos con un adaptador de puntuaciones instalado) o puntuar dentro de SPSS Modeler.

Propiedades de applytwostepnode

Los nodos de modelado Bietápico pueden utilizarse para generar un nugget de modelo Bietápico. El nombre de script de este nugget de modelo es *applytwostepnode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de twostepnode" en la página 228.

Propiedades de applytwostepAS

Los nodos de modelado Bietápico AS pueden utilizarse para generar un nugget de modelo Bietápico AS. El nombre de script de este nugget de modelo es *applytwostepAS*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de twostepAS" en la página 229.

Capítulo 15. Propiedades de nodos de modelado de bases de datos

IBM SPSS Modeler admite la integración con herramientas de modelado y minería de datos ofrecidas por proveedores de bases de datos, incluidos Microsoft SQL Server Analysis Services, Oracle Data Mining, IBM DB2 InfoSphere Warehouse, y IBM Netezza Analytics. Podrá crear y almacenar modelos mediante algoritmos nativos de bases de datos, todo desde la aplicación IBM SPSS Modeler. Los modelos de base de datos también se pueden crear y manipular a través de scripts utilizando las propiedades descritas en esta sección.

Por ejemplo, el siguiente fragmento de script muestra la creación de un modelo de árboles de decisión de Microsoft mediante la interfaz de scripts de IBM SPSS Modeler:

```
ruta = 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([])
```

Propiedades de nodos de modelado de Microsoft

Propiedades de nodos de modelado de Microsoft

Propiedades comunes

Las siguientes propiedades son comunes a los nodos de modelado de bases de datos de Microsoft.

Tabla 166. Propiedades comunes de nodos de Microsoft

Propiedades comunes de nodo de Microsoft	Valores	Descripción de la propiedad
`analysis_database_name`	*string*	Nombre de la base de datos de Analysis Services.
`analysis_server_name`	*string*	Nombre del host de Analysis Services.
`use_transactional_data`	*marca*	Especifica si los datos de entrada están en formato tabular o transaccional.
`inputs`	*lista*	Campos de entrada de datos tabulares.

Tabla 166. Propiedades comunes de nodos de Microsoft (continuación)

Propiedades comunes de nodo de Microsoft	Valores	Descripción de la propiedad
target	*campo*	Campo predicho (no aplicable a nodo Clúster de MS o nodos de Agrupación en clústeres de secuencias).
unique_field	*campo*	Campos clave.
msas_parameters	*structured*	Parámetros del algoritmo. Consulte el tema "Parámetros del algoritmo" en la página 245 para obtener más información.
with_drillthrough	*marca*	Opción Con exploración.

Árbol de decisión de MS

No hay propiedades específicas para los nodos del tipo mstreenode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Clúster de MS

No hay propiedades específicas para los nodos del tipo msclusternode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Reglas de asociación de MS

Las siguientes propiedades específicas están disponibles para los nodos del tipo msassocnode:

Tabla 167. Propiedades de msassocnode

Propiedades de msassocnode	Valores	Descripción de la propiedad
id_field	*campo*	Identifica todas las transacciones en los datos.
trans_inputs	*lista*	Los campos de entrada de datos transaccionales.
transactional_target	*campo*	Campo predicho (datos transaccionales).

Bayesiano ingenuo de MS

No hay propiedades específicas para los nodos del tipo msbayesnode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Regresión lineal de MS

No hay propiedades específicas para los nodos del tipo msregressionnode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Red neuronal de MS

No hay propiedades específicas para los nodos del tipo msneuralnetworknode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Regresión logística de MS

No hay propiedades específicas para los nodos del tipo mslogisticnode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Series temporales de MS

No hay propiedades específicas para los nodos del tipo mstimeseriesnode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Clúster de secuencias de MS

Las siguientes propiedades específicas están disponibles para los nodos del tipo mssequenceclusternode:

Tabla 168. Propiedades de mssequenceclusternode

Propiedades de mssequenceclusternode	Valores	Descripción de la propiedad
id_field	*campo*	Identifica todas las transacciones en los datos.
input_fields	*lista*	Los campos de entrada de datos transaccionales.
sequence_field	*campo*	Identificador de secuencia.
target_field	*campo*	Campo predicho (datos tabulares).

Parámetros del algoritmo

Cada tipo de modelo de base de datos de Microsoft tiene parámetros concretos que se pueden establecer mediante la propiedad msas_parameters. Por ejemplo:

```
ruta = modeler.script.stream()
msregressionnode = stream.findByType("msregression", None)
msregressionnode.setPropertyValue("msas_parameters", [["MAXIMUM_INPUT_ATTRIBUTES", 255],
["MAXIMUM_OUTPUT_ATTRIBUTES", 255]])
```

Estos parámetros se derivan de SQL Server. Para ver los parámetros relevantes para cada nodo:

1. Coloque un nodo de origen de base de datos en el lienzo.
2. Abra el nodo de origen de base de datos.
3. Seleccione un origen válido en la lista desplegable **Origen de datos**.
4. Seleccione una tabla válida en la lista **Nombre de tabla**.
5. Pulse en **Aceptar** para cerrar el nodo de origen de base de datos.
6. Conecte un nodo de modelado de bases de datos de Microsoft cuyas propiedades desee conocer.
7. Abra el nodo de modelado de bases de datos.
8. Seleccione la pestaña **Experto**.

Aparecerán las propiedades msas_parameters disponibles de este nodo.

Propiedades de nugget de modelo de Microsoft

Las siguientes propiedades son para los nugget de modelo creados mediante los nodos de modelado de bases de datos de Microsoft.

Árbol de decisión de MS

Tabla 169. Propiedades de Árbol de decisión de MS.

Propiedades de applymstreenode	Valores	Descripción
analysis_database_name	*cadena*	Este nodo se puede puntuar directamente en una ruta. Esta propiedad se utiliza para identificar el nombre de la base de datos de Analysis Services.
analysis_server_name	*cadena*	Nombre del host del servidor de análisis.
datasource	*cadena*	Nombre del origen de datos (DSN) ODBC de SQL Server.
sql_generate	*flag*	Activa la generación de SQL.

Regresión lineal de MS

Tabla 170. Propiedades de Regresión lineal de MS.

Propiedades de applymsregressionnode	**Valores**	**Descripción**
analysis_database_name	*cadena*	Este nodo se puede puntuar directamente en una ruta. Esta propiedad se utiliza para identificar el nombre de la base de datos de Analysis Services.
analysis_server_name	*cadena*	Nombre del host del servidor de análisis.

Red neuronal de MS

Tabla 171. Propiedades de Red neuronal de MS.

Propiedades de applymsneuralnetworknode	**Valores**	**Descripción**
analysis_database_name	*cadena*	Este nodo se puede puntuar directamente en una ruta. Esta propiedad se utiliza para identificar el nombre de la base de datos de Analysis Services.
analysis_server_name	*cadena*	Nombre del host del servidor de análisis.

Regresión logística de MS

Tabla 172. Propiedades de Regresión logística de MS.

Propiedades de applymslogisticnode	**Valores**	**Descripción**
analysis_database_name	*cadena*	Este nodo se puede puntuar directamente en una ruta. Esta propiedad se utiliza para identificar el nombre de la base de datos de Analysis Services.
analysis_server_name	*cadena*	Nombre del host del servidor de análisis.

Series temporales de MS

Tabla 173. Propiedades de MS Time Series.

Propiedades de applymstimeseriesnode	**Valores**	**Descripción**
analysis_database_name	*cadena*	Este nodo se puede puntuar directamente en una ruta. Esta propiedad se utiliza para identificar el nombre de la base de datos de Analysis Services.
analysis_server_name	*cadena*	Nombre del host del servidor de análisis.
start_from	new_prediction historical_ prediction	Especifica si se realizarán predicciones futuras o históricas.
new_step	*number*	Define el período de tiempo inicial de predicciones futuras.
historical_step	*number*	Define el período de tiempo inicial de predicciones históricas.

Tabla 173. Propiedades de MS Time Series (continuación).

Propiedades de applymstimeseriesnode	**Valores**	**Descripción**
end_step	*number*	Define el período de tiempo final de las predicciones.

Clúster de secuencias de MS

Tabla 174. Propiedades de Agrupación en clústeres de secuencias de MS.

Propiedades de applymssequenceclusternode	**Valores**	**Descripción**
analysis_database_name	*cadena*	Este nodo se puede puntuar directamente en una ruta. Esta propiedad se utiliza para identificar el nombre de la base de datos de Analysis Services.
analysis_server_name	*cadena*	Nombre del host del servidor de análisis.

Propiedades de nodos de modelado de Oracle

Propiedades de nodos de modelado de Oracle

Las siguientes propiedades son comunes a los nodos de modelado de bases de datos de Oracle.

Tabla 175. Propiedades comunes de nodos de Oracle.

Propiedades comunes de nodos de Oracle	**Valores**	**Descripción de la propiedad**
objetivo	*campo*	
inputs	*Lista de campos*	
partición	*campo*	Campo usado para dividir los datos en muestras independientes para las fases de entrenamiento, comprobación y validación en la generación del modelo.
datasource		
nombre de usuario		
contraseña		
epassword		
use_model_name	*tag*	
model_name	*cadena*	Nombre personalizado para nuevo modelo.
use_partitioned_data	*tag*	Si se ha definido un campo de partición, esta opción garantiza que sólo se utilizarán los datos de la partición de entrenamiento para la generación del modelo.
unique_field	*campo*	
auto_data_prep	*tag*	Activa o desactiva la característica de preparación de datos automática de Oracle (solamente para bases de datos 11g).
costs	*estructurado*	Propiedad estructurada de la siguiente forma: `[[drugA drugB 1.5] [drugA drugC 2.1]]`, donde los argumentos en [] son costes pronosticados reales.

Tabla 175. Propiedades comunes de nodos de Oracle (continuación).

Propiedades comunes de nodos de Oracle	Valores	Descripción de la propiedad
mode	Simple Expert	Hace que se ignoren ciertas propiedades si se establece como Simple, como se indica en las propiedades de nodos individuales.
use_prediction_probability	*tag*	
prediction_probability	*cadena*	
use_prediction_set	*tag*	

Bayesiano ingenuo de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo oranbnode.

Tabla 176. Propiedades de oranbnode.

Propiedades de oranbnode	Valores	Descripción de la propiedad
singleton_threshold	*number*	0.0–1.0.*
pairwise_threshold	*number*	0.0–1.0.*
priors	Datos Equal Custom	
custom_priors	*estructurado*	Propiedad estructurada de la siguiente forma: set :oranbnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* Propiedad ignorada si Modo se establece como Simple.

Bayesiano adaptativo de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo oraabnnode.

Tabla 177. Propiedades de oraabnnode.

Propiedades de oraabnnode	Valores	Descripción de la propiedad
model_type	SingleFeature MultiFeature NaiveBayes	
use_execution_time_limit	*tag*	*
execution_time_limit	*entero*	El valor debe ser mayor que 0.*
max_naive_bayes_predictors	*entero*	El valor debe ser mayor que 0.*
max_predictors	*entero*	El valor debe ser mayor que 0.*
priors	Datos Equal Custom	
custom_priors	*estructurado*	Propiedad estructurada de la siguiente forma: set :oraabnnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* Propiedad ignorada si Modo se establece como Simple.

Máquinas de vectores de soporte de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo `orasvmnode`.

Tabla 178. Propiedades de orasvmnode.

Propiedades de orasvmnode	Valores	Descripción de la propiedad
active_learning	Enable Disable	
kernel_function	Lineal Gaussian Sistema	
normalization_method	zscore minmax ninguno	
kernel_cache_size	*entero*	Solamente kernel gaussiano. El valor debe ser mayor que 0.*
convergence_tolerance	*number*	El valor debe ser mayor que 0.*
use_standard_deviation	*tag*	Solamente kernel gaussiano.*
standard_deviation	*number*	El valor debe ser mayor que 0.*
use_epsilon	*tag*	Solamente modelos de regresión.*
epsilon	*number*	El valor debe ser mayor que 0.*
use_complexity_factor	*tag*	*
complexity_factor	*number*	*
use_outlier_rate	*tag*	Solamente variantes de una clase.*
outlier_rate	*number*	Solamente variantes de una clase. 0.0–1.0.*
weights	Datos Equal Custom	
custom_weights	*estructurado*	Propiedad estructurada de la siguiente forma: `set :orasvmnode.custom_weights = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]`

* Propiedad ignorada si `Modo` se establece como `Simple`.

Modelos lineales generalizados de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo `oraglmnode`.

Tabla 179. Propiedades de oraglmnode.

Propiedades de oraglmnode	Valores	Descripción de la propiedad
normalization_method	zscore minmax ninguno	
missing_value_handling	ReplaceWithMean UseCompleteRecords	
use_row_weights	*tag*	*
row_weights_field	*campo*	*

Tabla 179. Propiedades de oraglmnode (continuación).

Propiedades de oraglmnode	Valores	Descripción de la propiedad
save_row_diagnostics	*tag*	*
row_diagnostics_table	*cadena*	*
coefficient_confidence	*number*	*
use_reference_category	*tag*	*
reference_category	*cadena*	*
ridge_regression	Auto Off On	*
parameter_value	*number*	*
vif_for_ridge	*tag*	*

* Propiedad ignorada si Modo se establece como Simple.

Árbol de decisión de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo oradecisiontreenode.

Tabla 180. Propiedades de oradecisiontreenode.

Propiedades de oradecisiontreenode	Valores	Descripción de la propiedad
use_costs	*tag*	
impurity_metric	Entropy Gini	
term_max_depth	*entero*	2–20.*
term_minpct_node	*number*	0.0–10.0.*
term_minpct_split	*number*	0.0–20.0.*
term_minrec_node	*entero*	El valor debe ser mayor que 0.*
term_minrec_split	*entero*	El valor debe ser mayor que 0.*
display_rule_ids	*tag*	*

* Propiedad ignorada si Modo se establece como Simple.

O-clúster de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo oraoclusternode.

Tabla 181. Propiedades de oraoclusternode.

Propiedades de oraoclusternode	Valores	Descripción de la propiedad
max_num_clusters	*entero*	El valor debe ser mayor que 0.
max_buffer	*entero*	El valor debe ser mayor que 0.*
sensitivity	*number*	0.0–1.0.*

* Propiedad ignorada si Modo se establece como Simple.

K-medias de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo orakmeansnode.

Tabla 182. Propiedades de orakmeansnode.

Propiedades de orakmeansnode	Valores	Descripción de la propiedad
num_clusters	*entero*	El valor debe ser mayor que 0.
normalization_method	zscore minmax ninguno	
distance_function	Euclidean Cosine	
iteraciones	*entero*	0–20.*
conv_tolerance	*number*	0.0–0.5.*
split_criterion	Variance Size	El valor predeterminado es Variance.*
num_bins	*entero*	El valor debe ser mayor que 0.*
block_growth	*entero*	1–5.*
min_pct_attr_support	*number*	0.0–1.0.*

* Propiedad ignorada si Modo se establece como Simple.

NMF de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo oranmfnode.

Tabla 183. Propiedades de oranmfnode.

Propiedades de oranmfnode	Valores	Descripción de la propiedad
normalization_method	minmax ninguno	
use_num_features	*tag*	*
num_features	*entero*	0–1. El algoritmo estima el valor predeterminado a partir de los datos.*
random_seed	*number*	*
num_iterations	*entero*	0–500.*
conv_tolerance	*number*	0.0–0.5.*
display_all_features	*tag*	*

* Propiedad ignorada si Modo se establece como Simple.

Apriori de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo oraapriorinode.

Tabla 184. Propiedades de oraapriorinode.

Propiedades de oraapriorinode	Valores	Descripción de la propiedad
content_field	*campo*	
id_field	*campo*	

Tabla 184. Propiedades de oraapriorinode (continuación).

