Caso de analítica escalable. Análisis con tecnologías de computación paralela y escalable © EDICIONES ROBLE, S.L.

Indice

Caso de analítica escalable. Análisis con tecnologías de computación paralela y escalable

Caso de analítica escalable. Análisis con tecnologías de computación paralela y escalable | Introducción - Miguel Monzón

I. Introducción

Dentro del mundo relacionado con el concepto de Big Data se pueden encontrar dos áreas de alguna manera separadas: **arquitecturas Big Data** y **analítica** junto con ***machine learning.***

Arquitecturas Big Data

La primera busca diseñar e implementar diferentes soluciones arquitectónicas para resolver problemas de computación y almacenamiento en entornos en los que son necesarias la escalabilidad, la veracidad de los datos o la velocidad de acceso o procesamiento de estos. En este ámbito se trabaja con grandes cantidades de datos, posiblemente en continua ingesta, y se preparan para ser preprocesados y analizados con posterioridad.

Área de analítica y *machine learning*

En cambio, en el área de analítica y *machine learning,* el conjunto de datos con el que se trabaja es más reducido. Si bien el conocimiento que se aplica es diferente, ya que se trata de encontrar posibles "patrones" dentro de los datos para poder realizar diversos cálculos como predicciones, clasificaciones, búsqueda de datos anómalos, recomendaciones, etc., los datos que se suelen estudiar pueden ser muestreos, que generan modelos que se aplican posteriormente al resto de los datos.

Sin embargo, existe una convergencia entre ambas áreas. Se pueden encontrar casos en los que sea necesario realizar diferentes procesos de analítica, pero entrenando los modelos con grandes cantidades de datos. Es decir, se pueden unir los dos ámbitos creando un flujo de analítica que abarque desde la ingesta y el preprocesamiento hasta el análisis, la creación de modelos y la evaluación de estos. Todo ello en un entorno escalable, entendiendo escalabilidad como la propiedad de no perder rendimiento (o perder el menor rendimiento posible) a medida que crece la cantidad de datos/peticiones/computación con la que se trabaja.

En esta unidad, se va a mostrar un **caso de analítica escalable** con una de las librerías más utilizadas y con mejor proyección en este ámbito: **Apache Spark.** Este *framework* de computación permite trabajar con grandes cantidades de datos de forma sencilla, con API en distintos lenguajes de programación, de modo que no hace falta preocuparse por la gestión de las peticiones de computación a lo largo de los clústeres, ya que se trabaja a alto nivel, y es posible centrarse en el diseño y la implementación de las soluciones directamente.

Además, se estudiarán dos portales donde poder buscar tanto fuentes de datos abiertas, reconocidas y estudiadas por una comunidad, como para ejecutar este código de Apache Spark accediendo a un clúster de forma totalmente gratuita.

Para el correcto seguimiento de la unidad, se requiere la consulta de los siguientes *notebooks*/documentos HTML en el orden descrito a continuación:

AnaliticaEscalablePySpark.dbc/html

Este *notebook* de Databricks recoge todo el contenido teórico de esta unidad, es una mezcla entre documentación y código y debe utilizarse en el apartado VI de la presente unidad. También se entrega el *notebook* en formato HTML para poder visualizarlo sin necesidad de abrirlo con Databricks. Se puede descargar en este enlace y también se puede consultar en HTML aquí.

AnaliticaEscalablePySparkEjercicios.dbc/html

Una vez visto el *notebook* teórico, se solicitará al alumno que resuelva diferentes ejercicios de forma gradual (lo que significa que, para resolver la mayoría de los ejercicios, hay que resolver los anteriores). Este *notebook* se llama AnaliticaEscalablePySparkEjercicios.dbc y también tiene un archivo análogo en HTML. Estos ejercicios conforman la primera actividad del caso práctico final y evaluable de este módulo. El archivo se facilitará al final de la unidad.

AnaliticaEscalablePySparkEjerciciosSolucion.dbc/html

Una vez completado y evaluado el caso práctico final del módulo, el alumno podrá disponer de las soluciones de cada uno de los ejercicios en ambos formatos (Databricks y HTML): AnaliticaEscalablePySparkEjerciciosSolucion.dbc.

Se facilitará dicho archivo en el módulo 10, es decir, después de haber sido calificado/a por el tutor en este módulo 9.

AnaliticaEscalablePySparkCompleto.dbc/html

Por último, también se entregará un *notebook* final en el que se integrarán todos los ejercicios solucionados junto con la parte teórica (facilitada anteriormente) y explicaciones adicionales, en ambos formatos: AnaliticaEscalablePySparkCompleto.dbc.

Se facilitará dicho archivo en el módulo 10, es decir, después de haber sido calificado/a por el tutor en este módulo 9.

II. Objetivos

En este caso práctico de analítica escalable, los alumnos alcanzarán los siguientes objetivos:

1. Buscar fuentes de datos de diversos temas y tamaños, para aplicar sobre ellos diferentes tareas analíticas y de procesamiento según cada fuente en particular.
2. Utilizar una herramienta web gratuita que ofrece un clúster de computación y una interfaz similar a los *notebooks* de Jupyter para ejecutar y documentar código de forma muy sencilla.
3. Realizar una primera ingesta y preprocesamiento de datos utilizando la librería de Python Pandas.
4. Diseñar y programar diferentes soluciones de procesamiento y analítica utilizando Apache Spark, más concretamente PySpark, ingestando datos para crear objetos representativos de datos tabulares con Spark-SQL.
5. Realizar búsquedas y analizar estos datos sin necesidad de crear modelos.
6. Crear diferentes modelos de *machine learning* sobre estos datos con Spark-ML. Entre los modelos se encuentran:
    a. *Clustering*.
    b. Clasificación supervisada.
    c. Regresión lineal.
7. Rediseñar los modelos de *machine learning* para ejecutarlos en forma de flujos de analítica o *pipelines.*

III. Familiarizarse con Apache Spark

3.1 Breve introducción sobre Apache Spark

Apache Spark es uno de los *frameworks* de computación paralela más utilizados en la actualidad.

Este proyecto fue creado por MateiZaharia en 2009 en la UC Berkeley's AMP Lab, y se convirtió en *open source* en 2010 bajo la licencia BSD.

Toda su funcionalidad se encuentra orientada en torno a los ***resilient distributed datasets*** (RDD), *sets* de datos de solo lectura (*read-only*) que se encuentran distribuidos en diferentes nodos y que, además, tienen tolerancia a fallos. La propia arquitectura de Apache Spark ofrece una alternativa a Map Reduce y utiliza más la memoria principal y menos los accesos a disco, por lo que asegura mejores rendimientos: cien veces más rápido que Hadoop *in-memory* y diez veces más cuando se accede a disco.

