|  |  |  |
| --- | --- | --- |
|  | | |
| TALLER Analytics  DE SOFIA2 | | |
|  | | |
|  | | |
| Septiembre 2016  Versión 1 | | |
|  | | |
|  |  |  |

# INDICE

[1 INDICE 2](#_Toc461726191)

[2 INTRODUCCION 3](#_Toc461726192)

[3 INGESTA DE LOS DATOS 4](#_Toc461726193)

[3.1 Creación Pipeline 4](#_Toc461726194)

[3.2 Definir Componente Origen 4](#_Toc461726195)

[3.3 Procesado de los datos 6](#_Toc461726196)

[3.4 Destino componente destino 8](#_Toc461726197)

[4 NOTEBOOK 10](#_Toc461726198)

[4.1 Definición de las rutas de los datos de entrada 10](#_Toc461726199)

[4.2 Estructurar los datos 10](#_Toc461726200)

[4.3 Comprobaciones de los datos 11](#_Toc461726201)

[4.4 Dividir el dataset 11](#_Toc461726202)

[4.5 Función para evaluar el modelo 11](#_Toc461726203)

[4.6 Elección del modelo 12](#_Toc461726204)

[4.7 Ejecutar las recomendaciones 12](#_Toc461726205)

# INTRODUCCION

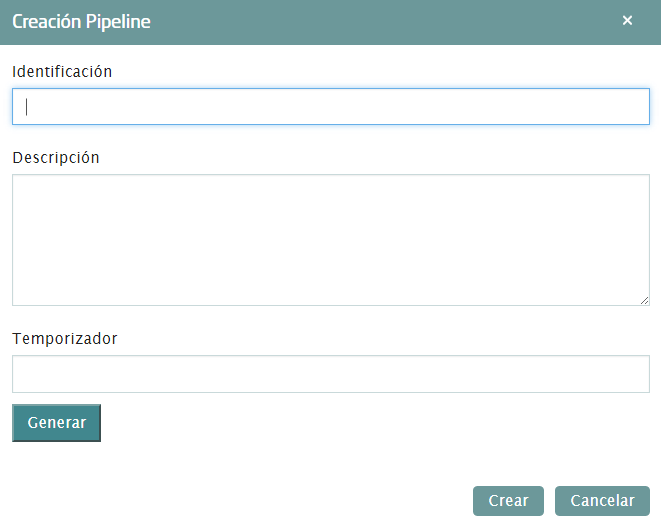
El objetivo de este taller es crear un sistema de recomendación en base a los ratings de los usuarios. Utilizaremos uno de los Dataset de [Movielens](https://movielens.org/) que ya reside en la plataforma. Lo haremos en dos pasos:

* Ingesta y preparación de los datos mediante Pipelines.
* Creación del modelo mediante Notebook.

# INGESTA DE LOS DATOS

## Creación Pipeline

Vamos a realizar la ingesta de los datos de películas con el Dataflow. Lo primero que hay que hacer es crear un Pipeline desde cero. Dentro de las opciones de Menú de Analytics, “Mis Pipelines”, y dentro de esta pantalla, hay que pulsar el botón de Crear. Aparecerá una ventana en la que introducir el nombre del Pipeline, una descripción y un temporizador, que para esta práctica no aplica:



Al crear el pipeline accede directamente al espacio de trabajo en el que crearemos el flujo de información.

## Definir Componente Origen

Los datos ya están descargados en la máquina de Sofia2. Concretamente, la ruta es “/datadrive/ftp/movielens”. En este directorio deberían existir dos ficheros: *movies.dat* y *ratings.dat*. Para este pipeline nos interesan los datos de las películas.

Primero, es necesario crear un Origen de los datos. Como los ficheros ya residen en la máquina de Sofia2, el componente que se necesita es Directory. Pulsa sobre el componente y aparecerá en el espacio de trabajo. Verás que salen alertas de errores. No te preocupes, al crear el componente vacío, los parámetros de configuración obligatorios están vacíos. Eso es justamente lo que hay que hacer en el siguiente paso.