Propiedades de oraapriorinode	Valores	Descripción de la propiedad
max_rule_length	*entero*	2–20.
min_confidence	*number*	0.0–1.0.
min_support	*number*	0.0–1.0.
use_transactional_data	*tag*	

Longitud mínima de la descripción de Oracle (LMD)

No hay propiedades específicas para los nodos del tipo oramdlnode. Consulte las propiedades comunes de Oracle que se indican al comienzo de esta sección.

Importancia del atributo de Oracle (AI)

Las siguientes propiedades están disponibles para los nodos del tipo oraainode.

Tabla 185. Propiedades de oraainode.

Propiedades de oraainode	Valores	Descripción de la propiedad
custom_fields	*tag*	Si es verdadero, le permite especificar el objetivo, la entrada y otros campos del nodo actual. Si es falso, se utiliza la configuración actual de un nodo Tipo situado en un punto anterior de la ruta.
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*tag*	Si selection_mode se establece en ImportanceLevel, determina si se seleccionan los campos importantes.
important_label	*cadena*	Especifica la etiqueta para la clasificación como "important".
select_marginal	*tag*	Si selection_mode se establece en ImportanceLevel, determina si se seleccionan los campos marginales.
marginal_label	*cadena*	Especifica la etiqueta para la clasificación como "marginal".
important_above	*number*	0.0–1.0.
select_unimportant	*tag*	Si selection_mode se establece en ImportanceLevel, determina si se seleccionan los campos sin importancia.
unimportant_label	*cadena*	Especifica la etiqueta para la clasificación como "unimportant".
unimportant_below	*number*	0.0–1.0.
importance_value	*number*	Si selection_mode se establece en ImportanceValue, determina el valor de corte que se va a usar. Acepta valores de 0 a 100.
top_n	*number*	Si selection_mode se establece en TopN, determina el valor de corte que se va a usar. Acepta valores de 0 a 1000.

Propiedades de nugget de modelo de Oracle

Las siguientes propiedades son para los nugget de modelo creados mediante los modelos de Oracle.

Bayesiano ingenuo de Oracle

No hay propiedades específicas para los nodos del tipo `applyoranbnode`.

Bayesiano adaptativo de Oracle

No hay propiedades específicas para los nodos del tipo `applyoranbnode`.

Máquinas de vectores de soporte de Oracle

No hay propiedades específicas para los nodos del tipo `applyorasvmnode`.

Árbol de decisión de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo `applyoradecisiontreenode`.

Tabla 186. Propiedades de applyoradecisiontreenode

Propiedades de applyoradecisiontreenode	Valores	Descripción de la propiedad
use_costs	*marca*	
display_rule_ids	*marca*	

O-clúster de Oracle

No hay propiedades específicas para los nodos del tipo `applyoraoclusternode`.

K-medias de Oracle

No hay propiedades específicas para los nodos del tipo `applyorakmeansnode`.

NMF de Oracle

La siguiente propiedad está disponible para los nodos del tipo `applyoranmfnode`:

Tabla 187. Propiedades de applyoranmfnode

Propiedades de applyoranmfnode	Valores	Descripción de la propiedad
display_all_features	*marca*	

Apriori de Oracle

Este nugget de modelo no se puede aplicar en los scripts.

LMD de Oracle

Este nugget de modelo no se puede aplicar en los scripts.

Propiedades de nodos para modelado de IBM DB2

Propiedades de nodos de modelado de IBM DB2

Las siguientes propiedades son comunes a los nodos de modelado de bases de datos de IBM InfoSphere Warehouse (ISW).

Tabla 188. Propiedades comunes de nodo de ISW.

Propiedades comunes de nodo de ISW	**Valores**	**Descripción de la propiedad**
inputs	*Lista de campos*	
datasource		
nombre de usuario		
contraseña		
epassword		
enable_power_options	*flag*	
power_options_max_memory	*entero*	El valor debe ser mayor que 32.
power_options_cmdline	*cadena*	
mining_data_custom_sql	*cadena*	
logical_data_custom_sql	*cadena*	
mining_settings_custom_sql		

Árbol de decisión de ISW

Las siguientes propiedades están disponibles para los nodos del tipo db2imtreenode.

Tabla 189. Propiedades de db2imtreenode.

Propiedades de db2imtreenode	**Valores**	**Descripción de la propiedad**
objetivo	*campo*	
perform_test_run	*flag*	
use_max_tree_depth	*flag*	
max_tree_depth	*entero*	Valor mayor que 0.
use_maximum_purity	*flag*	
maximum_purity	*number*	Número entre 0 y 100.
use_minimum_internal_cases	*flag*	
minimum_internal_cases	*entero*	Valor mayor que 1.
use_costs	*flag*	
costs	*estructurado*	Propiedad estructurada de la siguiente forma: [[drugA drugB 1.5] [drugA drugC 2.1]], donde los argumentos en [] son costes pronosticados reales.

Asociación de ISW

Las siguientes propiedades están disponibles para los nodos del tipo db2imassocnode.

Tabla 190. Propiedades de db2imassocnode.

Propiedades de db2imassocnode	Valores	Descripción de la propiedad
use_transactional_data	*flag*	
id_field	*campo*	
content_field	*campo*	
data_table_layout	básico limited_length	
max_rule_size	*entero*	El valor debe ser mayor que 2.
min_rule_support	*number*	0–100%
min_rule_confidence	*number*	0–100%
use_item_constraints	*flag*	
item_constraints_type	Incluir Exclude	
use_taxonomy	*flag*	
taxonomy_table_name	*cadena*	Nombre de la tabla de DB2 para almacenar los detalles de taxonomía.
taxonomy_child_column_name	*cadena*	Nombre de la columna hija de la tabla de taxonomía. La columna hija contiene los nombres de los elementos o de las categorías.
taxonomy_parent_column_name	*cadena*	Nombre de la columna padre en la tabla de taxonomía. La columna padre contiene los nombres de las categorías.
load_taxonomy_to_table	*flag*	Comprueba si la información sobre taxonomía almacenada en IBM SPSS Modeler se debe cargar en la tabla de taxonomía en el momento de la generación del modelo. Tenga en cuenta que si la tabla de taxonomía ya existe, se eliminará. La información de taxonomía se almacena con el nodo de generación de modelos y se puede editar mediante los botones **Editar categorías** y **Editar taxonomía**.

Secuencia de ISW

Las siguientes propiedades están disponibles para los nodos del tipo db2imsequencenode.

Tabla 191. Propiedades de db2imsequencenode.

Propiedades de db2imsequencenode	Valores	Descripción de la propiedad
id_field	*campo*	
group_field	*campo*	
content_field	*campo*	
max_rule_size	*entero*	El valor debe ser mayor que 2.
min_rule_support	*number*	0–100%
min_rule_confidence	*number*	0–100%
use_item_constraints	*flag*	
item_constraints_type	Incluir Exclude	
use_taxonomy	*flag*	

Tabla 191. Propiedades de db2imsequencenode (continuación).

Propiedades de db2imsequencenode	Valores	Descripción de la propiedad
taxonomy_table_name	*cadena*	Nombre de la tabla de DB2 para almacenar los detalles de taxonomía.
taxonomy_child_column_name	*cadena*	Nombre de la columna hija de la tabla de taxonomía. La columna hija contiene los nombres de los elementos o de las categorías.
taxonomy_parent_column_name	*cadena*	Nombre de la columna padre en la tabla de taxonomía. La columna padre contiene los nombres de las categorías.
load_taxonomy_to_table	*flag*	Comprueba si la información sobre taxonomía almacenada en IBM SPSS Modeler se debe cargar en la tabla de taxonomía en el momento de la generación del modelo. Tenga en cuenta que si la tabla de taxonomía ya existe, se eliminará. La información de taxonomía se almacena con el nodo de generación de modelos y se puede editar mediante los botones **Editar categorías** y **Editar taxonomía**.

Regresión de ISW

Las siguientes propiedades están disponibles para los nodos del tipo db2imregnode.

Tabla 192. Propiedades de db2imregnode.

Propiedades de db2imregnode	Valores	Descripción de la propiedad
objetivo	*campo*	
regression_method	transform lineal polynomial rbf	Consulte en la tabla siguiente propiedades que sólo se aplican si regression_method se establece en rbf.
perform_test_run	*campo*	
limit_rsquared_value	*flag*	
max_rsquared_value	*number*	Valor entre 0,0 y 1,0.
use_execution_time_limit	*flag*	
execution_time_limit_mins	*entero*	Valor mayor que 0.
use_max_degree_polynomial	*flag*	
max_degree_polynomial	*entero*	
use_intercept	*flag*	
use_auto_feature_selection_method	*flag*	
auto_feature_selection_method	normal adjusted	
use_min_significance_level	*flag*	
min_significance_level	*number*	
use_min_significance_level	*flag*	

Las siguientes propiedades solo se aplican si regression_method se establece en rbf.

Tabla 193. Propiedades de db2imregnode si regression_method se establece en rbf.

Propiedades de db2imregnode	Valores	Descripción de la propiedad

Tabla 193. Propiedades de db2imregnode si regression_method se establece en rbf (continuación).

use_output_sample_size	*flag*	Si es verdadero, el valor se ajusta automáticamente como el valor predeterminado.
output_sample_size	*entero*	El valor predeterminado es 2. El mínimo es 1.
use_input_sample_size	*flag*	Si es verdadero, el valor se ajusta automáticamente como el valor predeterminado.
input_sample_size	*entero*	El valor predeterminado es 2. El mínimo es 1.
use_max_num_centers	*flag*	Si es verdadero, el valor se ajusta automáticamente como el valor predeterminado.
max_num_centers	*entero*	El valor predeterminado es 20. El mínimo es 1.
use_min_region_size	*flag*	Si es verdadero, el valor se ajusta automáticamente como el valor predeterminado.
min_region_size	*entero*	El valor por omisión es 15. El mínimo es 1.
use_max_data_passes	*flag*	Si es verdadero, el valor se ajusta automáticamente como el valor predeterminado.
max_data_passes	*entero*	El valor por omisión es de 5. El valor mínimo es 2.
use_min_data_passes	*flag*	Si es verdadero, el valor se ajusta automáticamente como el valor predeterminado.
min_data_passes	*entero*	El valor por omisión es de 5. El valor mínimo es 2.

Agrupación en clústeres de ISW

Las siguientes propiedades están disponibles para los nodos del tipo db2imclusternode.

Tabla 194. Propiedades de db2imclusternode.

Propiedades de db2imclusternode	**Valores**	**Descripción de la propiedad**
cluster_method	demográfico kohonen birch	
kohonen_num_rows	*entero*	
kohonen_num_columns	*entero*	
kohonen_passes	*entero*	
use_num_passes_limit	*flag*	
use_num_clusters_limit	*flag*	
max_num_clusters	*entero*	Valor mayor que 1.
birch_dist_measure	log_likelihood euclidean	El valor predeterminado es log_likelihood.

Tabla 194. Propiedades de db2imclusternode (continuación).

Propiedades de db2imclusternode	Valores	Descripción de la propiedad
birch_num_cfleaves	*entero*	El valor predeterminado es 1000.
birch_num_refine_passes	*entero*	El valor predeterminado es 3; el mínimo es 1.
use_execution_time_limit	*flag*	
execution_time_limit_mins	*entero*	Valor mayor que 0.
min_data_percentage	*number*	0–100%
use_similarity_threshold	*flag*	
similarity_threshold	*number*	Valor entre 0,0 y 1,0.

Bayesiano ingenuo ISW

Las siguientes propiedades están disponibles para los nodos del tipo db2imnbsnode.

Tabla 195. Propiedades de db2imnbnode.

Propiedades de db2imnbnode	Valores	Descripción de la propiedad
perform_test_run	*flag*	
probability_threshold	*number*	El valor predeterminado es 0.001. El valor mínimo es 0; el valor máximo es 1.000.
use_costs	*flag*	
costs	*estructurado*	Propiedad estructurada de la siguiente forma: [[drugA drugB 1.5] [drugA drugC 2.1]], donde los argumentos en [] son costes pronosticados reales.

Regresión logística ISW

Las siguientes propiedades están disponibles para los nodos del tipo db2imlognode.

Tabla 196. Propiedades de db2imlognode.

Propiedades de db2imlognode	Valores	Descripción de la propiedad
perform_test_run	*flag*	
use_costs	*flag*	
costs	*estructurado*	Propiedad estructurada de la siguiente forma: [[drugA drugB 1.5] [drugA drugC 2.1]], donde los argumentos en [] son costes pronosticados reales.

Serie temporal ISW

Note: el parámetro de campos de entrada no se utiliza para este nodo. Si se encuentra el parámetro de campos de entrada en el script, se muestra una advertencia para indicar que el nodo tiene *tiempo* y *objetivos* como campos entrantes, pero no de entrada.

Las siguientes propiedades están disponibles para los nodos del tipo db2imtimeseriesnode.

Tabla 197. Propiedades de db2imtimeseriesnode.

Propiedades de db2imtimeseriesnode	Valores	Descripción de la propiedad
time	*campo*	Se permite entero, hora o fecha.

Tabla 197. Propiedades de db2imtimeseriesnode (continuación).

Propiedades de db2imtimeseriesnode	Valores	Descripción de la propiedad
targets	*lista de campos*	
forecasting_algorithm	arima exponential_ suavizar seasonal_trend_ decomposition	
forecasting_end_time	auto entero date (fecha) time	
use_records_all	*booleano*	Si es falso, deben establecerse use_records_start y use_records_end.
use_records_start	*entero / hora / fecha*	Depende del tipo de campo de tiempo.
use_records_end	*entero / hora / fecha*	Depende del tipo de campo de tiempo.
interpolation_method	ninguno lineal exponential_splines cubic_splines	

Propiedades de nugget de modelo de IBM DB2

Las siguientes propiedades son para los nugget de modelo creados mediante los modelos de IBM DB2 ISW.

Árbol de decisión de ISW

No hay propiedades específicas para los nodos del tipo applydb2imtreenode.

Asociación de ISW

Este nugget de modelo no se puede aplicar en los scripts.

Secuencia de ISW

Este nugget de modelo no se puede aplicar en los scripts.

Regresión de ISW

No hay propiedades específicas para los nodos del tipo applydb2imregnode.

Agrupación en clústeres de ISW

No hay propiedades específicas para los nodos del tipo applydb2imclusternode.

Bayesiano ingenuo ISW

No hay propiedades específicas para los nodos del tipo applydb2imnbnode.

Regresión logística ISW

No hay propiedades específicas para los nodos del tipo applydb2imlognode.

Serie temporal ISW

Este nugget de modelo no se puede aplicar en los scripts.

Propiedades de nodos de modelado de IBM Netezza Analytics

Propiedades de nodos de modelado de Netezza

Las siguientes propiedades son comunes a los nodos de modelado de bases de datos de IBM Netezza.

Tabla 198. Propiedades comunes de nodos de Netezza.

Propiedades comunes de nodos de Netezza	Valores	Descripción de la propiedad
custom_fields	*tag*	Si es verdadero, le permite especificar el objetivo, la entrada y otros campos del nodo actual. Si es falso, se utiliza la configuración actual de un nodo Tipo situado en un punto anterior de la ruta.
inputs	*[campo1 ... campoN]*	Campos de entrada o predictor utilizados por el modelo.
objetivo	*campo*	Campo de destino (continuo o categórico).
record_id	*campo*	El campo que se debe utilizar como identificador de registros exclusivo.
use_upstream_connection	*tag*	Si es verdadero (valor predeterminado), los detalles de conexión especificados en un nodo anterior. No se utiliza si se especifica move_data_to_connection.
move_data_connection	*tag*	Si es verdadero, transfiere los datos a la base de datos especificada mediante connection. No se utiliza si se especifica use_data_upstream_connection.
connection	*estructurado*	La cadena de conexión para la base de datos de Netezza donde se almacena el modelo. Propiedad estructurada de la siguiente forma: ['odbc' '<dsn>' '<nombreusuario>' '<psw>' '<nombrecat>' '<atribs_conex>' [true|false]] donde: <dsn> es el nombre del origen de datos <username> y <psw> son el nombre de usuario y la contraseña para la base de datos <catname> es el nombre de catálogo <conn_attribs> son los atributos de conexión true | false indica si la contraseña es necesaria.
table_name	*cadena*	Nombre de la base de datos donde se debe almacenar el modelo.
use_model_name	*tag*	Si es true, utiliza el nombre que especifica mediante model_name como el nombre del modelo, de lo contrario el sistema crea el nombre del modelo.
model_name	*cadena*	Nombre personalizado para nuevo modelo.
include_input_fields	*tag*	Si es verdadero, transmite todos los campos de entrada siguientes, de lo contrario solo transmite record_id y los campos generados por el modelo.