A la hora de ejecutar código y procesos, Apache Spark utiliza el concepto de ***lazy evaluation,*** que significa que no ejecuta realmente el código hasta que sea necesario. De esta forma, el código de Spark se puede considerar una "receta" hasta llegar a la ejecución final, la cual activa todo el proceso y ejecuta todos los pasos. Así se consigue que no se ejecute código de forma innecesaria.

Por último, Apache Spark permite programarlo en una amplia gama de lenguajes. Ofrece API para Java, R (SparkR), Scala y Python (PySpark), aunque los dos últimos son los más utilizados. **Scala** presenta el mayor uso ya que es totalmente compatible con el *core* de Spark, además de que la filosofía de la programación funcional que este lenguaje representa (lambdas) encaja con el procesamiento que se realiza sobre los datos en este *framework.* **Python** también se utiliza con mucha frecuencia ya que permite, de forma muy sencilla, la transición de la programación de librerías de manipulación de datos o de *machine learning* como Pandas o Scikit-learn.

3.2 Componentes de Apache Spark

Una de las grandes ventajas de Apache Spark respecto a otros *frameworks* de paralelización es que incluye una gran variedad de componentes con diferentes finalidades.

Si, por ejemplo, se escoge el ecosistema Hadoop, se puede encontrar:

- HDFS: almacenamiento distribuido.
- MapReduce: procesamiento distribuido.
- Hive: tratamiento de datos SQL.
- Mahout: *machine learning.*
- Storm: *streaming.*
- Giraph: *graph processing.*

Esto significa que hay que aprender una tecnología nueva para cada caso de uso que se quiera aplicar.

Sin embargo, Apache Spark ofrece una serie de componentes en su *framework* que permiten integrarlos todos sin ningún tipo de problema.

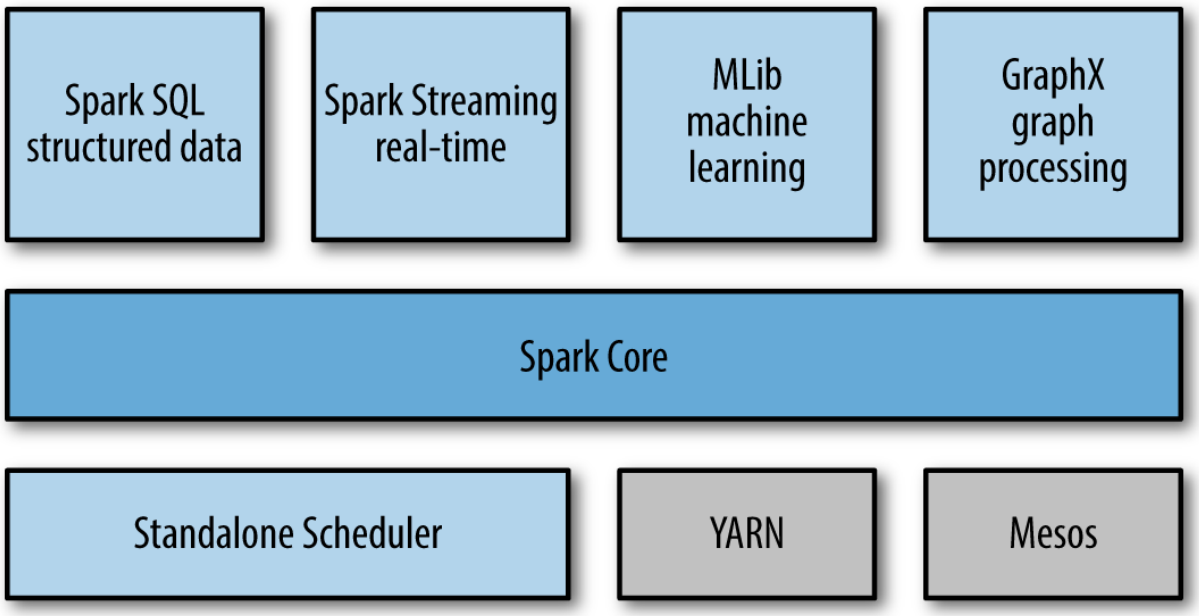

Figura 1: Componentes de Spark. Fuente: https://www.safaribooksonline.com/library/view/learning-spark/9781449359034/assets/lnsp_0101.png

Los componentes que ofrece Apache Spark son:

Spark Core
Funcionalidad básica de Spark. Permite el manejo de los RDD, *datasets* distribuidos sobre los que se puede aplicar computación paralela.
Spark SQL
Permite transformar los RDD a un formato tabular (DataFrames) aplicando ciertas mejoras y optimizaciones sobre ellos. Además, permite la ingesta de numerosas fuentes de datos.
Spark Streaming/Structured Streaming
Librería para poder ingestar y procesar datos en un flujo continuo, con respuestas de baja latencia. La primera de ellas trabaja con RDD, mientras que la segunda trabaja con DataFrames.
Spark MLLib/Spark ML
Funcionalidades para aplicar algoritmos de *machine learning* de forma escalable y distribuida. La primera de ellas trabaja con RDD, mientras que la segunda trabaja con DataFrames. Spark ML además integra el diseño y programación de flujos de analítica o *pipelines*.
Spark GraphX/Spark GraphFrames
Permite la ingesta y aplicación de diversas funciones y cálculos sobre datos tratados como grafos. La primera de ellas trabaja con RDD, mientras que la segunda trabaja con DataFrames.

IV. Buscar una fuente de datos en Kaggle

Kaggle es una plataforma web con diferentes competiciones de análisis donde los usuarios compiten por lograr el mejor modelo que se pueda aplicar sobre ciertos *sets* de datos subidos por la comunidad o por empresas. Se basa en la premisa de que existen incontables formas de abordar un problema de analítica y en que no hay forma de saber que una solución vaya a ser la óptima.

Gracias a esta plataforma, se puede conseguir:

- *Sets* de datos totalmente gratuitos, de diversos temas y de diversos tamaños.
- Soluciones ya publicadas de otros usuarios, por lo que se puede aprender de las que ya han sido propuestas e intentar mejorarlas.
- Publicar tus propias soluciones y conseguir con ello un *portfolio* orientado hacia el *data science.*
- Participar en una competición contra otros, o bien crear una nueva.
- Incluso se pueden conseguir oportunidades laborales en este ámbito.

1

En esta unidad, se va a buscar un *dataset* que esté relacionado con el campo empresarial; por lo tanto, se buscará en la sección “Business”. Para ello, hay que acceder a la página web de Kaggle (https://www.kaggle.com) y seleccionar el botón “Datasets”.