Pulsa sobre el componente y accederás a su configuración. Para el origen de directorio local, los parámetros de configuración obligatorios son:

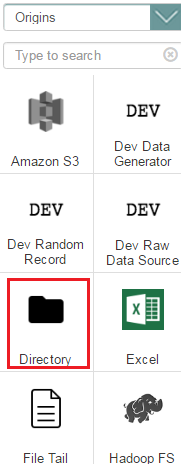
**Files → Data Format**: Representa el formato de los datos de entrada. Hay diferentes opciones, pero la que se necesita en este ejemplo es Text.

**Files → Files Directory**: Es el directorio de entrada, donde residen los ficheros a leer. En nuestro caso, esta ruta es /datadrive/ftp/movielens.

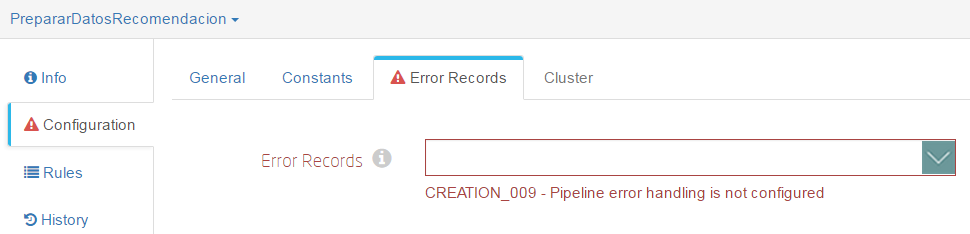
**Files → Name Pattern**: Es la expresión regular con la que buscará los ficheros a cargar dentro del directorio configurado en el parámetro anterior.

Nos interesa leer un solo fichero, por lo que hay que asignar a este campo en movies.dat.

Dependiendo del formato de entrada elegido, se activa la pestaña correspondiente en la ventana de configuración. Verás que en este caso, la pestaña activa es Text. Solo tiene un parámetro que es Max Line *Length* que tiene un valor por defecto que no vamos a modificar.



Ya está configurado el origen. Para empezar, es muy recomendable echar un vistazo a los datos que se van a leer. Para ello, podemos configurar un destino “Dummy” y previsualizar la información. Para esto, accede a los componentes destino y elige “Trash”. Como antes, al pulsar sobre el icono, aparece el componente en el espacio de trabajo. Une origen y destino, y ya casi está preparado este flujo. Como observarás, todavía hay errores de configuración. Esto es porque en la configuración general hay que definir la gestión de registros erróneos. Pulsa en cualquier sitio que no sea un componente dentro del espacio de trabajo. La ventana inferior mostrará la configuración general, y verás que la alerta aparece en la pestaña “Error Records”.