Árbol de decisión de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzadectreenode.

Tabla 199. Propiedades de netezzadectreenode.

Propiedades de netezzadectreenode	Valores	Descripción de la propiedad
impurity_measure	Entropy Gini	La medición de impureza, utilizada para valorar la mejor ubicación para dividir el árbol.
max_tree_depth	*entero*	Número máximo de niveles que puede alcanzar el crecimiento de árbol. El valor predeterminado es 62 (el máximo posible).
min_improvement_splits	*número*	Mejoras mínimas que se pueden realizar en la impureza de la división. El valor predeterminado es 0.01.
min_instances_split	*entero*	Número mínimo de registros por dividir antes de realizar la división. El valor predeterminado es 2 (el mínimo posible).
weights	*estructurado*	Ponderaciones relativas para clases. Propiedad estructurada de la siguiente forma: set :netezza_dectree.weights = [[drugA 0.3][drugB 0.6]] El valor predeterminado es la ponderación de 1 para todas las clases.
pruning_measure	Acc wAcc	El valor predeterminado es Acc (precisión). wAcc alternativo (precisión ponderada) tiene en cuenta las ponderaciones de clase mientras se aplica la poda.
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	El valor predeterminado es utilizar allTrainingData para calcular la precisión del modelo. Utilice partitionTrainingData para especificar un porcentaje de datos de prueba por utilizar, o useOtherTable para utilizar un conjunto de datos de prueba desde una tabla específica de la base de datos.
perc_training_data	*número*	Si prune_tree_options se establece en partitionTrainingData, especifica el porcentaje de datos que se utilizará para entrenamiento.
prune_seed	*entero*	Semilla aleatoria que se debe utilizar para replicar los resultados del análisis si prune_tree_options se establece en partitionTrainingData; el valor predeterminado es 1.
pruning_table	*cadena*	Nombre de tabla de un conjunto de datos de poda separado para estimar la precisión del modelo.

Tabla 199. Propiedades de netezzadectreenode (continuación).

Propiedades de netezzadectreenode	Valores	Descripción de la propiedad
compute_probabilities	*tag*	Si es verdadero, produce un campo de nivel de confianza (probabilidad) y el campo de predicción.

K-medias de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzakmeansnode.

Tabla 200. Propiedades de netezzakmeansnode.

Propiedades de netezzakmeansnode	Valores	Descripción de la propiedad
distance_measure	Euclidean Manhattan Canberra máximo	Método que se debe utilizar para medir la distancia entre puntos de datos.
num_clusters	*entero*	Número de clústeres que se deben crear; el valor predeterminado es 3.
max_iterations	*entero*	Número de iteraciones de algoritmos después de los cuáles se debe detener la prueba del modelo; el valor predeterminado es 5.
rand_seed	*entero*	Semilla aleatoria que se debe utilizar para replicar los resultados del análisis; el valor predeterminado es 12345.

Red bayesiana de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzabayesnode.

Tabla 201. Propiedades de netezzabayesnode.

Propiedades de netezzabayesnode	Valores	Descripción de la propiedad
base_index	*entero*	Identificador numérico asignado al primer campo de entrada de gestión interna; el valor predeterminado es 777.
sample_size	*entero*	Tamaño de la muestra que se tomará si el número de atributos es muy grande; el valor predeterminado es 10.000.
display_additional_information	*tag*	Si es verdadero, muestra información adicional sobre el progreso en un cuadro de diálogo de mensaje.
type_of_prediction	best vecinos nn-neighbors	Tipo de algoritmo de predicción que se utilizará: el mejor (vecino más correlacionado), vecinos (predicción ponderada de vecinos), o vecinos NN (vecinos no nulos).

Bayesiano ingenuo de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzanaivebayesnode.

Tabla 202. Propiedades de netezzanaivebayesnode.

Propiedades de **netezzanaivebayesnode**	Valores	Descripción de la propiedad
compute_probabilities	*tag*	Si es verdadero, produce un campo de nivel de confianza (probabilidad) y el campo de predicción.
use_m_estimation	*tag*	Si es verdadero, utiliza la técnica m-estimation para evitar probabilidades de cero durante el cálculo.

KNN de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo `netezzaknnnode`.

Tabla 203. Propiedades de netezzaknnnode.

Propiedades de **netezzaknnnode**	Valores	Descripción de la propiedad
weights	*estructurado*	Propiedad estructurada que se utiliza para asignar ponderaciones a clases individuales. Ejemplo: `set :netezzaknnnode.weights = [[drugA 0.3][drugB 0.6]]`
distance_measure	Euclidean Manhattan Canberra Máximo	Método que se utiliza para medir la distancia entre puntos de datos.
num_nearest_neighbors	*entero*	Número de vecinos más próximos de un caso concreto; el valor predeterminado es 3.
standardize_measurements	*tag*	Si es verdadero, estandariza las mediciones de campos de entrada continuos antes de calcular los valores de distancia.
use_coresets	*tag*	Si es verdadero, utiliza el muestreo del conjunto principal para acelerar el cálculo de conjuntos de datos grandes.

Clúster divisivo de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo `netezzadivclusternode`.

Tabla 204. Propiedades de netezzadivclusternode.

Propiedades de **netezzadivclusternode**	Valores	Descripción de la propiedad
distance_measure	Euclidean Manhattan Canberra Máximo	Método que se utiliza para medir la distancia entre puntos de datos.
max_iterations	*entero*	Número máximo de iteraciones de algoritmo que se ejecutarán antes de detener el entrenamiento del modelo; el valor predeterminado es 5.
max_tree_depth	*entero*	El número máximo de niveles en los que se puede subdividir el conjunto de datos; el valor predeterminado es 3.
rand_seed	*entero*	Semilla aleatoria, se utiliza para replicar los análisis; el valor predeterminado es 12345.
min_instances_split	*entero*	El número mínimo de registros que se pueden dividir, el valor predeterminado es 5.

Tabla 204. Propiedades de netezzadivclusternode (continuación).

Propiedades de netezzadivclusternode	Valores	Descripción de la propiedad
nivel	*entero*	El nivel de jerarquía en el que se guardan los registros; el valor predeterminado es -1.

PCA de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzapcanode.

Tabla 205. Propiedades de netezzapcanode.

Propiedades de netezzapcanode	Valores	Descripción de la propiedad
center_data	*tag*	Si es verdadero (opción predeterminada), ejecuta el centrado de datos (también conocido como "sustracción de media") antes del análisis.
perform_data_scaling	*tag*	Si es verdadero, ejecuta la adaptación de los datos antes del análisis. De esta forma el análisis será menos arbitrario si las diferentes variables se miden en unidades diferentes.
force_eigensolve	*tag*	Si es verdadero, utiliza un método menos preciso, pero más rápido para encontrar componentes principales.
pc_number	*entero*	El número principal de componentes al que se reducirá el conjunto de datos; el valor predeterminado es 1.

Árbol de regresión de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzaregtreenode.

Tabla 206. Propiedades de netezzaregtreenode.

Propiedades de netezzaregtreenode	Valores	Descripción de la propiedad
max_tree_depth	*entero*	Número máximo de niveles que puede crecer el árbol por debajo del nodo raíz; el valor predeterminado es 10.
split_evaluation_measure	Variance	Medida de clase de impureza, se utiliza para evaluar la mejor ubicación para dividir el árbol; el valor predeterminado (y la única opción actualmente) es Variance.
min_improvement_splits	*número*	Cantidad mínima para reducir la impureza antes de que se cree la nueva división en el árbol.
min_instances_split	*entero*	El número mínimo de registros que se pueden dividir.
pruning_measure	mse r2 pearson spearman	Método que se utilizará para la poda.

Tabla 206. Propiedades de netezzaregtreenode (continuación).

Propiedades de **netezzaregtreenode**	Valores	Descripción de la propiedad
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	El valor predeterminado es utilizar allTrainingData para calcular la precisión del modelo. Utilice partitionTrainingData para especificar un porcentaje de datos de prueba por utilizar, o useOtherTable para utilizar un conjunto de datos de prueba desde una tabla específica de la base de datos.
perc_training_data	*número*	Si prune_tree_options se establece en PercTrainingData, especifica el porcentaje de datos que se utilizará para entrenamiento.
prune_seed	*entero*	Semilla aleatoria que se debe utilizar para replicar los resultados del análisis si prune_tree_options se establece en PercTrainingData; el valor predeterminado es 1.
pruning_table	*cadena*	Nombre de tabla de un conjunto de datos de poda separado para estimar la precisión del modelo.
compute_probabilities	*tag*	Si es verdadero, especifica que las varianzas de las clases asignadas se deben incluir en el resultado.

Regresión lineal de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzalineregressionnode.

Tabla 207. Propiedades de netezzalineregressionnode.

Propiedades de **netezzalineregressionnode**	Valores	Descripción de la propiedad
use_svd	*tag*	Si es verdadero, utiliza la matriz de descomposición de valores singulares en lugar de la matriz original, para mayor velocidad y precisión numérica.
include_intercept	*tag*	Si es verdadero (valor predeterminado), aumenta la precisión global de la solución.
calculate_model_diagnostics	*tag*	Si es verdadero, calcula el diagnóstico del modelo.

Series temporales de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzatimeseriesnode.

Tabla 208. Propiedades de netezzatimeseriesnode.

Propiedades de **netezzatimeseriesnode**	Valores	Descripción de la propiedad
time_points	*campo*	El campo de entrada que contiene los valores de fecha u hora de la serie temporal.

Tabla 208. Propiedades de netezzatimeseriesnode (continuación).

Propiedades de netezzatimeseriesnode	**Valores**	**Descripción de la propiedad**
time_series_ids	*campo*	El campo de entrada que contiene diversos ID de series temporales; utilice esta opción si la entrada contiene más de una serie temporal.
model_table	*campo*	Nombre de la tabla de base de datos en la que se guardará el modelo de series temporales de Netezza.
description_table	*campo*	Nombre de la tabla de entrada que contiene los nombres y las descripciones de las series temporales.
seasonal_adjustment_table	*campo*	Nombre de la tabla de salida en la que se guardarán los valores ajustados calculados por los algoritmos de suavizado exponencial o de descomposición de tendencia estacional.
algorithm_name	SpectralAnalysis o spectral ExponentialSmoothing o esmoothing ARIMA SeasonalTrendDecomposition o std	Algoritmo que hay que utilizar para el modelado de series temporales.
trend_name	N A DA M DM	Tipo de tendencia del suavizado exponencial: N - none A - aditivo DA -aditivo amortiguado M - multiplicativo DM - multiplicativo amortiguado
seasonality_type	N A M	Tipo de estacionalidad del suavizado exponencial: N - none A - aditivo M - multiplicativo
interpolation_method	lineal cubicspline exponentialspline	Método de interpolación que hay que utilizar.
timerange_setting	SD SP	Valor de rango de tiempo que se debe utilizar: SD - determinado por el sistema (utiliza el rango completo de datos de series de tiempo) SP - especificado por el usuario mediante earliest_time y latest_time

Tabla 208. Propiedades de netezzatimeseriesnode (continuación).

Propiedades de **netezzatimeseriesnode**	Valores	Descripción de la propiedad
earliest_time latest_time	*entero* *fecha* *hora* *marca de tiempo*	Valores de inicio y finalización si timerange_setting es SP. El formato debe seguir el valor time_points. Por ejemplo, si el campo time_points contiene una fecha, éste también debería ser una fecha. Ejemplo: set NZ_DT1.timerange_setting = 'SP' set NZ_DT1.earliest_time = '1921-01-01' set NZ_DT1.latest_time = '2121-01-01'
arima_setting	SD SP	Valor para el algoritmo ARIMA (sólo se utiliza si algorithm_name se establece en ARIMA): SD - determinado por el sistema SP - especificado por el usuario Si se utiliza arima_setting = SP, utilice los parámetros siguientes para establecer los valores estacionales y no estacionales. Ejemplo (solo no estacionales): 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 d_symbol q_symbol sp_symbol sd_symbol sq_symbol	sin eq lesseq	ARIMA - operador para los parámetros p, d, q, sp, sd y sq: less - menor que eq - igual a lesseq - menor o igual que
p	*entero*	ARIMA: grados no estacionales de autocorrelación.
q	*entero*	ARIMA: valor de derivación no estacional.
d	*entero*	ARIMA: número no estacional de órdenes de media móvil presentes en el modelo.

Tabla 208. Propiedades de netezzatimeseriesnode (continuación).

Propiedades de netezzatimeseriesnode	**Valores**	**Descripción de la propiedad**
sp	*entero*	ARIMA: grados estacionales de autocorrelación.
sq	*entero*	ARIMA: valor de derivación estacional.
sd	*entero*	ARIMA: número estacional de órdenes de media móvil presentes en el modelo.
advanced_setting	SD SP	Determina cómo se manejan los valores avanzados: SD - determinado por el sistema SP - especificado por el usuario mediante period, units_period y forecast_setting. Ejemplo: set NZ_DT1.advanced_setting = 'SP' set NZ_DT1.period = 5 set NZ_DT1.units_period = 'd'
punto	*entero*	Longitud de ciclo estacional, especificado junto con units_period. No aplicable para análisis espectrales.
units_period	ms s min h d semana q y	Unidades en que se expresa period: ms - milisegundos s - segundos min - minutos h - horas d - días wk - semanas q - trimestres y - años Por ejemplo, para una serie temporal semanal utilice 1 para period y wk para units_period.
forecast_setting	forecasthorizon forecasttimes	Especifica cómo se han de realizar las previsiones.
forecast_horizon	*entero* *fecha* *hora* *timestamp*	Si forecast_setting = forecasthorizon, especifica el valor de punto final para la previsión. El formato debe seguir el valor time_points. Por ejemplo, si el campo time_points contiene una fecha, éste también debería ser una fecha.

Tabla 208. Propiedades de netezzatimeseriesnode (continuación).

Propiedades de **netezzatimeseriesnode**	Valores	Descripción de la propiedad
forecast_times	*entero* *fecha* *hora* *marca de tiempo*	Si forecast_setting = forecasttimes, especifica los valores que se utilizarán para hacer previsiones. El formato debe seguir el valor time_points. Por ejemplo, si el campo time_points contiene una fecha, éste también debería ser una fecha.
include_history	*tag*	Indica si se deben incluir los valores históricos en los resultados.
include_interpolated_values	*tag*	Indica si se deben incluir los valores interpolados en los resultados. No es aplicable si include_history es false.

Lineal generalizado de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzaglmnode.

Tabla 209. Propiedades de netezzaglmnode.

Propiedades de **netezzaglmnode**	Valores	Descripción de la propiedad
dist_family	bernoulli de gauss poisson negativebinomial wald gamma	El tipo de distribución. El valor predeterminado es bernoulli.
dist_params	*número*	Valor del parámetro de distribución que hay que utilizar. Sólo se aplica si distribution es Negativebinomial.
trials	*entero*	Sólo se aplica si distribution es Binomial. Cuando la respuesta objetivo es un número de eventos que tienen lugar en un conjunto de ensayos, el campo target contiene el número de eventos, y el campo trials contiene el número de ensayos.
model_table	*campo*	Nombre de la tabla de base de datos en la que se guardará el modelo lineal generalizado de Netezza.
maxit	*entero*	Número máximo de iteraciones que debe ejecutar el algoritmo; el valor predeterminado es 20.
eps	*número*	Valor del error máximo (en notación científica) en el que el algoritmo debería dejar de buscar el modelo de mejor ajuste. El valor predeterminado es -3, lo que significa 1E-3 o 0,001.