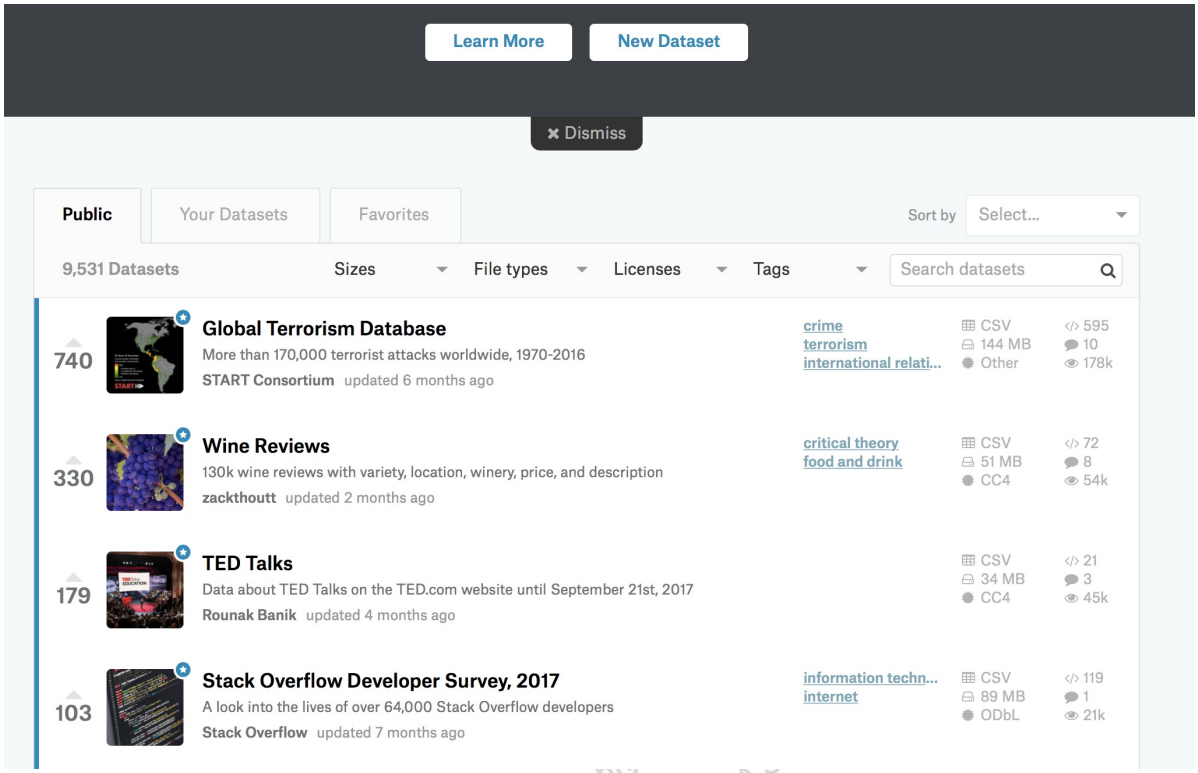

Figura 2: *datasets* de kaggle.

2

En los filtros, seleccionar el de etiquetas (“Tag”), escribir “Business” y aplicar el filtro. De los *datasets* devueltos, se va a escoger el que tiene por nombre: “515K Hotel Reviews Data in Europe”.

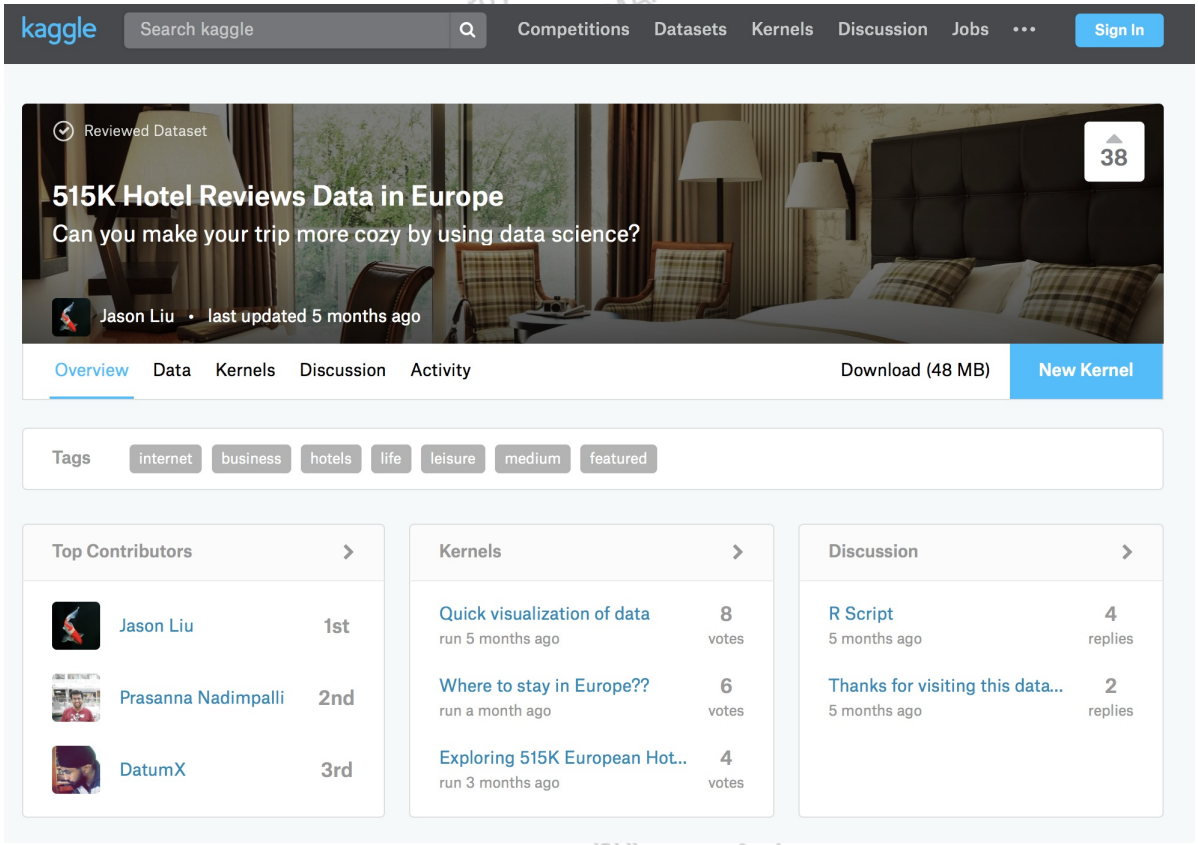

Figura 3: *Dataset* escogido para la unidad.

