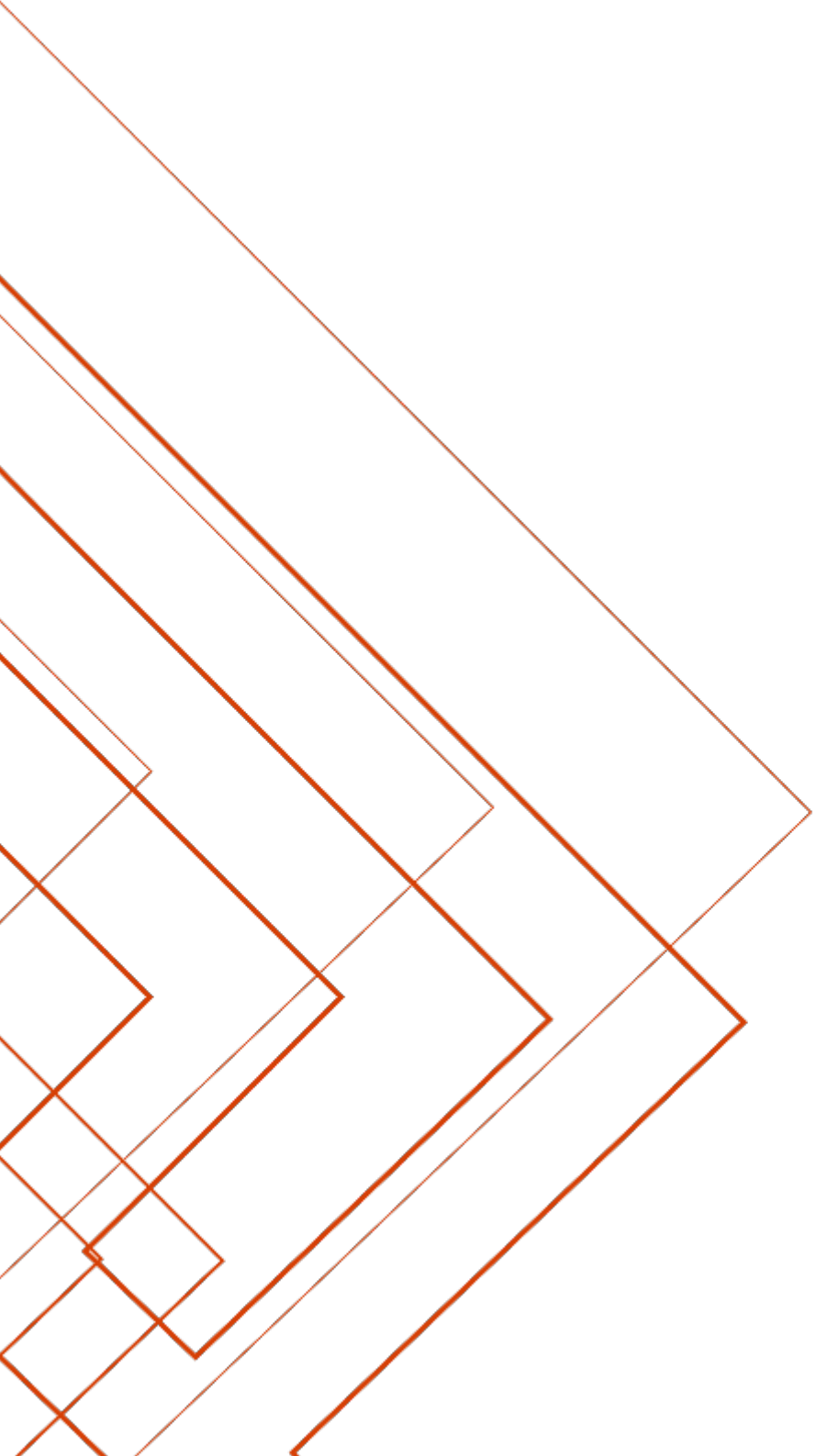


Afi Escuela

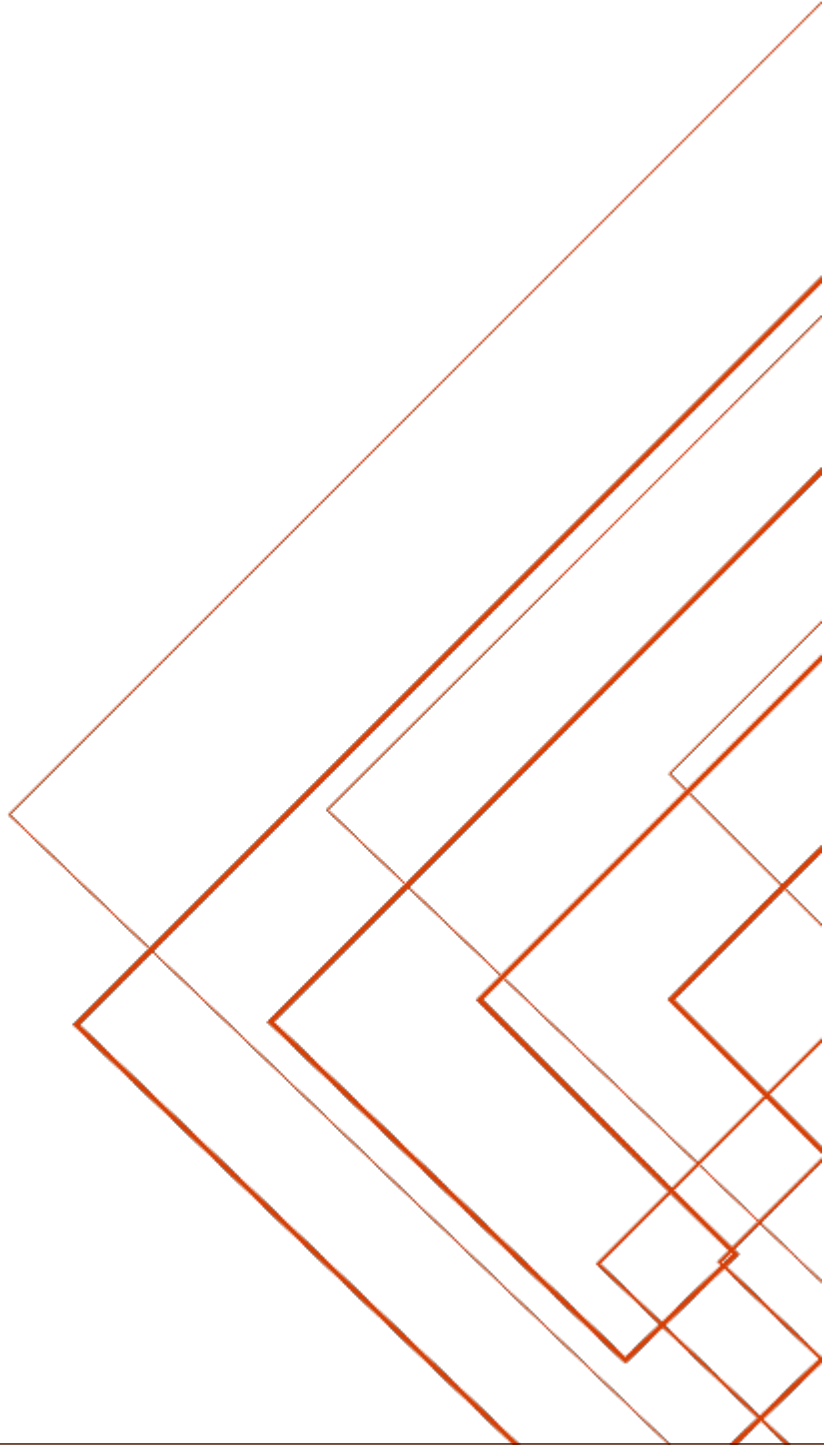
Apache Spark - ML

Máster en Data Sience y Big Data

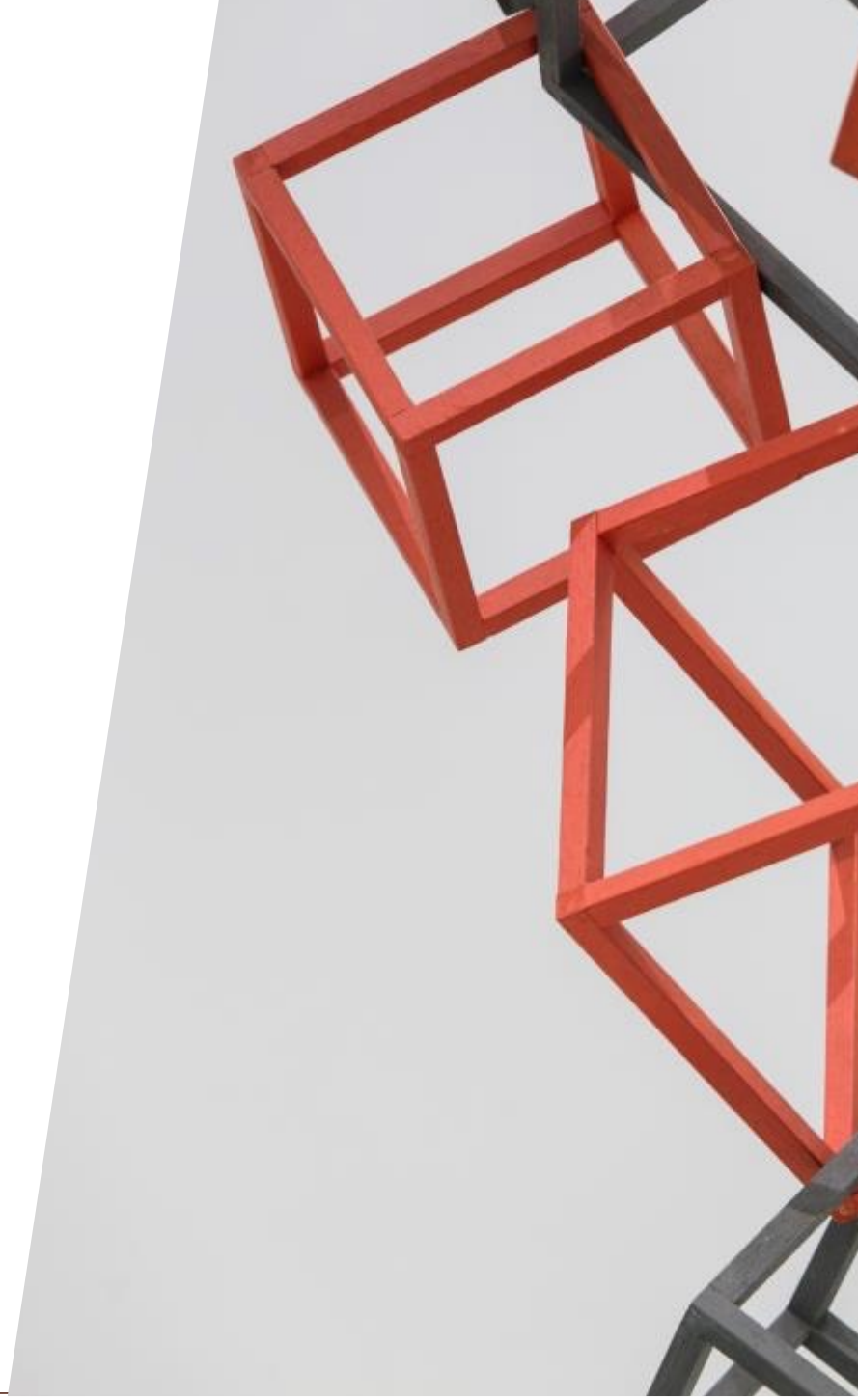
Miguel Ángel Corella
mcorella@geoblink.com

Marzo 2022

Contenido



Introducción



Spark Core + SQL, ¿es suficiente?

- Apache Spark ha dado una solución global a los principales problemas que Hadoop presentaba:
  - Velocidad, facilidad de uso, flexibilidad, homogeneidad.

- Su utilidad es clara cuando hablamos de tareas más cercanas al *data engineering:*
  - Carga de (grandes) datos, *parsing*, limpieza/filtrado de información, etc.

- Incluso, su utilidad es clara cuando hablamos de tareas de puro análisis/consulta de información:
  - Inferencia de estructura, queries sobre datos estructurados…

- Sin embargo, ambos módulos sólo dan soporte a una parte del trabajo de un DS.
  - Es difícil implementar **algunas operaciones estadísticas** sobre Spark Core (+ SQL).
  - Es difícil implementar **algoritmos de modelización** sobre Spark Core (+ SQL)

¿Qué es Apache Spark MLlib?

- Es un módulo de Apache Spark creado para facilitar la ejecución de tareas asociadas a **estadística y machine learning** sobre grandes volúmenes de información, en principio, distribuida.

- Se trata de **una capa de abstracción** ya que, internamente, la ejecución final de tareas la realiza Spark Core.