Tabla 209. Propiedades de netezzaglmnode (continuación).

Propiedades de netezzaglmnode	Valores	Descripción de la propiedad
tol	*número*	El valor (en notación científica) por debajo del que los errores se tratan como si su valor fuera cero. El valor predeterminado es -7, lo que significa que los valores de error por debajo de 1E-7 (o 0,0000001) se cuentan como insignificantes.
link_func	identidad inverse invnegative invsquare sqrt power oddspower anotaciones clog loglog cloglog logit probit gaussit cauchit canbinom cangeom cannegbinom	Función de enlace que se ha de utilizar; el valor predeterminado es logit.
link_params	*número*	Valor del parámetro de función de enlace que hay que utilizar. Sólo se aplica si link_function es power u oddspower.
interaction	[[[*nombrescol1*],[*niveles1*]], [[*nombrescol2*],[*niveles2*]], ...,[[*nombrescolN*],[*nivelesN*]],]	Especifica las interacciones entre los campos. *colnames* es una lista de campos de entrada, y *level* es siempre 0 para cada campo. Ejemplo: [[["K","BP","Sex","K"],[0,0,0,0]], [["Age","Na"],[0,0]]]
intercept	*tag*	Si es true, incluye la interceptación en el modelo.

Propiedades de nugget de modelo de Netezza

Las siguientes propiedades son comunes a los nuggets del modelo de la base de datos de Netezza.

Tabla 210. Propiedades comunes de nugget de nodos de Netezza

Propiedades comunes de nugget de modelo de Netezza	Valores	Descripción de la propiedad
connection	*string*	La cadena de conexión para la base de datos de Netezza donde se almacena el modelo.
table_name	*string*	Nombre de la base de datos donde se almacenará el modelo.

Otras propiedades del nugget de modelo son las mismas que las del nodo de modelado correspondiente.

Los nombres de script de los nuggets de modelo son los siguientes.

Tabla 211. Nombres de script de nuggets de modelos Netezza

Nugget de modelo	**Nombre de script**
Árbol de decisiones	`applynetezzadectreenode`
K-medias	`applynetezzakmeansnode`
Red bayesiana	`applynetezzabayesnode`
bayesiano ingenuo	`applynetezzanaivebayesnode`
KNN	`applynetezzaknnnode`
Clúster divisivo	`applynetezzadivclusternode`
PCA	`applynetezzapcanode`
Árbol de regresión	`applynetezzaregtreenode`
Regresión lineal	`applynetezzalineregressionnode`
Serie temporal	`applynetezzatimeseriesnode`
Lineal generalizado	`applynetezzaglmnode`

Capítulo 16. Propiedades de los nodos de resultados

Las propiedades de nodos de resultados se diferencian un poco de las de otros tipos de nodos. En lugar de hacer referencia a una opción determinada de nodo, las propiedades de nodos de resultados almacenan una referencia en el objeto de resultado. Esto resulta útil al tomar un valor de una tabla y establecerlo como un parámetro de ruta.

Esta sección describe las propiedades de scripts disponibles para los nodos de resultados.

Propiedades de analysisnode

El nodo Análisis evalúa la capacidad de los modelos predictivos para generar predicciones precisas. Los nodos Análisis realizan varias comparaciones entre los valores predichos y los valores reales para uno o más nugget de modelo. También pueden comparar modelos predictivos entre sí.

Ejemplo

```
node = stream.create("analysis", "My node")
# Pestaña Análisis
node.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)
# "Definir medida del usuario..."
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"])
# Pestaña "Resultados"
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/analysis_out.html")
```

Tabla 212. Propiedades de analysisnode.

Propiedad de analysisnode	Tipo de datos	Descripción de la propiedad
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
use_output_name	*tag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*cadena*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Se utiliza para especificar el tipo de resultado.
by_fields	*lista*	

Tabla 212. Propiedades de analysisnode (continuación).

Propiedad de analysisnode	Tipo de datos	Descripción de la propiedad
full_filename	*cadena*	Nombre del archivo de resultados, si se trata de resultados HTML, de datos o de disco.
coincidence	*tag*	
performance	*tag*	
evaluation_binary	*flag*	
confidence	*tag*	
umbral	*number*	
improve_accuracy	*number*	
inc_user_measure	*tag*	
user_if	*expr*	
user_then	*expr*	
user_else	*expr*	
user_compute	[Mean Sum Min Max SDev]	

Propiedades de dataauditnode

El nodo Auditoría de datos permite echar un primer vistazo exhaustivo a los datos, incluyendo estadísticos de resumen, histogramas y distribución para cada campo, así como información sobre valores atípicos, valores perdidos y extremos. Los resultados se muestran en una matriz fácil de leer que se puede ordenar y utilizar para generar nodos de preparación de datos y gráficos de tamaño completo.

Ejemplo

```
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")
```

Tabla 213. Propiedades de dataauditnode.

Propiedad de dataauditnode	Tipo de datos	Descripción de la propiedad
custom_fields	*tag*	
campos	*[campo1 ... campoN]*	
overlay	*campo*	

Tabla 213. Propiedades de dataauditnode (continuación).

Propiedad de **dataauditnode**	Tipo de datos	Descripción de la propiedad
display_graphs	*flag*	Se utiliza para activar o desactivar la representación de gráficos en la matriz de resultados.
basic_stats	*tag*	
advanced_stats	*tag*	
median_stats	*tag*	
calculate	Recuento Breakdown	Se utiliza para calcular valores perdidos. Seleccione uno de los métodos de cálculo, ambos o ninguno.
outlier_detection_method	std iqr	Se utiliza para especificar el método de detección de valores atípicos y extremos.
outlier_detection_std_outlier	*número*	Si outlier_detection_method es std, especifica el número que se utilizará para definir los valores atípicos.
outlier_detection_std_extreme	*número*	Si outlier_detection_method es std, especifica el número que se utilizará para definir los valores extremos.
outlier_detection_iqr_outlier	*número*	Si outlier_detection_method es iqr, especifica el número que se utilizará para definir los valores atípicos.
outlier_detection_iqr_extreme	*número*	Si outlier_detection_method es iqr, especifica el número que se utilizará para definir los valores extremos.
use_output_name	*tag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*cadena*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Se utiliza para especificar el tipo de resultado.
paginate_output	*tag*	Si output_format es HTML, los resultados se separarán por páginas.
lines_per_page	*número*	Si se usa con paginate_output, especifica las líneas por página del resultado.
full_filename	*cadena*	

Propiedades de matrixnode

El nodo Matriz crea una tabla que muestra las relaciones entre campos. Se suele utilizar normalmente para mostrar las relaciones entre dos campos simbólicos, pero también puede mostrar relaciones entre campos de marcas o entre campos numéricos.

Ejemplo

```
node = stream.create("matrix", "My node")
# Pestaña "Configuración"
node.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")
# Pestaña "Aspecto"
node.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)
# Pestaña "Resultados"
node.setPropertyValue("full_filename", "C:/output/matrix_output.html")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("paginate_output", True)
node.setPropertyValue("lines_per_page", 50)
```

Tabla 214. Propiedades de matrixnode.

Propiedades de **matrixnode**	Tipo de datos	Descripción de la propiedad
fields	Seleccionado Flags Numerics	
row	*campo*	
column	*campo*	
include_missing_values	*tag*	Determina si los valores perdidos por el sistema (nulos) o no especificados por el usuario (vacíos) se incluyen en los resultados de fila y columna.
cell_contents	CrossTabs Función	
function_field	*cadena*	
function	Sum Media Mín Máx SDev	
sort_mode	Unsorted Ascending Descending	
highlight_top	*número*	Si no es cero, es verdadero.
highlight_bottom	*número*	Si no es cero, es verdadero.

Tabla 214. Propiedades de matrixnode (continuación).

Propiedades de matrixnode	Tipo de datos	Descripción de la propiedad
display	[Counts Expected Residuos RowPct ColumnPct TotalPct]	
include_totals	*tag*	
use_output_name	*tag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*cadena*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Se utiliza para especificar el tipo de resultado. Los formatos Formatted y Delimited aceptan el modificador transposed, mediante el cual se transponen filas y columnas en la tabla.
paginate_output	*tag*	Si output_format es HTML, los resultados se separarán por páginas.
lines_per_page	*número*	Si se usa con paginate_output, especifica las líneas por página del resultado.
full_filename	*cadena*	

Propiedades de meansnode

El nodo Medias compara las medias de grupos independientes o de pares de campos relacionados para probar si existen diferencias significativas. Por ejemplo, puede comparar los ingresos medios antes y después de poner en marcha una promoción o comparar los ingresos de los clientes que no recibieron esa promoción con los que sí lo hicieron.

Ejemplo

```
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")
```

Tabla 215. Propiedades de meansnode.

Propiedad de meansnode	Tipo de datos	Descripción de la propiedad
means_mode	BetweenGroups BetweenFields	Especifica el tipo de estadístico de las medias que se va a ejecutar en los datos.
test_fields	[field1 ... fieldn]	Especifica el campo de prueba si means_mode se establece en BetweenGroups.
grouping_field	*campo*	Especifica el campo de agrupación.
paired_fields	[[campo1 campo2] [campo3 campo4] ...]	Especifica los pares de campos que se usan si means_mode se establece en BetweenFields.
label_correlations	*tag*	Determina si las etiquetas de correlación se muestran en el resultado. Esta configuración se aplica únicamente si means_mode se establece en BetweenFields.
correlation_mode	Probability Absolute	Determina si las correlaciones deben etiquetarse según la probabilidad o según el valor absoluto.
weak_label	*cadena*	
medium_label	*cadena*	
strong_label	*cadena*	
weak_below_probability	*número*	Si correlation_mode se establece en Probability, determina el valor de corte para las correlaciones débiles. Debe tratarse de un valor comprendido entre 0 y 1; por ejemplo, 0,90.
strong_above_probability	*número*	Valor de corte para correlaciones fuertes.
weak_below_absolute	*número*	Si correlation_mode se establece en Absolute, especifica el valor de corte para las correlaciones débiles. Debe tratarse de un valor comprendido entre 0 y 1; por ejemplo, 0,90.
strong_above_absolute	*número*	Valor de corte para correlaciones fuertes.
unimportant_label	*cadena*	
marginal_label	*cadena*	
important_label	*cadena*	
unimportant_below	*número*	Valor de corte para una importancia del campo baja. Debe tratarse de un valor comprendido entre 0 y 1; por ejemplo, 0,90.
important_above	*número*	
use_output_name	*tag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*cadena*	Nombre que se va a usar.

Tabla 215. Propiedades de meansnode (continuación).

Propiedad de meansnode	Tipo de datos	Descripción de la propiedad
output_mode	Screen File	Especifica la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Especifica el tipo de resultado.
full_filename	*cadena*	
output_view	Simple Advanced	Determina si el resultado muestra la vista simple o la avanzada.

Propiedades de reportnode

El nodo Informe crea informes con formato que contienen texto fijo, así como datos y otras expresiones derivadas de los datos. Puede especificar el formato del informe utilizando plantillas de texto para definir el texto fijo y las construcciones de resultados de datos. Puede proporcionar formato de texto personalizado utilizando etiquetas HTML de la plantilla y configurando opciones en la pestaña Resultado. Puede incluir valores de datos y otros resultados condicionales mediante el uso de expresiones CLEM en la plantilla.

Ejemplo

```
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)
```

Tabla 216. Propiedades de reportnode.

Propiedad de reportnode	Tipo de datos	Descripción de la propiedad
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	HTML (*.html*) Text (*.txt*) Output (*.cou*)	Se utiliza para especificar el tipo de resultado.
use_output_name	*tag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*cadena*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.
text	*cadena*	
full_filename	*cadena*	
highlights	*tag*	
título	*cadena*	
lines_per_page	*número*	

Propiedades de routputnode

El nodo Routput permite analizar datos y resultados de la puntuación del modelo utilizando su propio script R personalizado. El resultado del análisis puede ser en texto o en gráficos. El resultado se añade a la pestaña **Resultado** del panel de gestor. De forma alternativa, el resultado se puede redirigir a un archivo.

Tabla 217. Propiedades de routputnode

Propiedades de routputnode	Tipo de datos	Descripción de la propiedad
syntax	*cadena*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*flag*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*flag*	
output_name	Auto Custom	
custom_name	*cadena*	
output_to	Screen File	
output_type	Graph Text	
full_filename	*cadena*	
graph_file_type	HTML COU	
text_file_type	HTML TEXTCOU	

Propiedades de setglobalsnode

El nodo Val. globales explora los datos y calcula los valores de resumen que se pueden utilizar en expresiones CLEM. Por ejemplo, puede utilizar este nodo para calcular estadísticos para un campo denominado *edad* y, a continuación, utilizar la media global de *edad* en expresiones CLEM insertando la función `@GLOBAL_MEAN(edad)`.

Ejemplo

```
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)
```

Tabla 218. Propiedades de setglobalsnode.

Propiedad de setglobalsnode	Tipo de datos	Descripción de la propiedad
globals	[Sum Mean Min Max SDev]	Propiedad estructurada en la que los campos que se van a establecer deben hacer referencia a la sintaxis siguiente: node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
clear_first	*tag*	
show_preview	*tag*	

Propiedades de simevalnode

El nodo de evaluación de simulación evalúa un campo de destino predicho y presenta información sobre la distribución y correlación del campo de destino.

Tabla 219. Propiedades de simevalnode.

Propiedades de simevalnode	Tipo de datos	Descripción de la propiedad
target	*campo*	
iteration	*campo*	
presorted_by_iteration	*booleano*	
max_iterations	*número*	
tornado_fields	*[field1...fieldN]*	
plot_pdf	*booleano*	
plot_cdf	*booleano*	
show_ref_mean	*booleano*	
show_ref_median	*booleano*	
show_ref_sigma	*booleano*	
num_ref_sigma	*número*	
show_ref_pct	*booleano*	
ref_pct_bottom	*número*	
ref_pct_top	*número*	
show_ref_custom	*booleano*	
ref_custom_values	*[number1...numberN]*	
category_values	Category Probabilities Both	
category_groups	Categories Iterations	
create_pct_table	*booleano*	
pct_table	Quartiles Intervals Custom	

Tabla 219. Propiedades de simevalnode (continuación).

Propiedades de simevalnode	Tipo de datos	Descripción de la propiedad
pct_intervals_num	*número*	
pct_custom_values	*[number1...numberN]*	

Propiedades de simfitnode

El nodo de simulación de ajuste examina la distribución de las estadísticas de los datos de cada campo y genera (o actualiza) un nodo de generación de simulación, con la mejor distribución de ajuste asignada a cada campo. El nodo de generación de simulación se puede utilizar, a continuación, para generar datos simulados.

Tabla 220. Propiedades de simfitnode.

Propiedades de simfitnode	Tipo de datos	Descripción de la propiedad
build	Nodo XMLExport Both	
use_source_node_name	*booleano*	
source_node_name	*cadena*	El nombre personalizado del nodo de origen que se está generando o actualizando.
use_cases	All LimitFirstN	
use_case_limit	*entero*	
fit_criterion	AndersonDarling KolmogorovSmirnov	
num_bins	*entero*	
parameter_xml_filename	*cadena*	
generate_parameter_import	*booleano*	

Propiedades de statisticsnode

El nodo Estadísticos ofrece información básica de resumen acerca de los campos numéricos. Calcula estadísticos de resumen para campos individuales y correlaciones entre campos.