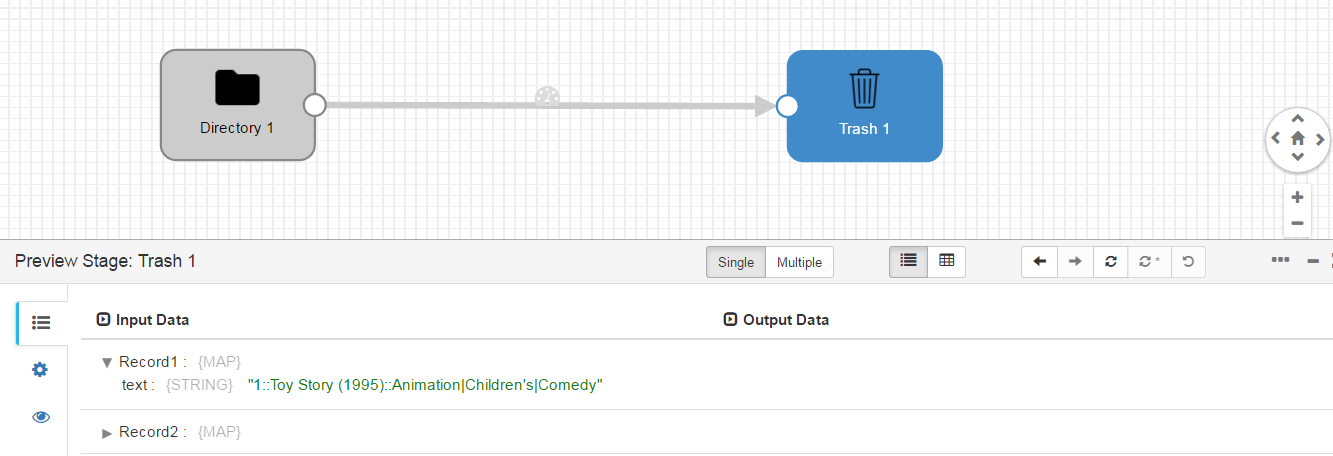


Dentro de esas opciones elige “Discard”. Con esto, ya no debería haber errores, pero aún así vamos a validar el flujo. En las opciones del menú de la barra superior, pulsa sobre el botón “Validate”:



Si todo es correcto, mostrará un mensaje de OK.

Ya podemos hacer la previsualización. Dentro del menú anterior, el botón justo a la izquierda de Validate es “Preview”. Pulsa sobre él y aparecerá una ventana con unos datos de configuración. Lo único que realmente hay que tener en cuenta para este caso es el check de “Write to destinations”. Si está marcado, además de previsualizar los datos los escribirá en destino. Desmárcalo si es que está marcado y pulsa sobre “Run Preview”:

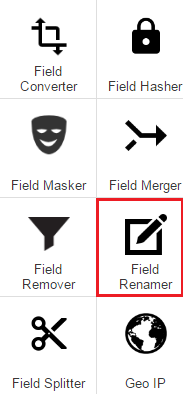


En input data puedes ver lo que lee en cada registro y en cada uno de los componentes. Si pulsas sobre el componente directory, verás lo que genera y si pulsas sobre Trash lo que recibe. En este caso es lo mismo.

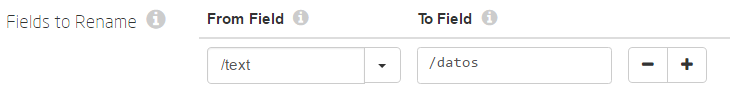
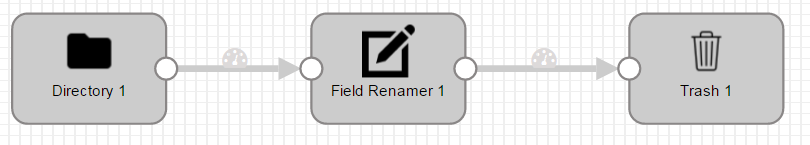
## Procesado de los datos

Ahora vamos a hacer la preparación de los datos. Como has podido observar en el preview del paso anterior, los campos están separados por “::”. El Dataflow, interpreta los separadores como un solo carácter, por lo que no se puede definir como delimitador “::”. Esto es lo siguiente que haremos.

Por comodidad, vamos a incluir, antes del cambio de delimitador, un renombrado de campos. En el preview, al desplegar cada registro aparecen los campos definidos. Al leer como formato Text, para cada línea se genera un campo que por defecto se llama “text”. Este es el que vamos a renombrar. Para ello, dentro de “Processors”, pulsa sobre “Field Renamer”. Crea un flujo como el siguiente:



Ahora hay que configurarlo. Este componente es muy simple. Pulsa sobre él, y en su configuración accede a la pestaña “Rename”. En “Fields to Rename” hay que introducir el campo origen y el nombre al que cambiarlo. Escribe como “From Field” /text y como “To Field” /datos.



Puedes probar a previsualizar para comprobar que efectivamente está renombrando el campo.