3

Como se puede observar, Kaggle ofrece información sobre el *dataset* elegido, como las etiquetas que tiene, los mejores contribuidores, los *kernels* existentes (soluciones de otros usuarios) e incluso discusiones sobre el propio *dataset.* Si, por ejemplo, se elige el primero de los *kernels* (https://www.kaggle.com/jiashenliu/quick-visualization-of-data/), se puede ver cómo ha resuelto uno de los usuarios la visualización de los datos, en este caso utilizando R. Gracias a esto, se puede aprender a conseguir, en este caso, potentes visualizaciones para casos similares a este de la siguiente forma:

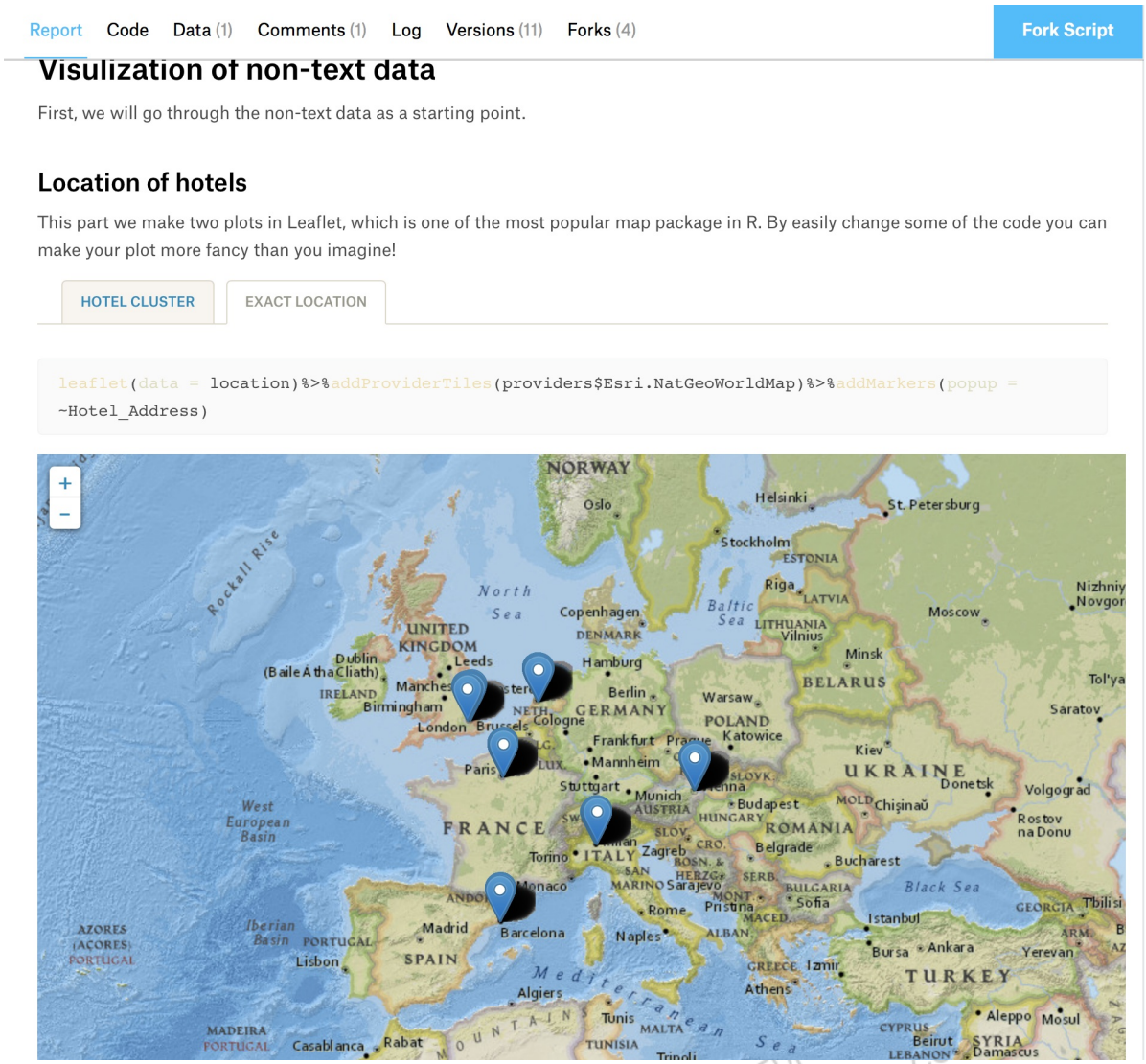

Figura 4: una solución analítica del *dataset* elegido.

```
4
```

Volviendo a la página principal, si se selecciona la pestaña de datos ("Data"), se puede ver información relevante como el tamaño del *dataset,* si está formado por uno o más ficheros e incluso una vista previa de estos. También se pueden ver los metadatos de las columnas, es decir, sus tipos y posibles descripciones.

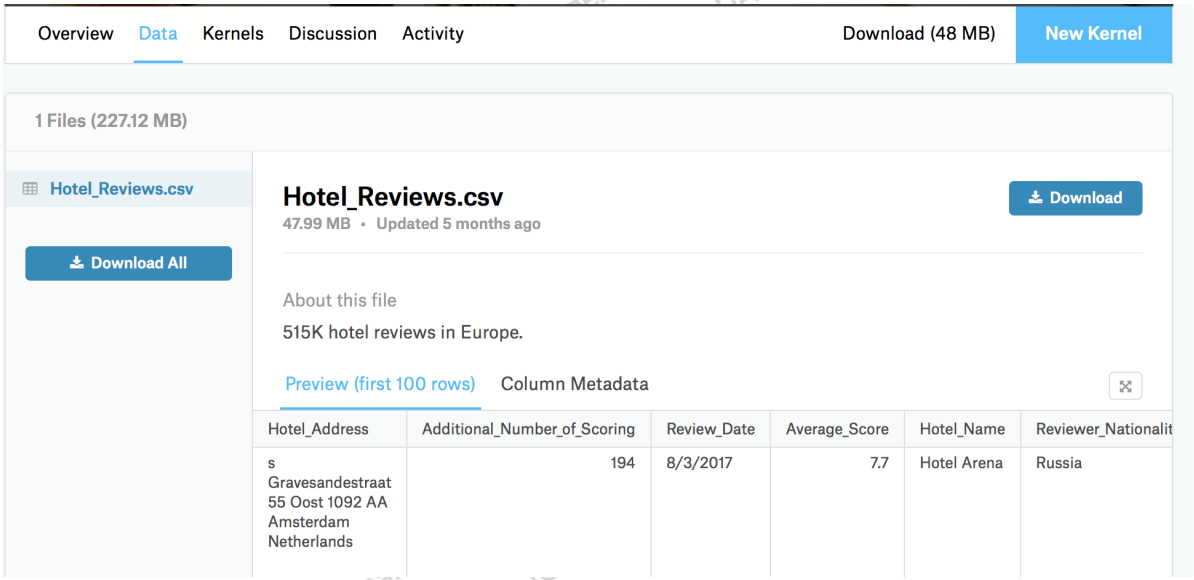

Figura 5: *preview* de los datos.