Ejemplo

```
node = stream.create("statistics", "My node")
# Pestaña "Configuración"
node.setPropertyValue("examine", ["Age", "BP", "Drug"])
node.setPropertyValue("statistics", ["Mean", "Sum", "SDev"])
node.setPropertyValue("correlate", ["BP", "Drug"])
# "Etiquetas de correlación..." 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")
# Pestaña "Resultados"
node.setPropertyValue("full_filename", "c:/output/statistics_output.html")
node.setPropertyValue("output_format", "HTML")
```

Tabla 221. Propiedades de statisticsnode.

Propiedad de **statisticsnode**	Tipo de datos	Descripción de la propiedad
use_output_name	*tag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*cadena*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Se utiliza para especificar el tipo de resultado.
full_filename	*cadena*	
examine	*lista*	
correlate	*lista*	
statistics	[Count Mean Sum Min Max Range Variance SDev SErr Median Mode]	
correlation_mode	Probability Absolute	Determina si las correlaciones deben etiquetarse según la probabilidad o según el valor absoluto.
label_correlations	*tag*	
weak_label	*cadena*	
medium_label	*cadena*	
strong_label	*cadena*	
weak_below_probability	*número*	Si correlation_mode se establece en Probability, determina el valor de corte para las correlaciones débiles. Debe tratarse de un valor comprendido entre 0 y 1; por ejemplo, 0,90.
strong_above_probability	*número*	Valor de corte para correlaciones fuertes.
weak_below_absolute	*número*	Si correlation_mode se establece en Absolute, especifica el valor de corte para las correlaciones débiles. Debe tratarse de un valor comprendido entre 0 y 1; por ejemplo, 0,90.
strong_above_absolute	*número*	Valor de corte para correlaciones fuertes.

Propiedades de statisticsoutputnode

El nodo Resultados de Statistics le permite llamar a un procedimiento de IBM SPSS Statistics para analizar los datos de IBM SPSS Modeler. Se puede acceder a una gran variedad de procedimientos analíticos de IBM SPSS Statistics. Este nodo requiere una copia de IBM SPSS Statistics con licencia.

Las propiedades de este nodo están descritas en "Propiedades de statisticsoutputnode" en la página 302.

Propiedades de tablenode

El nodo Tabla muestra los datos en formato de tabla, que también se puede escribir en un archivo. Esto es útil en cualquier momento en que necesite inspeccionar sus valores de datos o exportarlos en un formato fácilmente legible.

Ejemplo

```
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)
```

Tabla 222. Propiedades de tablenode.

Propiedad de tablenode	Tipo de datos	Descripción de la propiedad
full_filename	*cadena*	Nombre del archivo de resultados, si se trata de resultados HTML, de datos o de disco.
use_output_name	*tag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*cadena*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Se utiliza para especificar el tipo de resultado.
transpose_data	*tag*	Transpone los datos antes de exportarlos de manera que las filas representan campos y las columnas, registros.
paginate_output	*tag*	Si output_format es HTML, los resultados se separarán por páginas.
lines_per_page	*number*	Si se usa con paginate_output, especifica las líneas por página del resultado.
highlight_expr	*cadena*	

Tabla 222. Propiedades de tablenode (continuación).

Propiedad de tablenode	Tipo de datos	Descripción de la propiedad
output	*cadena*	Propiedad de sólo lectura que mantiene una referencia en la última tabla creada por el nodo.
value_labels	*[[Valor CadenaEtiqueta] [Valor CadenaEtiqueta] ...]*	Se utiliza para especificar etiquetas para los pares de valores.
display_places	*entero*	Establece el número de cifras decimales para el campo cuando se muestra (sólo se aplica a campos con almacenamiento REAL). Un valor de -1 utilizará el valor predeterminado de la ruta.
export_places	*entero*	Establece el número de cifras decimales para el campo cuando se exporta (sólo se aplica a campos con almacenamiento REAL). Un valor de -1 utilizará el valor predeterminado de la ruta.
decimal_separator	DEFAULT PERIOD COMMA	Establece el separador decimal para el campo (sólo se aplica a campos con almacenamiento REAL).
date_format	"DDMMAA" "MMDDYY" "AAMMDD" "YYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-AAAA" "DD-MES-YY" "DD-MES-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.AAAA" "MM.DD.YYYY" "DD.MES.YY" "DD.MES.YYYY" "DD/MM/YY" "DD/MM/AAAA" "MM/DD/YY" "MM/DD/YYYY" "DD/MES/YY" "DD/MES/YYYY" MON YYYY q Q YYYY ww WK YYYY	Establece el formato de fecha para el campo (sólo se aplica a campos con almacenamiento FECHA o MARCADETIEMPO).

Tabla 222. Propiedades de tablenode (continuación).

Propiedad de tablenode	Tipo de datos	Descripción de la propiedad
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"	Establece el formato de hora para el campo (sólo se aplica a campos con almacenamiento TIME o TIMESTAMP).
column_width	*entero*	Establece el ancho de columna para el campo. Un valor de -1 establecerá el ancho de columna en Auto.
justify	AUTO CENTER LEFT RIGHT	Establece la justificación de columna para el campo.

Propiedades de transformnode

El nodo Transformación permite seleccionar y previsualizar los resultados de las transformaciones antes de aplicarlas a los campos seleccionados.

Ejemplo

```
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)
```

Tabla 223. Propiedades de transformnode.

Propiedades de transformnode	Tipo de datos	Descripción de la propiedad
campos	[*field1... fieldn*]	Campos que se utilizarán en la transformación.
formula	All Select	Indica si se deben calcular todas las transformaciones o sólo las seleccionadas.
formula_inverse	*flag*	Indica si se debe utilizar la transformación inversa.
formula_inverse_offset	*número*	Indica el desplazamiento de los datos que se utilizará en la fórmula. De forma predeterminadao es 0, a menos que el usuario especifique un valor.

Tabla 223. Propiedades de transformnode (continuación).

Propiedades de transformnode	Tipo de datos	Descripción de la propiedad
formula_log_n	*flag*	Indica si debe utilizarse la transformación $\log_n$.
formula_log_n_offset	*número*	
formula_log_10	*flag*	Indica si debe utilizarse la transformación $\log_{10}$.
formula_log_10_offset	*número*	
formula_exponential	*flag*	Indica si se debe utilizar la transformación exponencial (e^x).
formula_square_root	*flag*	Indica si se debe utilizar la transformación de raíz cuadrada.
use_output_name	*flag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*string*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	HTML (*.html*) Output (*.cou*)	Se utiliza para especificar el tipo de resultado.
paginate_output	*tag*	Si output_format es HTML, los resultados se separarán por páginas.
lines_per_page	*número*	Si se usa con paginate_output, especifica las líneas por página del resultado.
full_filename	*string*	Indica el nombre de archivo que se utilizará para el resultado de archivo.

Capítulo 17. Propiedades de nodos Exportar

Propiedades de nodos Exportar comunes

Las siguientes propiedades son comunes a todos los nodos de exportación:

Tabla 224. Propiedades comunes de nodos de exportación

Propiedad	Valores	Descripción de la propiedad
publish_path	*string*	Introduzca el nombre raíz que se utilizará para la imagen publicada y los archivos de parámetros.
publish_metadata	*marca*	Especifica si un archivo de metadatos se crea y que describe las entradas y los resultados de la imagen y sus modelos de datos.
publish_use_parameters	*marca*	Especifica si se incluyen parámetros de ruta en el archivo *.par.
publish_parameters	*lista de cadenas*	Especifica los parámetros que se van a incluir.
execute_mode	export_data publish	Especifica si el nodo se ejecuta sin publicar la ruta, o si la ruta se publica automáticamente cuando se ejecuta el nodo.

Propiedades de asexport

La exportación de Analytic Server permite ejecutar una ruta en el sistema de archivos distribuido de Hadoop (HDFS).

Ejemplo

```
node = stream.create("asexport", "My node")
node.setPropertyValue("data_source", "Drug1n")
node.setPropertyValue("export_mode", "overwrite")
```

Tabla 225. Propiedades de asexport.

Propiedades de asexport	Tipo de datos	Descripción de la propiedad
data_source	*cadena*	Nombre del origen de datos.
export_mode	*string*	Especifica si se **añaden** (append) los datos exportados al origen de datos existente o si se **sobrescriben** (overwrite) al origen de datos existente.

Propiedades del nodo de exportación Cognos

El nodo Exportar de IBM Cognos BI exporta datos en un formato que pueden leer las bases de datos de Cognos BI.

Para este nodo, debe definir una conexión de Cognos y una conexión ODBC.

Conexión de Cognos

Las propiedades de la conexión de Cognos son las siguientes.

Tabla 226. Propiedades del nodo de exportación Cognos

Propiedades del nodo de exportación Cognos	**Tipo de datos**	**Descripción de la propiedad**
cognos_connection	*["cadena","distintivo","cadena","cadena","cadena"]*	Una propiedad de la lista que contiene los detalles de conexión para el servidor de Cognos. El formato es: `["Cognos_server_URL", login_mode, "namespace", "username", "password"]` donde: `Cognos_server_URL` es la URL del servidor de Cognos que contiene el origen. `login_mode` indica si se utiliza el inicio de sesión anónimo, y es `true` o `false`; si se establece en `true`, los campos siguientes deben establecerse en `""`. `namespace` especifica el proveedor de autenticación de seguridad utilizado para registrarse en el servidor. `username` y `password` son los utilizados para registrarse en el servidor de Cognos. En lugar de `login_mode`, también hay disponibles las modalidades siguientes: • `anonymousMode`. Por ejemplo: `['Cognos_server_url', 'anonymousMode', "namespace", "username", "password"]` • `credentialMode`. Por ejemplo: `['Cognos_server_url', 'credentialMode', "namespace", "username", "password"]` • `storedCredentialMode`. Por ejemplo: `['Cognos_server_url', 'storedCredentialMode', "stored_credential_name"]` Donde `stored_credential_name` es el nombre de una credencial de Cognos del repositorio.
cognos_package_name	*cadena*	La ruta y el nombre del paquete de Cognos al que está exportando datos, por ejemplo: `/Public Folders/MyPackage`
cognos_datasource	*cadena*	
cognos_export_mode	`Publicar` `ExportFile`	
cognos_filename	*cadena*	

Conexión ODBC

Las propiedades de la conexión ODBC son idénticas a las indicadas para databaseexportnode en la sección siguiente, a excepción de la propiedad datasource, que no es válida.

Propiedades de databaseexportnode

El nodo Exportar base de datos escribe datos en orígenes de datos relacionales compatibles con ODBC. Para escribir en un origen de datos ODBC, el origen de datos debe existir y debe tener permiso para escribir en él.

Ejemplo

```
'''
Se asume que se ha configurado un origen de datos denominado "Miorigendedatos"
'''
ruta = modeler.script.stream()
db_exportnode = stream.createAt("databaseexport", "DB Export", 200, 200)
applynn = stream.findByType("applyneuralnetwork", None)
stream.link(applynn, db_exportnode)

# pestaña Exportar
db_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)

# Cuadro de diálogo Esquema
db_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")

# Cuadro de diálogo Índices
db_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"]])
```

Tabla 227. Propiedades de databaseexportnode.

Propiedades de databaseexportnode	Tipo de datos	Descripción de la propiedad
datasource	*cadena*	
nombre de usuario	*cadena*	
contraseña	*cadena*	

Tabla 227. Propiedades de databaseexportnode (continuación).

Propiedades de databaseexportnode	Tipo de datos	Descripción de la propiedad
epassword	*cadena*	Este intervalo es de sólo lectura durante la ejecución. Para generar una contraseña codificada, utilice la herramienta Contraseña disponible del menú Herramientas. Consulte el tema "Generación de una contraseña codificada" en la página 51 para obtener más información.
table_name	*cadena*	
write_mode	Create Append Fundir	
map	*cadena*	Correlaciona un nombre de campo de ruta a un nombre de columna de la base de datos (sólo es válido si write_mode es Merge). Para una fusión, todos los campos se deben estar correlacionados para que se exporten. Los nombres de campos que no existen en la base de datos se añaden como nuevas columnas.
key_fields	*lista*	Especifica el campo de ruta que se utiliza para la clave; la propiedad map muestra los elementos que se corresponden con la base de datos.
join	Base de datos Añadir	
drop_existing_table	*tag*	
delete_existing_rows	*tag*	
default_string_size	*entero*	
type		Propiedad estructurada que se utiliza para establecer el tipo de esquema.
generate_import	*tag*	
use_custom_create_table_command	*tag*	Utilice el intervalo *custom_create_table* para modificar el comando de SQL estándar CREATE TABLE.
custom_create_table_command	*cadena*	Especifica el comando de cadena a utilizar en lugar del comando de SQL estándar CREATE TABLE.
use_batch	*tag*	Las siguientes propiedades son opciones avanzadas para la carga masiva de la base de datos. Un valor verdadero para Use_batch desactiva fila a fila las confirmaciones en la base de datos.
batch_size	*number*	Especifica el número de registros para enviar a la base de datos antes de confirmar en la memoria.

Tabla 227. Propiedades de databaseexportnode (continuación).

Propiedades de databaseexportnode	Tipo de datos	Descripción de la propiedad
bulk_loading	Off ODBC External	Especifica el tipo de carga masiva. A continuación se muestran las opciones adicionales para ODBC y External.
not_logged	*flag*	
odbc_binding	Row Columna	Especifique el enlace a lo largo de las filas o de las columnas para la carga masiva a través de ODBC.
loader_delimit_mode	Tabulador Space Other	Especifique el tipo de delimitador para la carga masiva a través de un programa externo. Seleccione Other junto con la propiedad loader_other_delimiter para especificar los delimitadores, como la coma (,).
loader_other_delimiter	*cadena*	
specify_data_file	*tag*	Una marca verdadera activa la siguiente propiedad data_file, en la que puede especificar el nombre de archivo y la ruta de acceso en la que se va a escribir al realizar la carga masiva en la base de datos.
data_file	*cadena*	
specify_loader_program	*tag*	Una marca verdadera activa la siguiente propiedad loader_program, en la que puede especificar el nombre y la ubicación de un programa o script del cargador externo.
loader_program	*cadena*	
gen_logfile	*tag*	Una marca verdadera activa la siguiente propiedad logfile_name, en la que puede especificar el nombre de un archivo en el servidor para generar un registro de errores.
logfile_name	*cadena*	
check_table_size	*tag*	Una marca verdadera permite la comprobación de la tabla para garantizar que el aumento del tamaño de la tabla de la base de datos se corresponde con el número de filas exportadas desde IBM SPSS Modeler.
loader_options	*cadena*	Especifique los argumentos adicionales, como -comment y -specialdir, en el programa cargador.
export_db_primarykey	*tag*	Determina si el campo especificado es una clave primaria.
use_custom_create_index_command	*tag*	Si se establece true, se activa SQL personalizado para todos los índices.

Tabla 227. Propiedades de databaseexportnode (continuación).

Propiedades de databaseexportnode	Tipo de datos	Descripción de la propiedad
custom_create_index_command	*cadena*	Especifica el comando de SQL empleado para crear índices cuando SQL personalizado está activado. (Este valor puede anularse para determinados índices, tal y como se indica a continuación.)
indexes.INDEXNAME.fields		Crea el índice especificado si procede y enumera los nombres de campos que se van a incluir en él.
INDEXNAME "use_custom_create_ ⌂index_command"	*tag*	Se usa para activar o desactivar el SQL personalizado para un índice específico. Consulte los ejemplos a continuación de la tabla siguiente.
INDEXNAME "custom_create_index_ command"	*cadena*	Especifica el SQL personalizado que se usa para el índice específico. Consulte los ejemplos a continuación de la tabla siguiente.
indexes.INDEXNAME.remove	*tag*	Si se establece True, se elimina el índice específico del grupo de índices.
table_space	*cadena*	Especifica el espacio de tabla que se creará.
use_partition	*tag*	Especifica que se utilizará el campo Distribuir Hash.
partition_field	*cadena*	Especifica el contenido del campo Distribuir Hash.