Ahora ya podemos crear el componente que sustituye el delimitador. Para llevar a cabo esta tarea se pueden usar diferentes processors, concretamente todos los que son “Evaluators”. Nosotros lo haremos con el de JavaScript. Como siempre, pulsa sobre el componente y crea un flujo como el siguiente:



Accede a la configuración del componente, y entra en la pestaña Javascript. Verás un editor de texto que se llama “script”, que ya tiene código predefinido dentro. Es la plantilla sobre la que definiremos nuestros cambios. Dentro del bucle for, añade la siguiente línea de código:

*records[i].value['datos'] = records[i].value['datos'].replace(/::/g, "%");*

Esta línea lo que hace es reemplazar “::” por “%”. Hemos elegido ese delimitador porque los típicos que suelen ser “;”, “,” y “|” aparecen en el dataset como parte de los campos. Lanza de nuevo el preview y comprueba que se ha realizado el cambio correctamente.

## Destino componente destino

De nuevo, pulsa sobre el componente y crea un flujo como el siguiente:

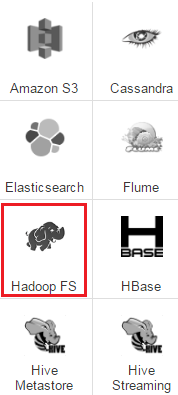


Accede a la configuración del destino. Hay que modificar 3 pestañas:

Hadoop FS: Corresponde a las conexiones y rutas del HDFS

**Hadoop FS URI**: hdfs://localhost:8020

**HDFS User**: cloudera-scm



Output Files: Es la definición de los ficheros de salida, rutas, formato, etc.

**File Type**: Text Files

**Data Format**: Text

**Files Prefix**: movie

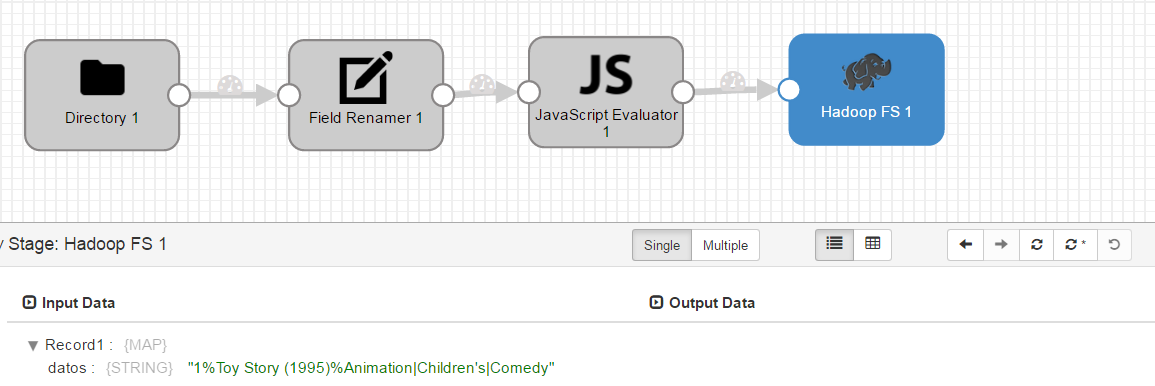
**Directory Template**: /user/cloudera-scm/movielens/alias\_alumno/

Text: Es la configuración del formato elegido en la pestaña anterior.

**Text Field Path**: /datos

Lanza el preview de nuevo y comprueba que los datos llegan correctamente al destino:

Si todo parece correcto, pulsa sobre el botón de “Start”, a la derecha del botón de validación que has usado anteriormente. Verás que se abre otra ventana con las estadísticas de los datos que se van leyendo, tiempos de proceso de cada componente, etc. Cuando veas que ya no está leyendo datos, significa que ya ha recorrido todo los ficheros de entrada. Como nosotros no necesitamos más datos que esos, podemos parar el pipeline.



¿Sabrías hacer lo mismo para el fichero de Ratings?