- Incluye capacidades y funcionalidades tales como:
  - Estructuras de datos: incluye nuevos tipos de datos y operaciones.
  - Estadística: estadística descriptiva, correlaciones, contraste de hipótesis…
  - Preprocesado: extracción, transformación y selección de variables.
  - Aprendizaje supervisado: regresión y clasificación.
  - Aprendizaje no supervisado: clustering.
  - Sistemas de recomendación.
  - Evaluación y selección de modelos.
  - Construcción de pipelines de datos + modelización.

RDDs vs. DataFrames

- En un principio, Spark incluyo MLLib que habilitaba la ejecución de un conjunto de tareas asociada a **estadística y machine learning** directamente sobre RDDs.

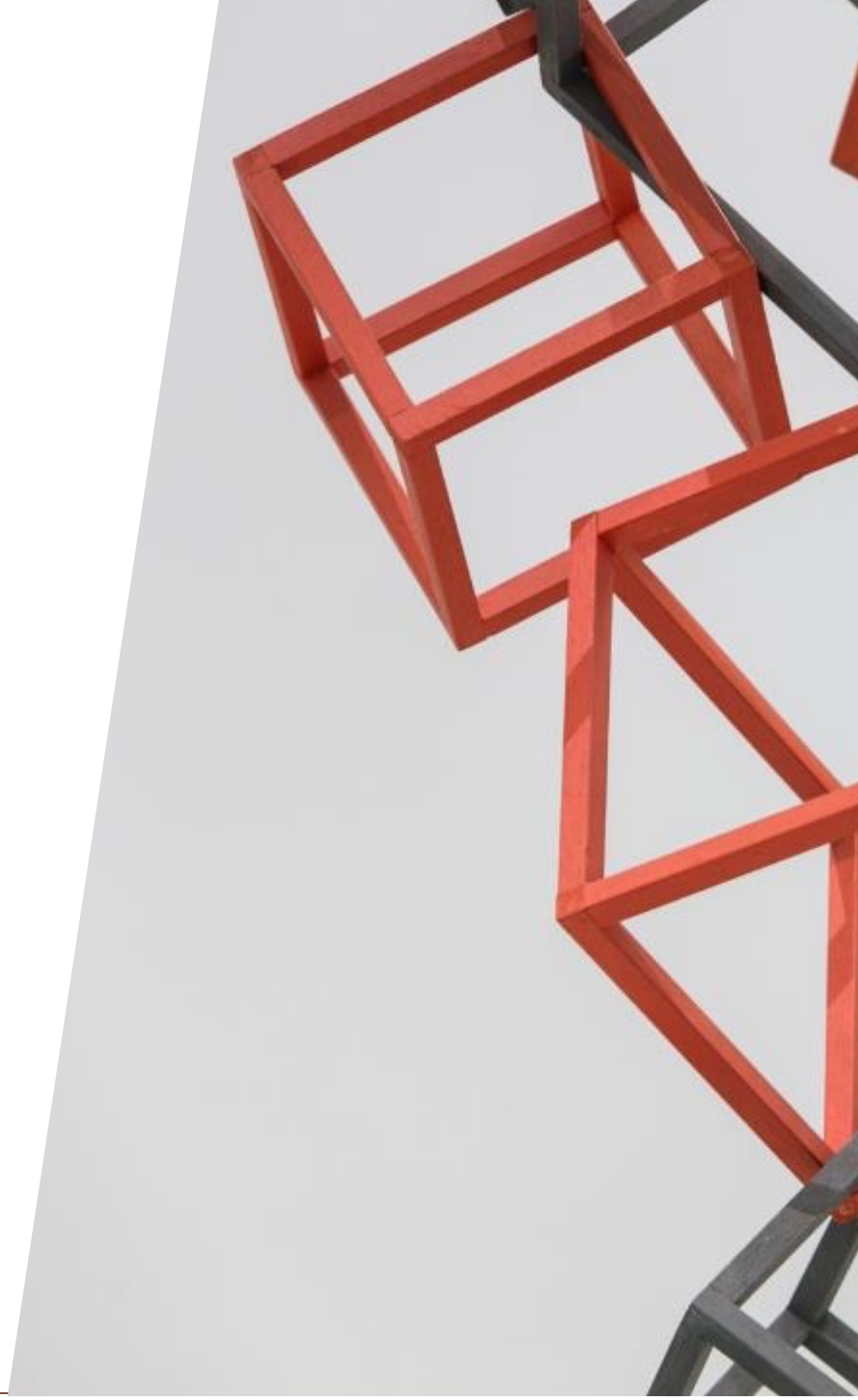
- Sin embargo, la mayor parte de estas tareas se aplican, generalmente, sobre **conjuntos de datos que tienen una estructura** bien definida, es decir, DataFrames.