Para descargar el *dataset,* hay que seleccionar el botón "Download" y registrarse si no se tiene ya un usuario en la plataforma. Una vez se ha accedido con el usuario, el *dataset* podrá descargarse.

V. Databricks Community

Databricks es una empresa creada por los propios fundadores de Apache Spark, entre ellos Matei Zaharia, que ofrece soluciones Big Data con este *framework* de paralelización. Ofrece un portal web en el que se podrá ejecutar código Spark con una gestión de clústeres sencilla y automatizada, además de una cómoda y potente forma de programar el código utilizando unos *notebooks* similares a los de Jupyter. Además, ofrece una gran cantidad de cursos *online* sobre Spark y organiza una de las mayores conferencias sobre Spark, Spark-summit.

En su versión Community, Databricks permite el acceso gratuito a su plataforma, con una gran cantidad de componentes disponibles, además de ofrecer un clúster de 6 GB sin tener que efectuar ningún pago.

5.1 Crear una cuenta

El primer paso para acceder a la plataforma es crearse una cuenta. Para ello, hay que acceder a la página web https://community.cloud.databricks.com/login.html y registrarse (si no se dispone ya de un perfil de usuario en ella). Al crear una cuenta nueva, hay que seleccionar el plan "Community Edition":

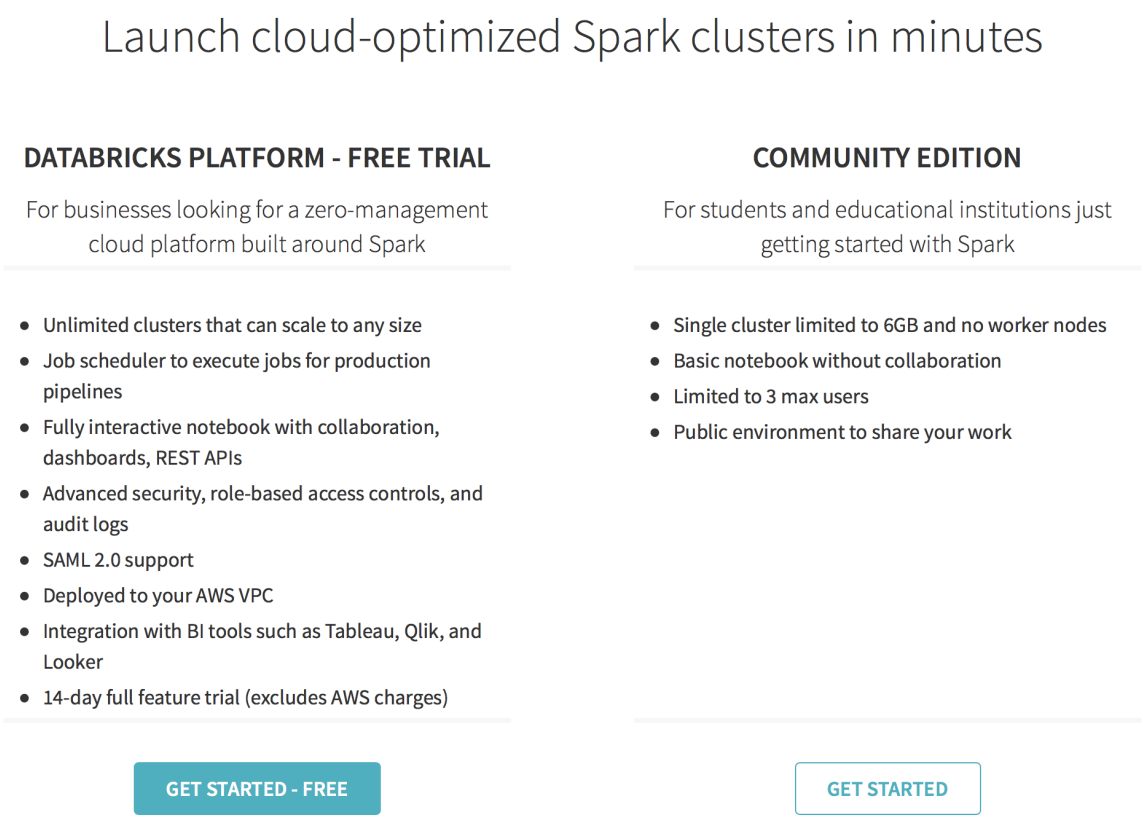

Figura 6: Planes de Databricks.

Tras completar el proceso de prueba, ya se puede acceder con un nuevo usuario a la plataforma.

5.2 Navegar por la interfaz

Al acceder a la plataforma se puede ver la siguiente pantalla:

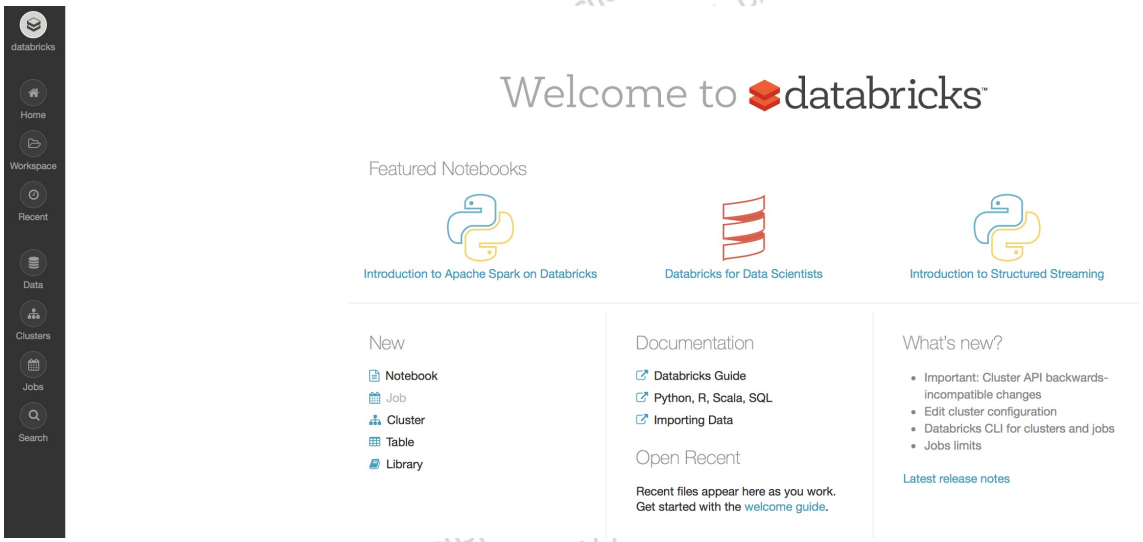

Figura 7: Interfaz de Databricks.