Nota: Para algunas bases de datos, puede especificar que se crearán tablas de bases de datos para la exportación con compresión (por ejemplo, el equivalente a `CREATE TABLE MYTABLE (...) COMPRESS YES;` en SQL). Las propiedades `use_compression` y `compression_mode` se proporcionan para dar soporte a esta característica, como se indica a continuación.

Tabla 228. Propiedades de databaseexportnode utilizando funciones de compresión.

Propiedades de databaseexportnode	Tipo de datos	Descripción de la propiedad
use_compression	*Booleana*	Si se establece en True, crea tablas para la exportación con compresión.
compression_mode	Row Page	Establece el nivel de compresión de las bases de datos de SQL Server.
	Predeterminado Direct_Load_Operations All_Operations Básico OLTP Query_High Query_Low Archive_High Archive_Low	Establece el nivel de compresión de las bases de datos de Oracle. Tenga en cuenta que los valores OLTP, Query_High, Query_Low, Archive_High y Archive_Low requieren un mínimo de Oracle 11gR2.

Ejemplo que muestra cómo cambiar el comando `CREATE INDEX` para un determinado índice:

```
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)>"])
```

De forma alternativa, esto puede hacerse a través de una tabla hash:

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields":["id", "region"],
 "use_custom_create_index_command":True, "custom_create_index_command":"CREATE INDEX <nombre-indice> ON
 <nombre-tabla> <(columnasindex-columns)>"])
```

Propiedades de datacollectionexportnode

El nodo de exportación IBM SPSS Data Collection abre los datos en el formato utilizado por el software de investigación de mercados IBM SPSS Data Collection. Se debe instalar la biblioteca de datos de IBM SPSS Data Collection para utilizar este nodo.

Ejemplo

```
ruta = 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)
```

Tabla 229. Propiedades de datacollectionexportnode

Propiedades de datacollectionexportnode	Tipo de datos	Descripción de la propiedad
metadata_file	*string*	Nombre del archivo de metadatos que se va a exportar.
merge_metadata	Overwrite MergeCurrent	
enable_system_variables	*marca*	Especifica si el archivo *.mdd* exportado debe incluir las variables de sistema de IBM SPSS Data Collection.
casedata_file	*string*	Nombre del archivo *.sav* donde se exportan los datos de casos.
generate_import	*marca*	

Propiedades de excelexportnode

El nodo de exportación a Excel genera datos en el formato de archivo .xlsx de Excel. Si lo desea, puede elegir iniciar automáticamente Excel y abrir el archivo exportado cuando se ejecute el nodo.

Ejemplo

```
ruta = 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)
```

Tabla 230. Propiedades de excelexportnode

Propiedad de **excelexportnode**	Tipo de datos	Descripción de la propiedad
full_filename	*string*	
excel_file_type	Excel2007	
export_mode	Create Append	
inc_field_names	*marca*	Especifica si los nombres de campos deben incluirse en la primera fila de la hoja de trabajo.
start_cell	*string*	Especifica la casilla de inicio de la exportación.
worksheet_name	*string*	Nombre de la hoja de trabajo que se va a escribir.
launch_application	*marca*	Determina si Excel debe invocarse para el archivo resultante. Tenga en cuenta que deberá especificar la ruta para iniciar Excel en el cuadro de diálogo Aplicaciones de ayuda (menú Herramientas, Aplicaciones de ayuda).
generate_import	*marca*	Determina si debe crearse un nodo Importar a Excel que leerá el archivo de datos exportado.

Propiedades de outputfilenode

El nodo Archivo sin formato produce datos en un archivo de texto delimitado. Esto es útil para exportar datos que se pueden leer con otro software de hoja de cálculo o de análisis.

Ejemplo

```
ruta = 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)
```

Tabla 231. Propiedades de outputfilenode

Propiedades de **outputfilenode**	Tipo de datos	Descripción de la propiedad
full_filename	*string*	Nombre del archivo de resultados.

Tabla 231. Propiedades de outputfilenode (continuación)

Propiedades de outputfilenode	Tipo de datos	Descripción de la propiedad
write_mode	Overwrite Append	
inc_field_names	*marca*	
use_newline_after_records	*marca*	
delimit_mode	Comma Tab Space Other	
other_delimiter	*carácter*	
quote_mode	Ninguno Single Double Other	
other_quote	*marca*	
generate_import	*marca*	
codificación	StreamDefault SystemDefault "UTF-8"	

Propiedades de sasexportnode

El nodo Exportar SAS produce datos en formato SAS, para leerlos en SAS o en un paquete de software compatible con SAS. Hay tres formatos de archivo SAS disponibles: SAS para Windows/OS2, SAS para UNIX o SAS versión 7/8.

Ejemplo

```
ruta = 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)
```

Tabla 232. Propiedades de sasexportnode

Propiedades de sasexportnode	Tipo de datos	Descripción de la propiedad
format	Windows UNIX SAS7 SAS8	Campos de etiquetas de propiedad de variantes.
full_filename	*string*	
export_names	NamesAndLabels NamesAsLabels	Se utiliza para correlacionar los nombres de campos de IBM SPSS Modeler que se vayan a exportar a nombres de variables de IBM SPSS Statistics o SAS.
generate_import	*marca*	

Propiedades de statisticsexportnode

El nodo Exportar Statistics ofrece los resultados en formato IBM SPSS Statistics *.sav* o *.zsav*. Los archivos *.sav* o *.zsav* se pueden leer con IBM SPSS Statistics Base y otros productos. Este es también el formato utilizado para los archivos caché de IBM SPSS Modeler.

Las propiedades de este nodo están descritas en "Propiedades de statisticsexportnode" en la página 303.

Propiedades del nodo tm1export

El nodo Exportar de IBM Cognos TM1 exporta datos en un formato que pueden leer las bases de datos de IBM Cognos TM1.

Tabla 233. Propiedades del nodo tm1export.

Propiedades del nodo tm1export	Tipo de datos	Descripción de la propiedad
pm_host	*string*	Nombre del host. Por ejemplo: `TM1_export.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')`
tm1_connection	*["campo","campo", ... ,"campo"]*	Una propiedad de la lista que contiene los detalles de conexión para el servidor de TM1. El formato es: `[ "TM1_Server_Name", "tm1_ username", "tm1_password"]` Por ejemplo: `TM1_export.setPropertyValue("tm1_connection", ['Planning Sample', "admin" "apple"])`
selected_cube	*campo*	El nombre del cubo al que está exportando datos. Por ejemplo: `TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan")`
spssfield_tm1element_mapping	*list*	El elemento de tm1 al que hay que correlacionar debe formar parte de la dimensión de columna para la vista de cubo seleccionada. El formato es: `[["param1", "valor"],...,["paramN","valor"]]` Por ejemplo: `TM1_export.setPropertyValue("spssfield_ tm1element_mapping", [["plan_version","plan_version"], ["plan_department","plan_department"]])`

Propiedades de xmlexportnode

El nodo de exportación XML exporta datos a un archivo en formato XML. También puede crear un nodo de origen XML para leer los datos exportados a la ruta.

Ejemplo

```
ruta = 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"]])
```

Tabla 234. Propiedades de xmlexportnode

Propiedades de xmlexportnode	**Tipo de datos**	**Descripción de la propiedad**
full_filename	*string*	(obligatorio) Ruta completa y nombre de archivo del archivo XML para exportar.
use_xml_schema	*marca*	Especifica si utilizar un esquema XML (archivo XSD o DTD) para controlar la estructura de los datos exportados.
full_schema_filename	*string*	Ruta completa y nombre de archivo del archivo XSD o DTD que se quiere utilizar. Es obligatorio si use_xml_schema está establecido como true.
generate_import	*marca*	Genera un nodo de origen XML que leerá el archivo de datos exportados de nuevo en la ruta.
registros	*string*	Expresión de XPath que denota el límite de registro.
map	*string*	Correlaciona el nombre de campo a la estructura XML.

Capítulo 18. Propiedades de nodos de IBM SPSS Statistics

Propiedades de statisticsimportnode

El nodo Archivo Statistics lee los datos desde un formato de archivo *.sav* o *.zsav* que utiliza IBM SPSS Statistics y archivos caché guardados en IBM SPSS Modeler, que también puede utilizar el mismo formato.

Ejemplo

```
ruta = 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)
```

Tabla 235. Propiedades de statisticsimportnode.

Propiedades de statisticsimportnode	Tipo de datos	Descripción de la propiedad
full_filename	*cadena*	El nombre completo del archivo, incluyendo la ruta.
contraseña	*cadena*	La contraseña. El parámetro password se debe establecer antes que el parámetro file_encrypted.
file_encrypted	*tag*	Indica si el archivo está protegido con contraseña.
import_names	NamesAndLabels LabelsAsNames	Método para gestionar nombres y etiquetas de variables.
import_data	DataAndLabels LabelsAsData	Método para gestionar valores y etiquetas.
use_field_format_for_storage	*Booleana*	Especifica si se utiliza la información de formato de campo de IBM SPSS Statistics al importar.

Propiedades de statisticstransformnode

El nodo Transformación Statistics ejecuta una selección de comandos de sintaxis de IBM SPSS Statistics en los orígenes de datos de IBM SPSS Modeler. Este nodo requiere una copia de IBM SPSS Statistics con licencia.

Ejemplo

```
ruta = 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)
```

Tabla 236. Propiedades de statisticstransformnode

Propiedades de statisticstransformnode	Tipo de datos	Descripción de la propiedad
syntax	*string*	
check_before_saving	*marca*	Valida la sintaxis introducida antes de guardar las entradas. Muestra un mensaje de error si la sintaxis no es válida.
default_include	*marca*	Consulte el tema "Propiedades de filternode" en la página 128 para obtener más información.
include	*marca*	Consulte el tema "Propiedades de filternode" en la página 128 para obtener más información.
new_name	*string*	Consulte el tema "Propiedades de filternode" en la página 128 para obtener más información.

Propiedades de statisticsmodelnode

El nodo Modelo Statistics permite analizar y trabajar con sus datos ejecutando los procedimientos de IBM SPSS Statistics que producen PMML. Este nodo requiere una copia de IBM SPSS Statistics con licencia.

Ejemplo

```
ruta = modeler.script.stream()
statisticsmodelnode = stream.createAt("statisticsmodel", "Model", 200, 200)
statisticsmodelnode.setPropertyValue("syntax", "COMPUTE NewVar = Na + K.")
statisticsmodelnode.setKeyedPropertyValue("new_name", "NewVar", "Mixed Drugs")
```

Propiedades de statisticsmodelnode	Tipo de datos	Descripción de la propiedad
syntax	*string*	
default_include	*marca*	Consulte el tema "Propiedades de filternode" en la página 128 para obtener más información.
include	*marca*	Consulte el tema "Propiedades de filternode" en la página 128 para obtener más información.
new_name	*string*	Consulte el tema "Propiedades de filternode" en la página 128 para obtener más información.

Propiedades de statisticsoutputnode

El nodo Resultados de Statistics le permite llamar a un procedimiento de IBM SPSS Statistics para analizar los datos de IBM SPSS Modeler. Se puede acceder a una gran variedad de procedimientos analíticos de IBM SPSS Statistics. Este nodo requiere una copia de IBM SPSS Statistics con licencia.

Ejemplo

```
ruta = 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")
```

Tabla 237. Propiedades de statisticsoutputnode

Propiedades de **statisticsoutputnode**	Tipo de datos	Descripción de la propiedad
mode	Dialog Syntax	Selecciona la opción "cuadro de diálogo de IBM SPSS Statistics" o el editor Sintaxis
syntax	*string*	
use_output_name	*marca*	
output_name	*string*	
output_mode	Screen File	
full_filename	*string*	
file_type	HTML SPV SPW	

Propiedades de statisticsexportnode

El nodo Exportar Statistics ofrece los resultados en formato IBM SPSS Statistics *.sav* o *.zsav*. Los archivos *.sav* o *.zsav* se pueden leer con IBM SPSS Statistics Base y otros productos. Este es también el formato utilizado para los archivos caché de IBM SPSS Modeler.

Ejemplo

```
ruta = 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)
```

Tabla 238. Propiedades de statisticsexportnode.

Propiedades de **statisticsexportnode**	Tipo de datos	Descripción de la propiedad
full_filename	*cadena*	
file_type	sav zsav	Guardar el archivo en formato *sav* o *zsav*. Por ejemplo: `statisticsexportnode.setPropertyValue("file_type","sav")`
encrypt_file	*distintivo*	Indica si el archivo está protegido con contraseña.
password	*cadena*	La contraseña.
launch_application	*distintivo*	

Tabla 238. Propiedades de statisticsexportnode (continuación).

Propiedades de `statisticsexportnode`	**Tipo de datos**	**Descripción de la propiedad**
`export_names`	`NamesAndLabels` `NamesAsLabels`	Se utiliza para correlacionar los nombres de campos de IBM SPSS Modeler que se vayan a exportar a nombres de variables de IBM SPSS Statistics o SAS.
`generate_import`	*distintivo*	

Capítulo 19. Propiedades de Supernodos

En las siguientes tablas se describen las propiedades específicas de los Supernodos. Tenga en cuenta que las propiedades de nodos comunes se aplican también a los Supernodos.

Tabla 239. Propiedades del supernodo de terminal

Nombre de la propiedad	Tipo de propiedad / Lista de valores	Descripción de la propiedad
execute_method	Script Normal	
script	*string*	

Parámetros de Supernodos

Puede utilizar scripts para crear o establecer parámetros de Supernodo utilizando el formato general:

```
mySuperNode.setParameterValue("minvalue", 30)
```

Puede recuperar el valor del parámetro con:

```
value mySuperNode.getParameterValue("minvalue")
```

Búsqueda de los supernodos existentes

Puede encontrar supernodos en rutas utilizando la función findByType() :

```
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)
```

Configuración de las propiedades de nodos encapsulados

Puede establecer las propiedades de determinados nodos encapsulados dentro un supernodo accediendo al diagrama hijo dentro del Supernodo. Por ejemplo, imaginemos que tiene un Supernodo de origen que incluye un nodo de archivo de variables encapsulado para leer los datos. Puede pasar el nombre del archivo para leer (especificado mediante la propiedad `full_filename`) accediendo al diagrama hijo y buscando el nodo relevante como se indica a continuación:

```
childDiagram = source_supernode.getChildDiagram()
varfilenode = childDiagram.findByType("variablefile", None)
varfilenode.setPropertyValue("full_filename", "c:/mydata.txt")
```

Creación de supernodos

Si desea crear un supernodo y su contenido de cero, puede hacerlo de forma similar a la creación del supernodo, accediendo al diagrama hijo y creando los nodos que desee. También debe asegurarse de que los nodos del diagrama del supernodo estén también vinculados a los nodos de los conectores de entrada y/o de salida. Por ejemplo, si desea crear un proceso Supernodo:

```
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)
```

Apéndice A. Referencia de nombres de nodo

Esta sección ofrece una referencia de todos los nombres de script de los nodos de IBM SPSS Modeler.

Nombres de nugget de modelo

Se puede hacer referencia a los nugget de modelo (también denominados modelos generados) según el tipo, como con los objetos de nodo y de resultado. Las siguientes tablas muestran los nombres de referencia de los objetos del modelo.

Tenga en cuenta que estos nombres se utilizan específicamente para hacer referencia a los nugget de modelo en la paleta Modelos (en la esquina superior derecha de la ventana de IBM SPSS Modeler). Para hacer referencia a los nodos de modelo que se han añadido a una ruta para la puntuación, se utiliza un conjunto diferente de nombres con el prefijo `apply...`. Consulte el tema Propiedades de nodos de nugget de modelo para obtener más información.

Nota: En circunstancias normales, se recomienda hacer referencia a los modelos por nombre *y* tipo para evitar confusiones.

Tabla 240. Nombres de nugget de modelo (paleta de modelado).

Nombre del modelo	**Modelo**
anomalydetection	Anomalía
a priori	A priori
autoclassifier	Clasificado automático
autocluster	Agrupación en clústeres automática
autonumeric	Autonumérico
bayesnet	Red bayesiana
c50	C5.0
carma	Carma
árbol cr	Árbol C&R
chaid	CHAID
coxreg	Regresión de Cox
decisionlist	Lista de decisiones
discriminant	Discriminante
factor	PCA/Factorial
featureselection	Sel. características
genlin	Regresión lineal generalizada
glmm	GLMM
kmeans	K-medias
knn	*k*: vecino más cercano
kohonen	Kohonen
lineal	Lineal
logreg	Regresión logística
neuralnetwork	Red neuronal

Tabla 240. Nombres de nugget de modelo (paleta de modelado) (continuación).

Nombre del modelo	Modelo
quest	QUEST
regresión	Regresión lineal
secuencia	Secuencia
slrm	Modelo de respuesta de autoaprendizaje
statisticsmodel	Modelo de IBM SPSS Statistics
svm	Máquina de vectores de soporte
timeseries	Serie temporal
twostep	Dos fases

Tabla 241. Nombres de nugget de modelo (paleta de modelado de bases de datos).

Nombre del modelo	Modelo
db2imcluster	Clúster de IBM ISW
db2imlog	Regresión logística de IBM ISW
db2imnb	Bayesiano ingenuo de IBM ISW
db2imreg	Regresión de IBM ISW
db2imtree	Árbol de decisión de IBM ISW
msassoc	Reglas de asociación de MS
msbayes	Bayesiano ingenuo de MS
mscluster	Clúster de MS
mslogistic	Regresión logística de MS
msneuralnetwork	Red neuronal de MS
msregression	Regresión lineal de MS
mssequencecluster	Clúster de secuencias de MS
mstimeseries	Series temporales de MS
mstree	Árbol de decisión de MS
netezzabayes	Red bayesiana de Netezza
netezzadectree	Árbol de decisión de Netezza
netezzadivcluster	Clúster divisivo de Netezza
netezzaglm	Lineal generalizado de Netezza
netezzakmeans	K-medias de Netezza
netezzaknn	KNN de Netezza
netezzalineregression	Regresión lineal de Netezza
netezzanaivebayes	Bayesiano ingenuo de Netezza
netezzapca	PCA de Netezza
netezzaregtree	Árbol de regresión de Netezza
netezzatimeseries	Series temporales de Netezza
oraabn	Bayesiano adaptativo de Oracle
oraai	Oracle AI
oradecisiontree	Árbol de decisión de Oracle
oraglm	GLM de Oracle

Tabla 241. Nombres de nugget de modelo (paleta de modelado de bases de datos) (continuación).

Nombre del modelo	Modelo
orakmeans	*K*-medias de Oracle
oranb	Bayesiano ingenuo de Oracle
oranmf	NMF de Oracle
oraocluster	O-clúster de Oracle
orasvm	SVM de Oracle

Evitar nombres duplicados del modelo

Al utilizar los scripts para manipular los modelos generados, debe tener en cuenta que el hecho de permitir nombres de modelo duplicados puede originar referencias ambiguas. Para evitarlo, resulta útil utilizar nombres exclusivos para los modelos generados en los scripts.

Para configurar las opciones de los nombres de modelo duplicados:

1. Seleccione en los menús:

   Herramientas > **Opciones de usuario**
2. Pulse en la pestaña **Notificaciones**.
3. Seleccione **Sustituir modelo anterior** para restringir los nombres duplicados de los modelos generados.

El comportamiento de la ejecución de scripts puede variar entre SPSS Modeler y IBM SPSS Collaboration and Deployment Services cuando haya referencias de modelo ambiguas. El cliente de SPSS Modeler incluye la opción "Reemplazar modelo anterior", que reemplaza automáticamente los modelos que tengan el mismo nombre (por ejemplo, cuando un script se itera a través de un bucle para producir un modelo diferente cada vez). Sin embargo, esta opción no está disponible cuando el mismo script se ejecuta en IBM SPSS Collaboration and Deployment Services. Puede evitar esta situación cambiando el nombre del modelo generado en cada iteración para evitar referencias ambiguas a los modelos o borrando el modelo actual (por ejemplo, añadiendo una instrucción `clear generated palette`) antes del final del bucle.

Nombres de tipo de resultados

La siguiente tabla indica los tipos de objetos de resultados y los nodos que los crean. Para obtener una lista completa de los formatos de exportación disponibles para cada tipo de objeto de salida, consulte la descripción de las propiedades del nodo que crea el tipo de salida, disponible en Propiedades comunes de nodos Gráfico y en Propiedades de los nodos de resultados.

Tabla 242. Tipos de objeto de salida y los nodos que los crean.

Tipo de objeto de resultado	Nodo
analysisoutput	Análisis
collectionoutput	Colección
dataauditoutput	Auditoría de datos
distributionoutput	Distribución
evaluationoutput	Evaluación
histogramoutput	Histograma
matrixoutput	Matriz
meansoutput	Medias
multiplotoutput	G. múltiple

Tabla 242. Tipos de objeto de salida y los nodos que los crean (continuación).

Tipo de objeto de resultado	**Nodo**
plotoutput	Gráfico
qualityoutput	Calidad
reportdocumentoutput	Este tipo de objeto no es de un nodo, es un resultado creado por un informe de proyecto
reportoutput	Informe
statisticsprocedureoutput	Resultado de Estadísticas
statisticsoutput	Estadísticos
tableoutput	Tabla
timeplotoutput	Gráfico de tiempo
weboutput	Malla

Apéndice B. Migración desde scripts de herencia a scripts Python

Visión general de la migración de scripts de herencia

Esta sección proporciona un resumen de las diferencias entre el script de Python y el script de herencia en IBM SPSS Modeler y proporciona información acerca de cómo migrar los scripts de herencia a scripts Python. En esta sección encontrar una lista de los comandos de herencia de SPSS Modeler estándar y los comandos Python equivalentes.

Diferencias generales

Una gran parte del diseño de los scripts de herencia se debe a los scripts de comandos del sistema operativo. Los scripts de herencia están orientados a líneas y, aunque existen algunas estructuras de bloque, por ejemplo `if...then...else...endif` y `for...endfor`, generalmente la indentación no es importante.

En los scripts Python, la indentación es importante y las líneas que pertenecen al mismo bloque lógico se deben indentar en el mismo nivel.

Nota: Debe prestar atención cuando copie y pegue el código Python. En el editor, una línea que se ha indentado utilizando pestañas puede parecer la misma que una línea que se ha indentado utilizando espacios. Sin embargo, el script Python generará un error porque no se considera que la indentación de las líneas sea la misma.

El contexto de los scripts

El contexto de script define el entorno en el que se ejecuta un script como, por ejemplo, la ruta o Supernodo que ejecuta el script. En los scripts heredados el contexto es implícito, lo que significa que, por ejemplo, se asume que toda referencia a un nodo de una ruta está dentro de la ruta que ejecuta el script.

En los scripts Python el contexto de script se proporciona de forma explícita mediante el módulo `modeler.script`. Por ejemplo, un script Python de ruta puede acceder a la ruta que ejecuta el script mediante el código siguiente:

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

A continuación podrán invocarse funciones relacionadas con la ruta a través del objeto devuelto.

Comparativa de comandos y funciones

Los scripts heredados están orientados a comando. Esto significa que cada línea del script suele comenzar con el comando a ejecutar seguido de los parámetros, por ejemplo:

```
connect 'Type':typenode to :filternode
rename :derivenode as "Compute Total"
```

Python utiliza funciones que suelen invocarse a través de un objeto (módulo, clase u objeto) que define la función, por ejemplo:

```
ruta = modeler.script.stream()
typenode = stream.findByType("type", "Type)
filternode = stream.findByType("filter", None)
ruta.link(nodotipo, nodofiltro)
derive.setLabel("Compute Total")
```

Literales y comentarios

Algunos de los literales y comandos de comentarios que normalmente se utilizan en IBM SPSS Modeler tienen sus equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 243. Correlación de scripts de herencia con scripts Python para literales y comentarios.

Scripts de herencia	Scripts Python
Entero, por ejemplo `4`	El mismo
Flotante, por ejemplo, `0,003`	El mismo
Cadenas entre comillas simples, por ejemplo, `'Hola'`	El mismo **Nota:** Los literales de cadena que contengan caracteres que no sean ASCII deberán tener el prefijo `u` para garantizar que se representen en Unicode.
Cadenas entre comillas dobles, por ejemplo, `"Hola de nuevo"`	El mismo **Nota:** Los literales de cadena que contengan caracteres que no sean ASCII deberán tener el prefijo `u` para garantizar que se representen en Unicode.

Cadenas largas, por ejemplo,
`"""Estas es una cadena`
`que abarca varias`
`líneas"""`
	El mismo

Listas, por ejemplo, `[1 2 3]`	`[1, 2, 3]`
Referencia de variable, por ejemplo, `set x = 3`	`x = 3`

Continuación de línea (\), por ejemplo,
`set x = [1 2 \`
`  3 4]`
	`x = [ 1, 2,\`
	`  3, 4]`

Comentario de bloque, por ejemplo,
`/* Éste es un comentario largo`
`a través de una línea. */`
	`""" Este es un comentario largo`
	`a través de una línea. """`

Comentario de línea, por ejemplo, `set x = 3  # make x 3`	`x = 3  # make x 3`
`undef`	`None`
`true`	`True`
`false`	`False`

Operadores

Algunos de los comandos de operadores que normalmente se utilizan en IBM SPSS Modeler tienen sus comandos equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 244. Correlación de scripts de herencia con scripts Python para operadores.

Scripts de herencia	scripts Python

`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`

Tabla 244. Correlación de scripts de herencia con scripts Python para operadores (continuación).

Scripts de herencia	scripts Python
NUM1 / NUM2	NUM1 / NUM2
= ==	==
/= /==	!=
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

Comandos condicionales y de bucle

Algunos comandos condicionales y de bucle utilizados habitualmente en IBM SPSS Modeler tienen sus comandos equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 245. Correspondencia de scripts de herencia con scripts Python en lo referente a comandos condicionales y de bucle.

Scripts de herencia	scripts Python

for VAR from INT1 to INT2
...
endfor
	for VAR in range(INT1, INT2):
	...
	or
	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	Sin equivalente

Tabla 245. Correspondencia de scripts de herencia con scripts Python en lo referente a comandos condicionales y de bucle (continuación).

Scripts de herencia	scripts Python
`var VAR1`	La declaración de variables no es obligatoria

Variables

En los scripts heredados, las variables se declaran antes de ser referenciadas, por ejemplo:

```
var minodo
set minodo = create typenode at 96 96
```

En los scripts Python, las variables se crean la primera vez que se referencian, por ejemplo:

```
minodo = stream.createAt("type", "Type", 96, 96)
```

En los scripts heredados, las referencias a variables deben eliminarse explícitamente mediante el operador ^, por ejemplo:

```
var minodo
set minodo = create typenode at 96 96
set ^minodo.direction."Age" = Input
```

Al igual que en la mayoría de lenguajes de script, esto no es necesario en los scripts Python, por ejemplo:

```
minodo = stream.createAt("type", "Type", 96, 96)
minodo.setKeyedPropertyValue("direction","Age","Input")
```

Tipos modelo, resultado y nodo

En los scripts heredados, a los distintos tipos de objeto (nodo, resultado y modelo) se les suele añadir el tipo al tipo de objeto. Por ejemplo, el nodo Derivar es de tipo `derivenode`:

```
set feature_name_node = create derivenode at 96 96
```

El API de IBM SPSS Modeler en Python no incluye el sufijo `node`, de modo que el nodo Derivar tiene el tipo `derive`, por ejemplo:

```
feature_name_node = stream.createAt("derive", "Feature", 96, 96)
```

La única diferencia en los tipos de nombre entre los scripts Python y los heredados es la ausencia del sufijo de tipo.

Nombres de propiedades

Los nombres de las propiedades son los mismos en scripts heredados y en scripts Python. Por ejemplo, en el nodo Archivo variable, la propiedad que define la ubicación del archivo es `full_filename` en ambos entornos de creación de scripts.

Referencias de nodos

Muchos scripts de herencia utilizan una búsqueda implícita para buscar y acceder al nodo que se ha de modificar. Por ejemplo, los comandos siguientes buscan en la ruta actual un nodo Type con la etiqueta "Type", a continuación, establecen la dirección (o el rol de modelado) del campo "Age" como entrada y el campo "Drug" como destino, esto es, el valor predicho:

```
set 'Type':typenode.direction."Age" = Input
set 'Type':typenode.direction."Drug" = Target
```

En los scripts Python, los objetos de nodo se han de localizar de forma explícita antes de llamar a la función para establecer el valor de propiedad, por ejemplo:

```
typenode = stream.findByType("type", "Type")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
typenode.setKeyedPropertyValue("direction", "Drug", "Target")
```

Nota: En este caso, "Target" debe estar encerrado entre comillas en la cadena.

Los scripts Python pueden utilizar de forma alternativa la enumeración `ModelingRole` del paquete `modeler.api`.

Aunque la versión de los scripts Python puede ser más verbosa, el rendimiento de tiempo de ejecución es mejor ya que la búsqueda del nodo generalmente solo se realiza una vez. En el ejemplo de scripts de herencia, la búsqueda del nodo se realiza para cada comando.

También está soportado buscar nodos por ID (el ID de nodo se puede ver en la pestaña Anotaciones del diálogo del nodo). Por ejemplo, en los scripts de herencia:

```
# id65EMPB9VL87 es el ID de un nodo Type
set @id65EMPB9VL87.direction."Age" = Input
```

El script siguiente muestra el mismo ejemplo en scripts Python:

```
typenode = stream.findByID("id65EMPB9VL87")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
```

Obtener y establecer propiedades

Los scripts de herencia utilizan el comando `set` para asignar un valor. El término que sigue al comando `set` puede ser una definición de propiedad. El script siguiente muestra dos formatos de script posibles para establecer una propiedad:

```
set <referencia de nodo>.<propiedad> = <valor>
set <referencia de nodo>.<propiedad-con claves>.<clave> = <valor>
```

En los scripts Python, se obtiene el mismo resultado utilizando las funciones `setPropertyValue()` y `setKeyedPropertyValue()`, por ejemplo:

```
objeto.setPropertyValue(propiedad, valor)
objeto.setKeyedPropertyValue(propiedad-con claves, clave, valor)
```

En los scripts de herencia, se puede acceder a los valores de las propiedades utilizando el comando `get`, por ejemplo:

```
var n v
set n = get node :filternode
set v = ^n.name
```

En los scripts Python, se obtiene el mismo resultado utilizando la función `getPropertyValue()`, por ejemplo:

```
n = stream.findByType("filter", None)
v = n.getPropertyValue("name")
```

Edición de rutas

En los scripts de herencia, se utiliza el comando `create` para crear un nodo nuevo, por ejemplo:

```
var agg select
set agg = create aggregatenode at 96 96
set select = create selectnode at 164 96
```

En los scripts Python, las rutas tienen varios métodos para crear nodos, por ejemplo:

```
stream = modeler.script.stream()
agg = stream.createAt("aggregate", "Aggregate", 96, 96)
select = stream.createAt("select", "Select", 164, 96)
```

En los scripts de herencia, se utiliza el comando `connect` para crear enlaces entre nodos, por ejemplo:

```
connect ^agg to ^select
```

En los scripts Python, se utiliza el método `link` para crear enlaces entre nodos, por ejemplo:

```
stream.link(agg, select)
```

En los scripts de herencia, se utiliza el comando `disconnect` para eliminar enlaces entre nodos, por ejemplo:

```
disconnect ^agg from ^select
```

En los scripts Python, se utiliza el método `unlink` para eliminar enlaces entre nodos, por ejemplo:

```
stream.unlink(agg, select)
```

En los scripts de herencia, se utiliza el comando `position` para posicionar los nodos en el lienzo de rutas o entre nodos, por ejemplo:

```
position ^agg at 256 256
position ^agg between ^myselect and ^mydistinct
```

En los scripts Python, se obtiene el mismo resultado utilizando dos métodos separados: `setXYPosition` y `setPositionBetween`. Por ejemplo:

```
agg.setXYPosition(256, 256)
agg.setPositionBetween(myselect, mydistinct)
```

Operaciones de nodo

Algunos de los comandos de operaciones de nodo que normalmente se utilizan en IBM SPSS Modeler tienen sus comandos equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 246. Correlación de scripts de herencia con scripts Python para operaciones de nodo.

Scripts de herencia	scripts Python
create *especificaciónodo* at x y	*ruta*.create(*tipo*, *nombre*) *ruta*.createAt(*tipo*, *nombre*, x, y) *ruta*.createBetween(*tipo*, *nombre*, preNode, postNode) *ruta*.createModelApplier(*modelo*, *nombre*)
connect *desdeNodo* to *aNodo*	*ruta*.link(*desdeNodo*, *aNodo*)
delete *nodo*	*ruta*.delete(*nodo*)
disable *nodo*	*ruta*.setEnabled(*nodo*, False)
enable *nodo*	*ruta*.setEnabled(*nodo*, True)
disconnect *desdeNodo* from *aNodo*	*ruta*.unlink(*desdeNodo*, *aNodo*) *ruta*.disconnect(*nodo*)
duplicate *nodo*	*nodo*.duplicate()
execute *nodo*	*ruta*.runSelected(*nodos*, *resultados*) *ruta*.runAll(*resultados*)
flush *nodo*	*nodo*.flushCache()
position *nodo* at x y	*nodo*.setXYPosition(x, y)
position *nodo* between *nodo1* and *nodo2*	*nodo*.setPositionBetween(*nodo1*, *nodo2*)
rename *nodo* as *nombre*	*nodo*.setLabel(*nombre*)

Bucle

En los scripts de herencia, hay dos opciones de bucle principales a las que se da soporte:

- Bucles de *Valor contado*, en los que una variable de índice se mueve entre dos límites de entero.
- Bucles de *secuencia* que avanzan en bucle por una secuencia de valores, enlazando el valor actual con la variable de bucle.

El script siguiente es un ejemplo de un bucle de valor contado en un script de herencia:

```
for i from 1 to 10
 println ^i
endfor
```

El script siguiente es un ejemplo de un bucle de secuencia en un script de herencia:

```
var items
set items = [a b c d]

for i in items
 println ^i
endfor
```

También existen otros tipos de bucles que se pueden utilizar:

- Iteración por los modelos de la paleta de modelos o por los resultados de la paleta de resultados.
- Iteración por los campos de entrada o salida de un nodo.

Los scripts Python también dan soporte a diferentes tipos de bucles. El script siguiente es un ejemplo de un bucle de valor contado en un script Python:

```
i = 1
while i <= 10:
 print i
 i += 1
```

El script siguiente es un ejemplo de un bucle de secuencia en un script Python:

```
items = ["a", "b", "c", "d"]
for i in items:
 print i
```

El bucle de secuencia es muy flexible y cuando se combina con los métodos de la API de IBM SPSS Modeler puede dar soporte a la mayoría de los casos de uso scripts de herencia. El siguiente ejemplo muestra cómo utilizar un bucle de secuencia en scripts Python para iterar por los campos de salida de un nodo:

```
node = modeler.script.stream().findByType("filter", None)
for column in node.getOutputDataModel().columnIterator():
 print column.getColumnName()
```

Ejecución de rutas

Durante la ejecución de la secuencia, el modelo o los objetos de resultados que se generan se añaden a uno de los gestores de objeto. En el script existente, el script debe localizar los objetos creados desde el gestor de objeto, o acceder al resultado generado más recientemente desde el nodo que ha generado el resultado.

La ejecución de rutas en Python es diferente, ya que cualquier objeto de modelo o resultados que genere la ejecución se devuelve una lista que se pasa a la función de ejecución. Esto hace que resulte más sencillo acceder a los resultados de la ejecución de la ruta.

Los scripts de herencia dan soporte a tres comandos de ejecución de ruta:

- `execute_all` ejecuta todos nodos terminales ejecutables en la ruta.
- `execute_script` ejecuta el script de ruta independientemente del valor de la ejecución del script.
- `execute` *`nodo`* ejecuta el nodo especificado.

Los scripts Python dan soporte a un conjunto de funciones similares:

- *`ruta`*`.runAll(`*`lista-resultados`*`)` ejecuta todos los nodos terminales ejecutables de la ruta.
- *`ruta`*`.runScript(`*`lista-resultados`*`)` ejecuta el script de ruta independientemente del valor de la ejecución del script.
- *`ruta`*`.runSelected(`*`matriz-nodos`*`, `*`lista-resultados`*`)` ejecuta el conjunto de nodos especificados en el orden en que se suministran.
- *`nodo`*`.run(`*`lista-resultados`*`)` ejecuta el nodo especificado.

En los scripts de herencia, la ejecución de ruta se puede finalizar con el comando `exit` con un código de entero opcional, por ejemplo:

```
exit 1
```

En los scripts Python, se puede obtener el mismo resultado con el script siguiente:

```
modeler.script.exit(1)
```

Acceso a objetos mediante el sistema de archivos y el repositorio

En los scripts heredados se puede abrir una ruta, un modelo o un resultado existentes mediante el comando `open`, por ejemplo:

```
var s
set s = open stream "c:/my streams/modeling.str"
```

En los scripts Python, existe la clase `TaskRunner`, accesible desde la sesión, que puede utilizarse para realizar tareas similares, por ejemplo:

```
taskrunner = modeler.script.session().getTaskRunner()
s = taskrunner.openStreamFromFile("c:/my streams/modeling.str", True)
```

Para guardar un objeto en los scripts heredados, puede utilizarse el comando `save`, por ejemplo:

```
save stream s as "c:/my streams/new_modeling.str"
```

El enfoque de un script Python consiste en utilizar la clase `TaskRunner`, por ejemplo:

```
taskrunner.saveStreamToFile(s, "c:/my streams/new_modeling.str")
```

Las operaciones basadas en un IBM SPSS Collaboration and Deployment Services Repository se soportan en los scripts heredados mediante los comandos `retrieve` y `store`, por ejemplo:

```
var s
set s = retrieve stream "/my repository folder/my_stream.str"
store stream ^s as "/my repository folder/my_stream_copy.str"
```

En los scripts Python, se accede a la funcionalidad equivalente a través del objeto Repository asociado a la sesión, por ejemplo:

```
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)
```

Nota: El acceso al repositorio exige que la sesión se haya configurado con una conexión de repositorio válida.

Operaciones de ruta

Algunos comandos de operación de ruta que normalmente se utilizan en IBM SPSS Modeler tienen sus comandos equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 247. Correlación de scripts de herencia con scripts Python para operaciones de ruta.

Scripts de herencia	scripts Python
`create stream NOMBREARCHIVO_PREDETERMINADO`	*ejecutortareas*.createStream(*nombre*, *autoConectar*, *autoGestionar*)
`close stream`	*ruta*.close()
`clear stream`	*ruta*.clear()
`get stream ruta`	Sin equivalente
`load stream vía de acceso`	Sin equivalente
`open stream vía de acceso`	*ejecutortareas*.openStreamFromFile(*vía de acceso*, *autoGestionar*)
`save ruta as vía de acceso`	*ejecutortareas*.saveStreamToFile(*ruta*, *vía de acceso*)
`retreive stream vía de acceso`	*repositorio*.retreiveStream(*vía de acceso*, *versión*, *etiqueta*, *autoGestionar*)
`store ruta as vía de acceso`	*repositorio*.storeStream(*ruta*, *vía de acceso*, *etiqueta*)

Operaciones de modelo

Algunos de los comandos de operación de modelo que normalmente se utilizan en IBM SPSS Modeler tienen sus comandos equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 248. Correlación de scripts de herencia con scripts Python para operaciones de modelo.

Scripts de herencia	scripts Python
`open model vía de acceso`	*ejecutortareas*.openModelFromFile(*vía de acceso*, *autoGestionar*)
`save modelo as vía de acceso`	*ejecutortareas*.saveModelToFile(*modelo*, *vía de acceso*)
`retrieve model vía de acceso`	*repositorio*.retrieveModel(*vía de acceso*, *versión*, *etiqueta*, *autoGestionar*)
`store modelo as vía de acceso`	*repositorio*.storeModel(*modelo*, *vía de acceso*, *etiqueta*)

Operaciones de resultado de documento

Algunos de los comandos de operaciones de resultado de documento que normalmente se utilizan en IBM SPSS Modeler tienen sus comandos equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 249. Correlación de scripts de herencia con scripts Python para operaciones de resultado de documento.

Scripts de herencia	scripts Python
`open output vía de acceso`	*ejecutortareas*.openDocumentFromFile(*vía de acceso*, *autoGestionar*)
`save resultado as vía de acceso`	*ejecutortareas*.saveDocumentToFile(*resultado*, *vía de acceso*)

Tabla 249. Correlación de scripts de herencia con scripts Python para operaciones de resultado de documento (continuación).

Scripts de herencia	scripts Python
`retrieve output` *`vía de acceso`*	*`repositorio`*`.retrieveDocument(`*`vía de acceso`*`, `*`versión`*`, `*`etiqueta`*`, `*`autoGestionar`*`)`
`store` *`resultado`* `as` *`vía de acceso`*	*`repositorio`*`.storeDocument(`*`resultado`*`, `*`vía de acceso`*`, `*`etiqueta`*`)`

Otras diferencias entre scripts heredados y scripts Python

Los scripts heredados soportan la manipulación de proyectos de IBM SPSS Modeler. Los scripts Python no soportan esto actualmente.

Los scripts heredados proporcionan cierto soporte de carga de objetos de *estado* (combinaciones de rutas y modelos). Los objetos de estado han caído en desuso desde IBM SPSS Modeler 8.0. Los scripts Python no soportan objetos de estado.

Los scripts Python proporcionan las siguientes funciones adicionales no disponibles en los scripts heredados:

- Definiciones de clase y función.
- Manejo de errores.
- Soporte más sofisticado de entrada/salida.
- Módulos externos y de terceros.

Avisos

Esta información se ha desarrollado para los productos y servicios ofrecidos en todo el mundo.

Puede que en otros países IBM no ofrezca los productos, servicios ni características que se describen en esta información. Póngase en contacto con el representante local de IBM, que le informará sobre los productos y servicios disponibles actualmente en su área. Las referencias a programas, productos o servicios de IBM no pretenden establecer ni implicar que sólo puedan utilizarse dichos productos, programas o servicios de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no infrinja ningún derecho de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y comprobar el funcionamiento de todo producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes pendientes que cubran la materia descrita en esta información. Este documento no le otorga ninguna licencia para estas patentes. Puede enviar preguntas acerca de las licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle
Drive
Armonk, NY 10504-1785
EE. UU.

Para consultas de licencias relacionadas con información de doble byte (DBCS), póngase en contacto con el Departamento de Propiedad intelectual de IBM de su país o envíe sus consultas, por escrito, a:

Intellectual Property Licensing
Derecho de propiedad intelectual y legal
IBM Japan Ltd.
1623-14,
Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país donde dichas disposiciones entren en conflicto la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN NINGÚN TIPO DE GARANTÍA, EXPLÍCITA O IMPLÍCITA, INCLUYENDO, PERO SIN LIMITARSE A, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERABILIDAD, COMERCIABILIDAD O ADECUACIÓN A UN PROPÓSITO DETERMINADO. Algunos estados no permiten la renuncia a expresar o a garantías implícitas en determinadas transacciones , por lo tanto , esta declaración no se aplique a usted.

Esta información puede incluir imprecisiones técnicas o errores tipográficos. Periódicamente, se efectúan cambios en la información aquí y estos cambios se incorporarán en nuevas ediciones de la publicación. IBM puede realizar en cualquier momento mejoras o cambios en los productos o programas descritos en esta publicación sin previo aviso.

Cualquier referencia a sitios Web que no sean de IBM en esta información solamente es ofrecida por comodidad y de ningún modo sirve como aprobación de esos sitios Web. Los materiales de dichos sitios web no forman parte de los materiales de este producto IBM y el uso de dichos sitios web se realiza a cuenta y riesgo del usuario.

IBM puede utilizar o distribuir cualquier información que se le proporcione en la forma que considere adecuada, sin incurrir por ello en ninguna obligación para con el remitente.

Los propietarios de licencia de este programa que deseen tener información sobre el mismo con el fin de: (i) intercambiar información entre programas creados de forma independiente y otros programas (incluido éste) y (ii) utilizar mutuamente la información que se ha intercambiado, deberán ponerse en contacto con:

Tel. 901 100 400
ATTN: Licensing
200 W. Madison St.
Chicago, IL; 60606
Estados Unidos de América

Esta información estará disponible, bajo las condiciones adecuadas, incluyendo en algunos casos el pago de una cuota.

El programa bajo licencia descrito en este documento y todo el material bajo licencia disponible se proporcionan bajo los términos de IBM Customer Agreement, IBM International Program License Agreement o cualquier otro acuerdo equivalente entre IBM y el cliente.

Cualquier dato de rendimiento mencionado aquí ha sido determinado en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar de forma significativa. Es posible que algunas mediciones se hayan realizado en sistemas en desarrollo y no existe ninguna garantía de que estas medidas sean las mismas en los sistemas comerciales. Además, es posible que algunas mediciones hayan sido estimadas a través de extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben consultar los datos que corresponden a su entorno específico.

Se ha obtenido información acerca de productos que no son de IBM de los proveedores de esos productos, de sus publicaciones anunciadas o de otros orígenes disponibles públicamente. IBM no ha probado estos productos y no puede confirmar la precisión del rendimiento, la compatibilidad ni ninguna otra afirmación relacionada con productos que no son de IBM. Las preguntas acerca de las aptitudes de productos que no sean de IBM deben dirigirse a los proveedores de dichos productos.

Todas las declaraciones sobre el futuro del rumbo y la intención de IBM están sujetas a cambio o retirada sin previo aviso y representan únicamente metas y objetivos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlos lo máximo posible, los ejemplos incluyen los nombres de las personas, empresas, marcas y productos. Todos esos nombres son ficticios y cualquier parecido con los nombres y direcciones utilizados por una empresa real es pura coincidencia.

Si está viendo esta información en copia electrónica, es posible que las fotografías y las ilustraciones en color no aparezcan.

Marcas comerciales

IBM, el logotipo de IBM e ibm.com son marcas registradas o marcas comerciales de International Business Machines Corp., registradas en muchas jurisdicciones en todo el mundo. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas. Encontrará una lista actual de marcas registradas de IBM en la Web en "Información de copyright y marca registrada" en www.ibm.com/legal/copytrade.shtml.

Intel, el logotipo de Intel, Intel Inside, el logotipo de Intel Inside, Intel Centrino, el logotipo de Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium y Pentium son marcas comerciales o marcas registradas de Intel Corporation o sus filiales en Estados Unidos y otros países.

Linux es una marca registrada de Linus Torvalds en Estados Unidos, otros países o ambos.

Microsoft, Windows, Windows NT, y el logotipo de Windows son marcas comerciales de Microsoft Corporation en Estados Unidos, otros países o ambos.

UNIX es una marca registrada de The Open Group en Estados Unidos y otros países.

Java y todas las marcas registradas y logotipos basados en Java son marcas comerciales o marcas registradas de Oracle y/o sus filiales.

El resto de nombres de productos y servicios pueden ser marcas comerciales de IBM o de otras empresas.

Índice

N

O

P

R

S

V

Impreso en España