- Esto hace que, desde muy temprano, se disponga de **dos versiones** de la librería:
  - **RDD-based API:** basada en Spark Core, con RDDs como estructura básica.
  - **DataFrame-based API:** basada en Spark SQL, con DataFrames como estructura básica.

- Aunque ambas siguen vigentes, **DataFrame-based API** se considera la versión **oficial y principal** del módulo, mientras que RDD-based API ya no está en desarrollo.

- Por la estructura de los paquetes internos de la librería, a la versión DataFrame-based API se le conoce también como **Spark ML**.

Estructuras de datos



Estructuras de datos

- Spark ML está implementado, principalmente, sobre **estructuras de datos numéricas**, del mismo modo que scikit-learn está implementado sobre arrays numéricos de numpy.

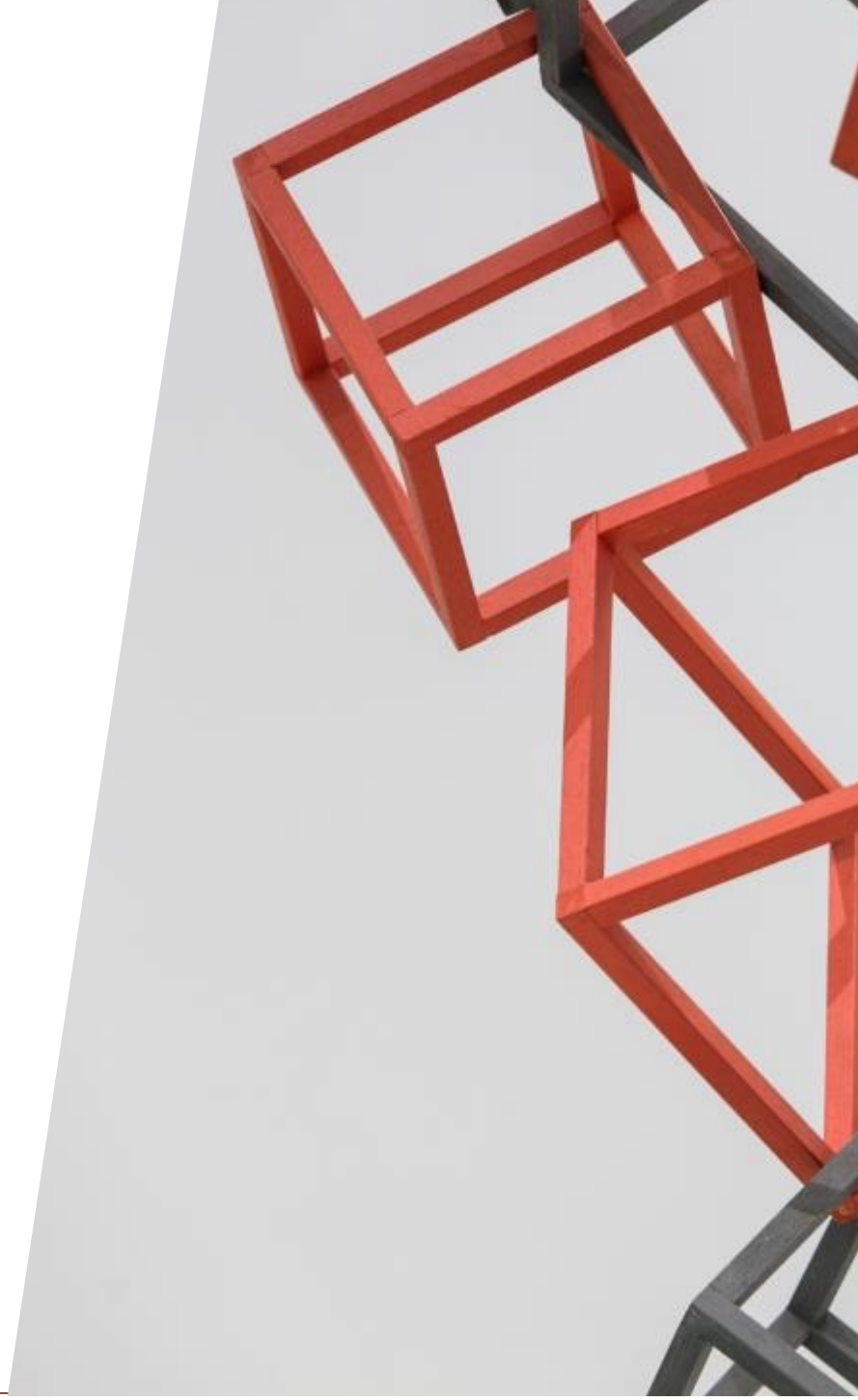
- Spark ML añade dos nuevos tipos de datos (que pueden ser asignados a una columna de un DataFrame):
  - Vectores: estructuras unidimensionales de valores numéricos (coma flotante).
  - Matrices: estructuras bidimensionales de valores numéricos (coma flotante).