Los elementos disponibles son:

- *Notebooks* y documentación de introducción y aprendizaje.
- *Home:* carpeta personal.
- *Workspace:* entorno de trabajo actual.
- *Recent:* ficheros abiertos recientemente.
- *Data.*
- *Clusters.*
- *Jobs.*
- *Search.*

Al seleccionar la carpeta personal, se puede observar que ofrece documentación, tutoriales y espacio compartido y por usuario. En pasos posteriores, se creará un *notebook* dentro de esta sección.

Una de las opciones que se utilizará primero es la de clústeres. Al seleccionar el botón, permitirá la creación de un clúster de 6 GB con las opciones de versión. Es importante recordar que pasadas dos horas el clúster se detendrá y se eliminará, aunque se puede crear otro clúster inmediatamente después. Para este ejercicio, se utilizará el clúster 3.5 LTS (Spark 2.2.1 y Scala 2.11):

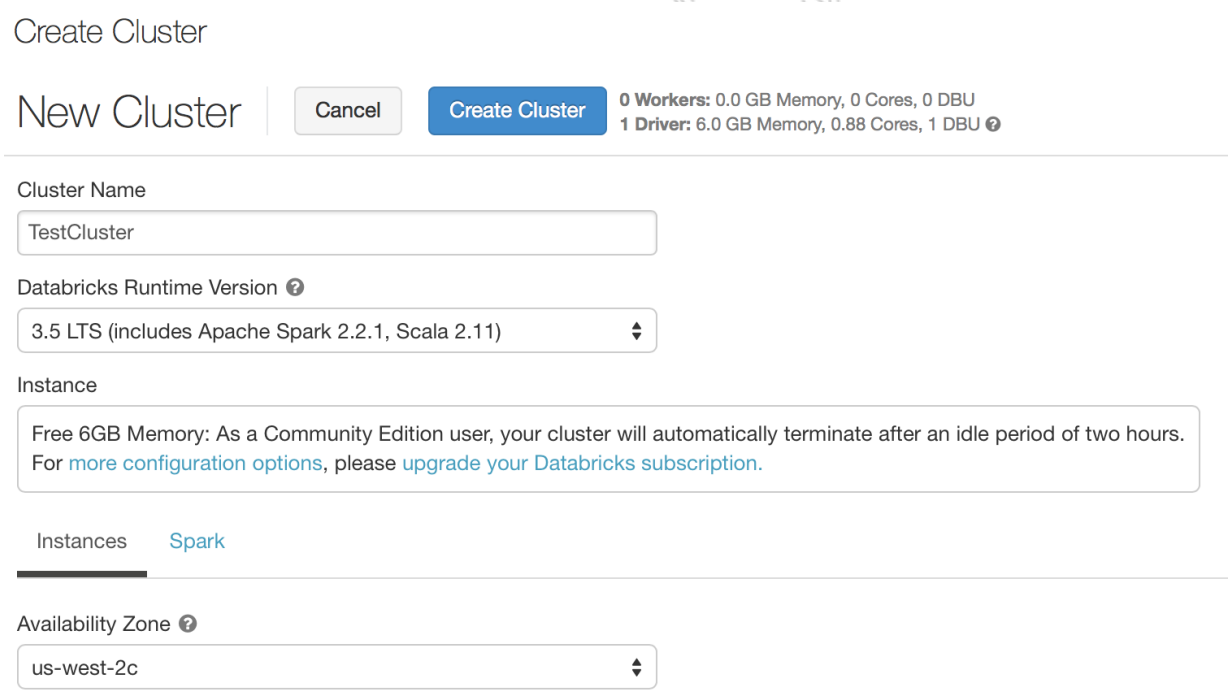

Figura 8: Crear un Clúster.

Por ahora no vamos a crear el clúster, cuando llegue el momento, es esencial remitirse a este paso.

5.3 Subir los datos

Al seleccionar la pestaña “Data”, se verá que efectivamente no hay nada subido todavía. En la pestaña “Tables”, se pueden añadir datos usando el símbolo “+”.

Como muestra la interfaz, se pueden subir ficheros desde un equipo local, Amazon S3, DBFS (similar a HDFS, pero dentro de Databricks) y otros conectores como Amazon Redshift, Amazon Kinesis, JDBC, Cassandra o Kafka. En este caso se optará por subir el *dataset* de Kaggle descargado mediante la opción de subida desde el equipo local. El fichero se guardará automáticamente en DBFS (se accederá a este posteriormente).