¿Sabrías generar el fichero en el HDFS como delimitado, definiendo los nombres de los campos separados por “;”?

# NOTEBOOK

Con ayuda de los notebooks de Sofia2 vamos a generar el modelo de recomendación de películas usando los datos que hemos cargado en la plataforma en el ejercicio anterior. Proponemos llevarlo a cabo con Spark usando Scala, y más concretamente implementaremos el ALS.

## Definición de las rutas de los datos de entrada

El primer paso es leer los datos de películas y ratings, y para eso primero hay que que definir la ruta de los datos. Define las variables *ratings path* y *movies\_path* con las correspondientes rutas donde hayas hecho la carga a la plataforma.

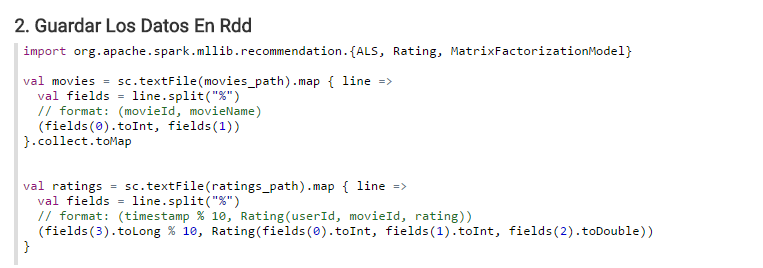


## Estructurar los datos

Lo siguiente es guardar la información de películas y puntuaciones. Vamos a leer dicha información mediante RDDs de Spark.

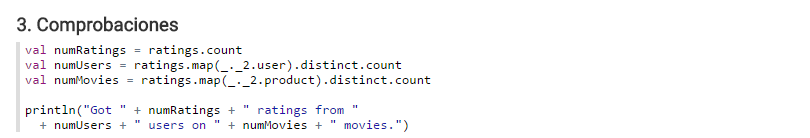
Hay que definir un formato concreto tanto para las películas: (movieId, movieName) como para los rating: (timestamp % 10, Rating(userId, movieId, rating)).

También aprovechamos a importar las librerías de Mlib que se van a usar en el ejemplo. En concreto se necesitan [**ALS**](https://spark.apache.org/docs/1.1.0/api/java/org/apache/spark/mllib/recommendation/ALS.html), [**Rating**](https://spark.apache.org/docs/1.1.0/api/java/org/apache/spark/mllib/recommendation/Rating.html) y [**MatrixFactorizationModel**](https://spark.apache.org/docs/1.4.0/api/java/org/apache/spark/mllib/recommendation/MatrixFactorizationModel.html).



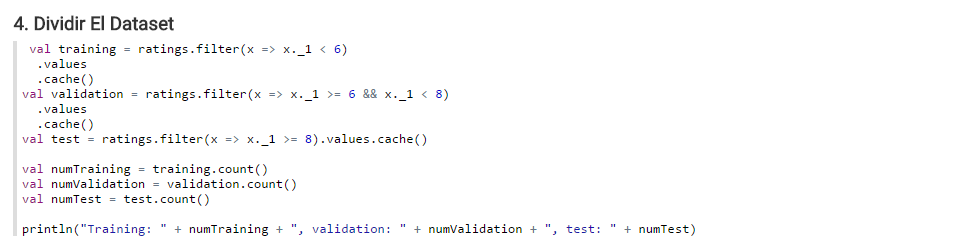
## Comprobaciones de los datos

Ahora, comprueba que efectivamente se han leído los datos. ¿Cuántas puntuaciones has descargado? ¿Cuántas películas hay en el catálogo? ¿Cuántas películas se han puntuado? ¿Y cuántos usuarios lo han hecho?