- Aunque partamos de **DataFrames con múltiples columnas con tipos distintos**, la mayor parte de las funcionalidades de **Spark ML** requerirá la conversión de **cada fila** de datos a **vectores numéricos**.

Juguemos un poco...

Spark ML – Data structures.ipynb

Análisis exploratorio / Estadística



Análisis exploratorio / Estadística

- En su inicio, Spark MLlib empezó a incorporar funciones que facilitaban la realización de operaciones estadísticas sobre los datos (p.e. resúmenes descriptivos).

- El cambio de filosofía al uso de Spark SQL y DataFrames como estructura básica frenó considerablemente dicho desarrollo ya que se incluían allí las principales funciones.

- Sin embargo, el subpaquete de estadística se sigue manteniendo principalmente para dar cobertura a:
  - Cálculo de correlaciones.
  - Contraste de hipótesis.

- Es importante destacar, que Spark ML (al igual que hace scikit-learn) se centra en "flujos de modelización" dejando de lado la fase de análisis exploratorio.

Flujo de modelización

Cualquier flujo de modelización implementado en Spark ML sigue unas fases de trabajo muy claras (equivalentes a las de librerías como scikit-learn).

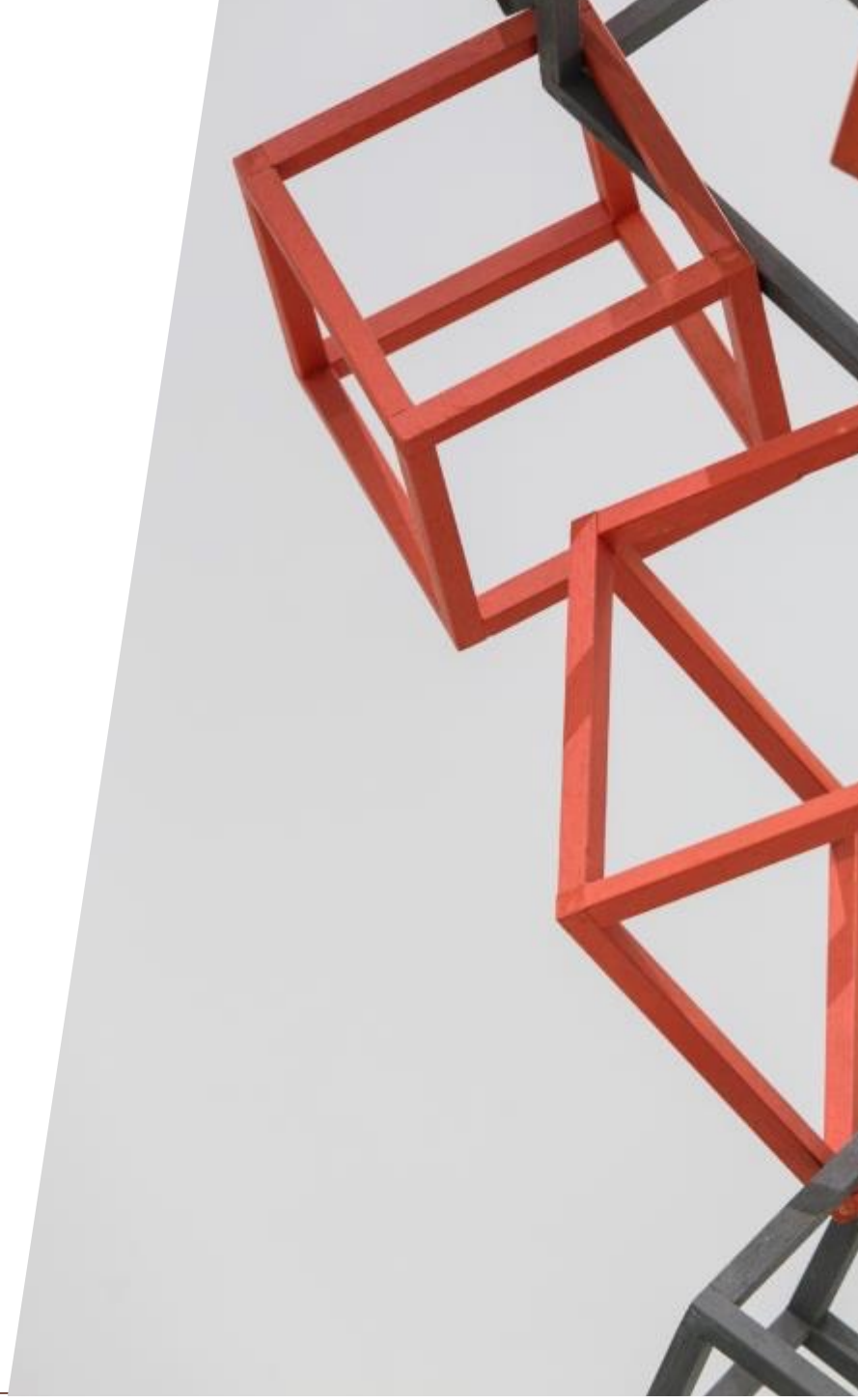
1. Carga de datos.
2. Limpieza y preparación (aplicando las funciones disponibles sobre DataFrames).
3. Adecuación de datos a Spark ML:
    1. Identificación de variables target y predictoras.
    2. Codificación y conversión de tipos de datos a numéricos.
    3. Conversión de variables predictoras a vectores.
4. Split train vs. test.
5. Configuración de modelos.
6. Entrenamiento/fit de modelos en train.
7. Predicción de modelos en test.
8. Evaluación de resultados.