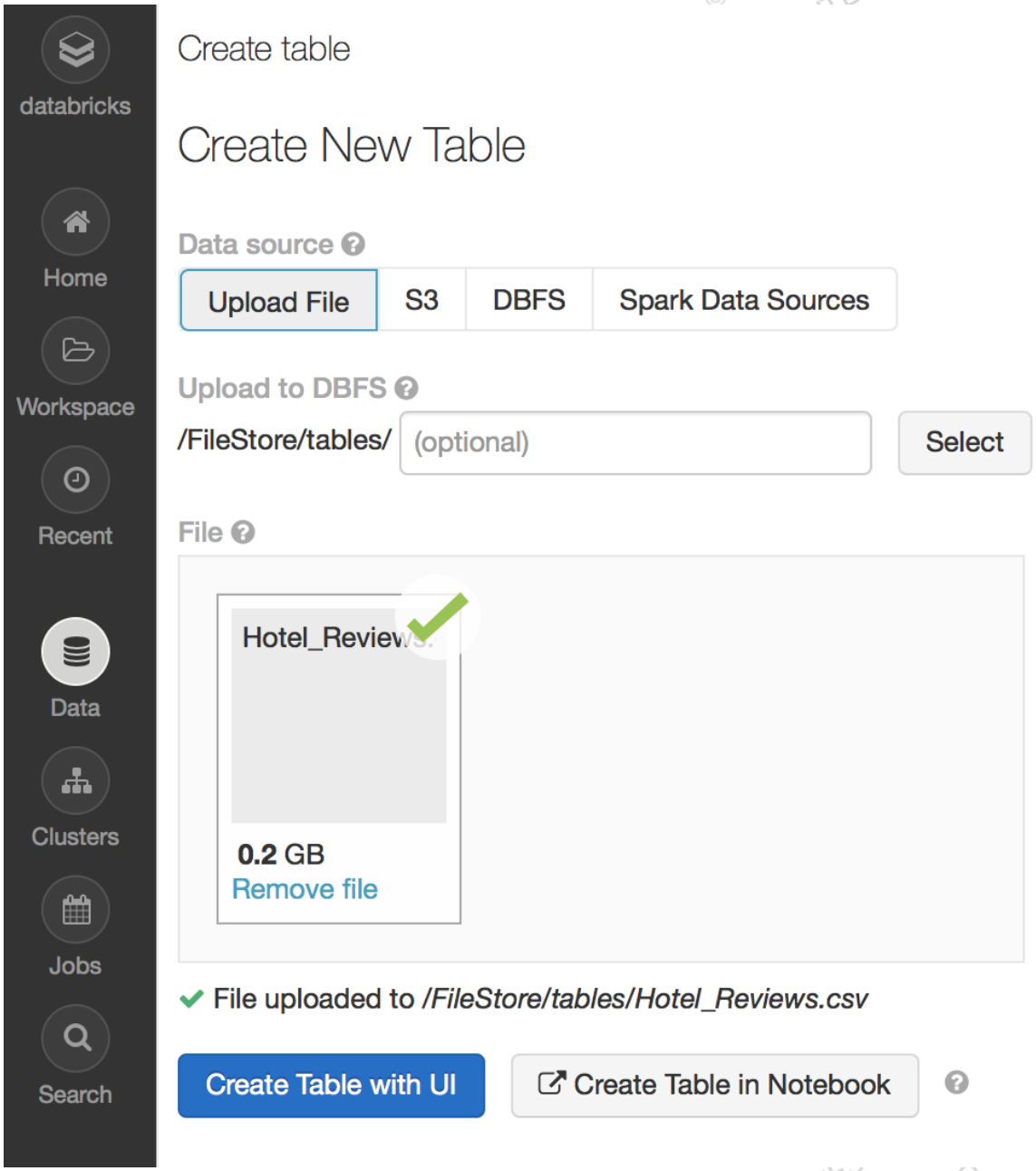

Figura 9: Ingestar los datos en Databricks.

El proceso nos indica que el fichero está creado en la ruta */FileStore/tables/Hotel_Reviews.csv,* lo cual es importante comprobar, ya que será la ruta que utilizará el *notebook* para acceder a los datos.

VI. Subir el caso práctico y ejecutarlo

Para esta práctica se proporciona un fichero llamado AnaliticaEscalablePySpark.dbc (véase la introducción de esta unidad), que contiene el código del caso práctico que se va a ejecutar.

Para importarlo, basta con ir al entorno de trabajo “Workspace” y pinchar con el botón derecho del ratón sobre la primera columna. A continuación, es preciso seleccionar la opción “Import” y así se importará el *notebook.*

1

Una vez importado, es hora de retomar el paso de arranque del clúster visto anteriormente. Se creará un clúster con los mismos parámetros que en la imagen, de modo que el resultado será el siguiente:

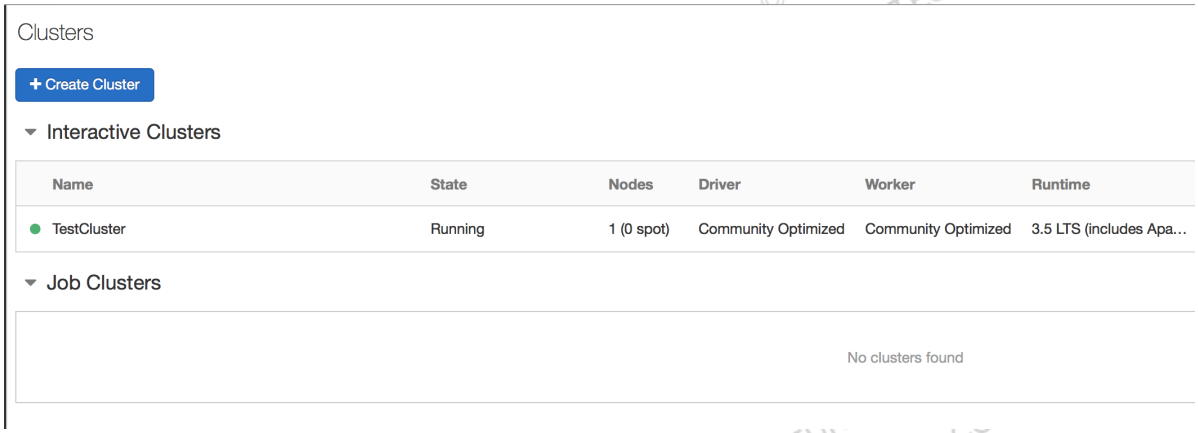

Figura 10: Nuestro clúster está creado y ejecutándose.

2

Entre las opciones disponibles, se puede terminar un clúster, reiniciarlo o clonarlo. Una vez creado el clúster, es momento de abrir el *notebook* subido con anterioridad. Al seleccionarlo, este se abrirá y mostrará una interfaz similar a la de un *notebook* de Jupyter:

Figura 11: Abrir el contenido del *notebook.*

3

Para ejecutar el código, es necesario enlazar el *notebook* a un clúster en ejecución; en este caso, el que se acaba de iniciar. Para ello, se selecciona la opción “Detached” y aparecerá el clúster recién creado en color verde (ya preparado). Para comprobar que funciona, pinchamos sobre la primera celda que contiene código y lo ejecutamos. Para ello, o bien se pincha sobre el símbolo de *play* que hay a la derecha de la celda, o bien se pulsan las teclas SHIFT + ENTER. A partir de este punto, se debe ejecutar el *notebook* siguiendo uno a uno los pasos descritos.

Los puntos, de forma breve, que se tratan en este *notebook* son los siguientes:

- Importar los datos insertados en apartados anteriores, dentro de variables del *notebook.*
- Realizar una primera aproximación utilizando la librería de Python Pandas.
- Cargar los datos anteriores utilizando la librería de PySpark.

- Observar los diferentes datos y realizar un ejercicio de preprocesamiento tratando los datos como un DataFrame, con Spark SQL.
- Con la librería de SparkML, aplicar diferentes algoritmos de *machine learning* sobre los DataFrames obtenidos gracias a Spark SQL.
- De forma análoga, realizar una implementación similar a la utilizada en el apartado anterior, solo que utilizando un apartado que se encuentra dentro de SparkML, llamado Pipeline.

Es importante que se siga el *notebook* ya mencionado para lograr los objetivos de esta unidad.