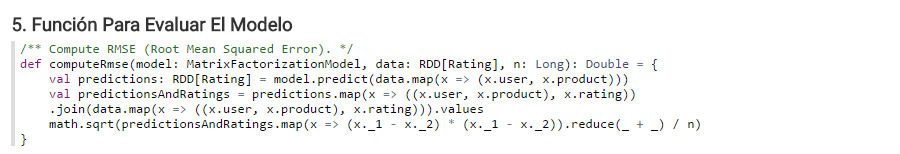
## Dividir el dataset

Antes de construir el modelo hay que dividir el dataset en partes más pequeñas, una para entrenamiento(60%), otra para validación(20%) y otra más para testing(20%).



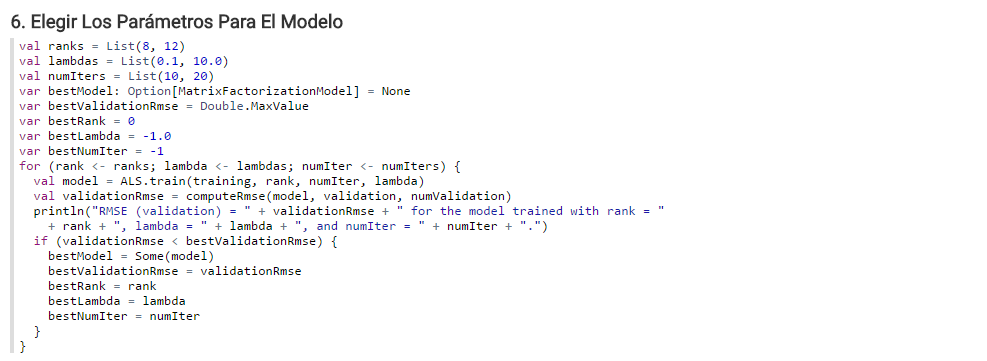
## Función para evaluar el modelo

Una vez divididos los datos, definamos la función que evaluará el rendimiento del modelo. En concreto usaremos [**Root Mean Squared Error (RMSE)**](https://en.wikipedia.org/wiki/Root-mean-square_deviation) y esta es la versión en Scala:



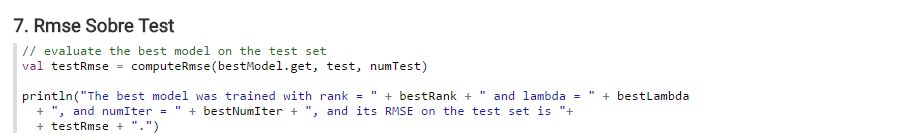
## Elección del modelo

Ahora puedes usar esta función para definir los parámetros para el algoritmo de entrenamiento. El algortimo ALS requiere 3 parámetros: el rango de la matriz de factores, el número de iteraciones y una lambda. Vamos a definir diferentes valores para estos parámetros y probar diferentes combinaciones de ellos para determinar cuál de ellas es la mejor:



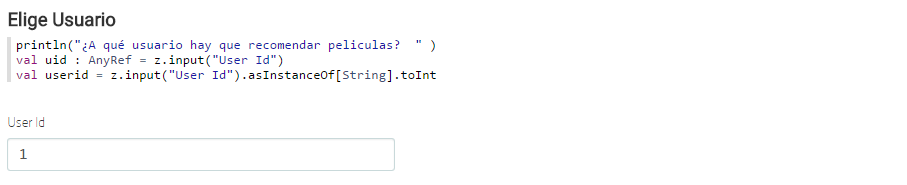
¿Cuál crees que es el mejor modelo?

Ahora vamos a lanzar nuestra función sobre los datos de Test.



## Ejecutar las recomendaciones

Una vez elegido el mejor modelo ya solo quedan las recomendaciones de películas por usuario. La idea es preguntar por el usuario, que para el Dataset usado es un numérico. Vamos a hacerlo tipo formulario, de tal forma que primero pregunte por el usuario, se inserte en un campo de texto y por último lance la recomendación. Para preguntar por el usuario:



Ahora, ya solo queda lanzar la recomendación. Para este ejemplo, definimos que se muestren las 10 mejores recomendaciones para el usuario insertado en el campo de texto.