Preprocesado

Juguemos un poco...

Spark ML – Exploratory analysis.ipynb

Preprocesado de información



Preprocesado de información

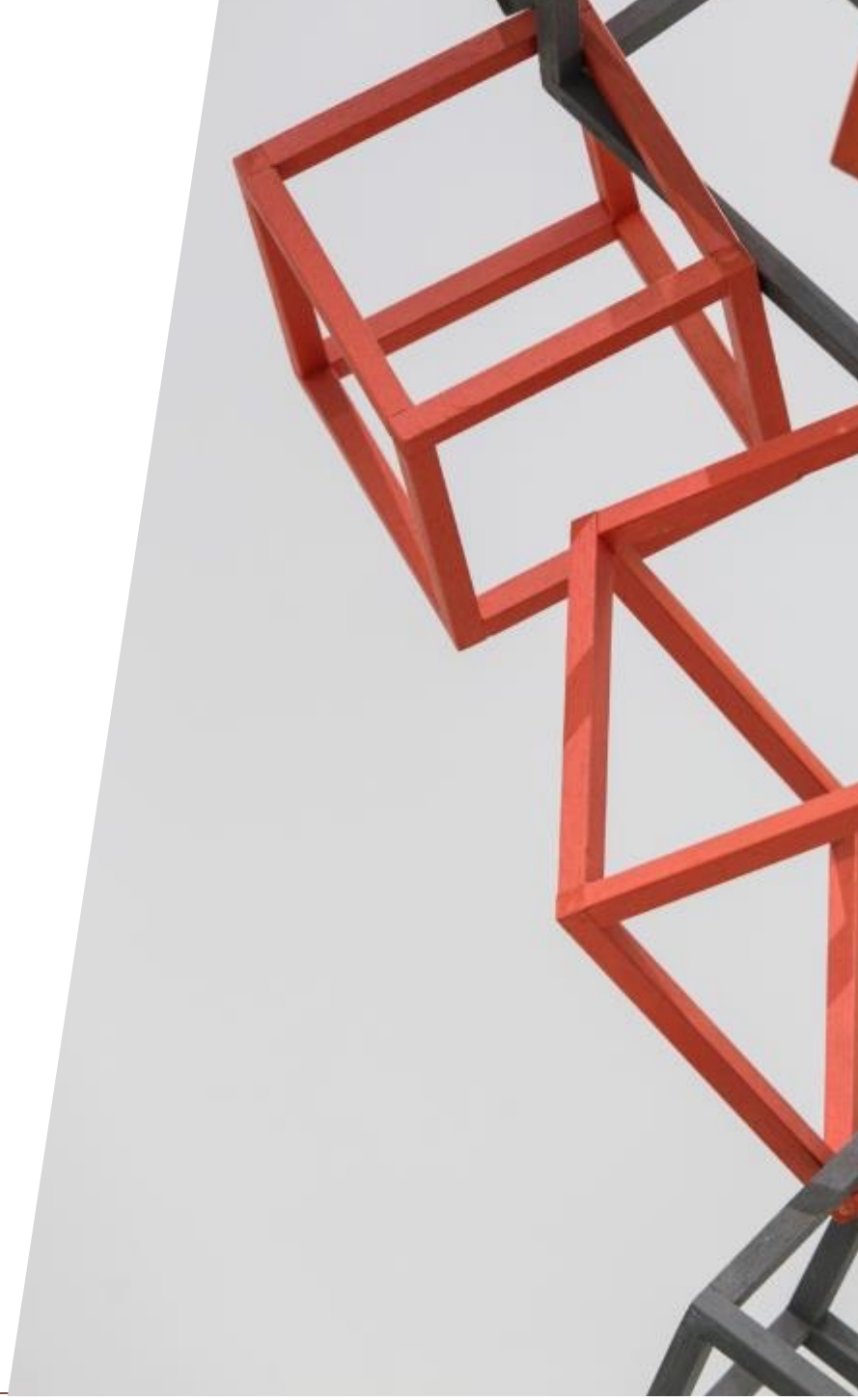
- Al igual que ocurre en scikit-learn, Spark ML impone ciertas restricciones sobre la estructura de datos que se utiliza para las tareas de modelización:
  - Todos los datos deben ser numéricos.
  - Los predictores deben estar unificados en una única columna → Vector numérico.
  - El target tiene que estar en una columna aislada → Valor numérico