Resumen

En esta unidad práctica, se ha desarrollado un proceso de analítica escalable completo en el que se ha conseguido:

- Obtener un *dataset* con grandes cantidades de datos, de cierta relevancia y saber dónde buscar otros de diferentes términos.
- Utilizar una plataforma donde poder ejecutar código escalable de Apache Spark con las diferentes herramientas que esta ofrece.
- Ingestar el *dataset* obtenido en formato tabular de dos formas diferentes.
- Preprocesar y analizar el *dataset,* preparándolo para futuras aplicaciones de *machine learning.*
- Aplicar diferentes técnicas de *machine learning* utilizando librerías de Apache Spark para la escalabilidad, y hacerlo de dos formas: utilizando flujos (*pipelines*) y sin flujos.
- Comprender las cuatro fases de analítica que se pueden aplicar:
  - División de los datos en *train-test.*
  - Entrenamiento de los modelos con el *set* de *train.*
  - Predicción usando los modelos en el *set* de *test.*
  - Evaluación de los modelos, aplicando diferentes métricas en función de cada uno de ellos.

Ejercicios

Caso práctico

Una vez revisado el contenido teórico, el alumno deberá abrir el *notebook* *AnaliticaEscalablePySparkEjercicios.dbc*, que se puede descargar en formato .dbc en este enlace o en HTML. Este *notebook* contiene una serie de 18 preguntas que no precisan muchas líneas de código, pero que deben escribirse y ejecutarse para su evaluación.

En el *notebook* de contenido teórico facilitado al inicio de la unidad y abordado en el apartado VI ("Subir el caso práctico y ejecutarlo"), se realizaba cierto procesamiento de los DataFrames con SparkSQL; después, se aplicaba Spark ML para los modelos de Kmeans y regresión lineal; y, por último, se volvían a implementar, pero utilizando *pipelines*. Ahora, en estos ejercicios, utilizando algunos fragmentos de la parte práctica, se solicita:

- Algunas utilidades de Spark SQL, como listar los esquemas de un DataFrame o realizar consultas agrupadas.
- Implementar, utilizando Spark ML, los árboles de decisión.
- De manera análoga, se implementarán esos árboles de decisión utilizando *pipelines*.

La **evaluación** se llevará a cabo sobre el conjunto de los 18 ejercicios, todos con la misma puntuación (1/18).

Condiciones de entrega

Se realizará todo el ejercicio desde un *notebook* desde Databricks, mediante el cual se contestará a las 18 preguntas, se incluirá el código necesario para la resolución de cada una de ellas y, además, se documentará desde el propio notebook mediante comentarios en Markdown.

Se recuerda: al igual que en el resto de los módulos, una vez superadas todas las unidades aparecerá la unidad "Evaluación final", en la que se volverán a encontrar todos los enunciados, así como las condiciones de entrega de todos los ejercicios.

Recursos

Bibliografía

- **Algoritmos propios de Apache Spark ML** :
  - Apache Spark, “Clustering”, **https://spark.apache.org/docs/2.2.1/ml-clustering.html**
  - Apache Spark, “Decision tree classifier”, **https://spark.apache.org/docs/latest/ml-classification-regression.html#decision-tree-classifier**
  - Apache Spark, “StringIndexer”, **https://spark.apache.org/docs/2.2.1/ml-features.html#stringindexer**
  - Apache Spark, “Linear regression”, **https://spark.apache.org/docs/latest/ml-classification-regression.html#linear-regression**
  - Apache Spark, “ML Pipelines”, **https://spark.apache.org/docs/2.2.1/ml-pipeline.html**
- **Apache Spark** :
  - Apache Spark, **https://spark.apache.org**
  - “Apache Spark”, en Wikipedia, s. f., [En línea] URL disponible en: **https://en.wikipedia.org/wiki/Apache_Spark**
  - Apache Spark, “Spark Overview”, **https://spark.apache.org/docs/latest/**
- **Componentes de Apache Spark** :
  - Apache Spark, *Spark SQL, DataFrames and Datasets Guide*, **https://spark.apache.org/docs/latest/sql-programming-guide.html**
  - Apache Spark, *Structured Streaming Programming Guide,* **https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html**
  - Apache Spark, *Spark Streaming Programming Guide,* **https://spark.apache.org/docs/latest/streaming-programming-guide.html**
  - Apache Spark, *Machine Learning Library (MLlib) Guide*, **https://spark.apache.org/docs/latest/ml-guide.html**
  - Apache Spark, *GraphX Programming Guide,* **https://spark.apache.org/docs/latest/graphx-programming-guide.html**
  - Apache Spark, *SparkR (R on Spark),* **https://spark.apache.org/docs/latest/sparkr.html**
- **Databricks** :
  - Databricks, **https://databricks.com**
  - “Databricks”, en Wikipedia, s. f., [En línea] URL disponible en: **https://en.wikipedia.org/wiki/Databricks**
  - Community Cloud Databricks, **https://community.cloud.databricks.com/**
- **Kaggle** :
  - Kaggle, **https://www.kaggle.com**
  - “Kaggle”, en Wikipedia, s. f., [En línea] URL disponible en:**https://en.wikipedia.org/wiki/Kaggle**

Glosario.

- **Apache spark**: Framework de computación paralela que puede programarse en diferentes lenguajes y con diferentes componentes bajo un mismo core o núcleo.

- **Aprendizaje no supervisado**: Aplicación de técnicas de machine learning sobre datos no etiquetados, utilizando otro tipo de métricas como distancias o densidades para generar sus modelos.

- **Aprendizaje supervisado**: Aplicación de técnicas de machine learning sobre datos que ya se encuentran etiquetados, para generar un modelo que suele utilizarse para clasificación de otros datos no etiquetados.

- **Clúster**: Conjunto de máquinas cuyo procesamiento/almacenamiento se comporta como una única unidad.

- **Databricks**: Empresa formada por los creadores de Apache Spark. Ofrece una plataforma web de gestión sencilla de clústeres y procesos ejecutando código Spark.

- **Dataframe**: En Apache Spark, consiste en un objeto representativo de un conjunto de datos en forma tabular, aunque por debajo está formado por RDD. Permite ejecutar, mediante su API, consultas sencillas parecidas a SQL.

- **Escalabilidad**: Propiedad deseable de un sistema, que indica la habilidad de un proceso de adaptarse y reaccionar sin perder calidad, o de estar preparado para trabajar con más peticiones/datos sin bajar su rendimiento.

- **Kaggle**: Plataforma web donde se encuentran publicados sets de datos bajo competiciones para encontrar el mejor modelo que se les puede aplicar.

- **Pipeline**: En Apache Spark ML, consiste en un flujo de analítica que abarca desde el preprocesamiento de los datos hasta la predicción resultante al aplicar los modelos, incluyendo un paso intermedio de definición y entrenamiento de estos modelos.

- **Rdd**: Unidad básica dentro del core de Spark, representa un dataset distribuido y tolerante a fallos.