- Dadas estas restricciones, Spark ML (al igual que hace scikit-learn) nos ofrece un conjunto MUY amplio de funciones que permiten:
  - Extracción de variables: principalmente orientado a text mining.
  - Codificación/conversión de variables categóricas y numéricas.
  - Agregación y conversión de columnas en vectores.
  - …

Juguemos un poco...

Spark ML – Preprocessing.ipynb

Aprendizaje automático



Aprendizaje automático

- Una vez disponemos de los datos en el formato adecuado, Spark ML pone a nuestra disposición una **amplio abanico de algoritmos de machine learning.**
  - Aprendizaje supervisado.
  - Aprendizaje no supervisado.
  - Sistemas de recomendación.

- Es importante entender que, aunque esto habilita el **entrenamiento de modelos con cantidades ingentes de datos**, el catálogo de algoritmos no es tan amplio (por el momento) como en otras librerías (e.g. scikit-learn, caret, etc.).

- La decisión entre **Spark ML vs. muestreo de datos + librarías locales**, dependerá del caso de uso y del valor/pérdida asociado al muestreo de datos.

Aprendizaje automático - Regresión

- En términos de modelos de regresión, Spark ML pone a nuestra disposición:
  - Regresión lineal + regularización (Lasso, Ridge, ElasticNet.).
  - Árboles de decisión.
  - Random Forest.
  - Gradient Boosted Trees.
  - …

- Y al mismo tiempo nos da una **herramienta común de evaluación** de resultados que nos da acceso al cálculo de métricas de evaluación sobre los resultados de una regresión:
  - R^2, RMSE, MSE, MAE

Juguemos un poco...

Spark ML – Regression.ipynb

Aprendizaje automático - Clasificación

- En términos de modelos de clasificación, Spark ML pone a nuestra disposición (tanto para clasificación binaria como multiclase):
  - Regresión logística + regularización (Lasso, Ridge, ElasticNet.).
  - Árboles de decisión.
  - Random Forest.
  - Gradient Boosted Trees.
  - Naive Bayes.
  - Perceptrón multicapa.
  - Support Vector Machines.
  - …
- Y al mismo tiempo nos da una **herramienta común de evaluación** de resultados que nos da acceso al cálculo de métricas de evaluación sobre los resultados de una clasificación:
  - AUC, AUPR…

Juguemos un poco...

Spark ML – Classification.ipynb

Aprendizaje automático - Clustering

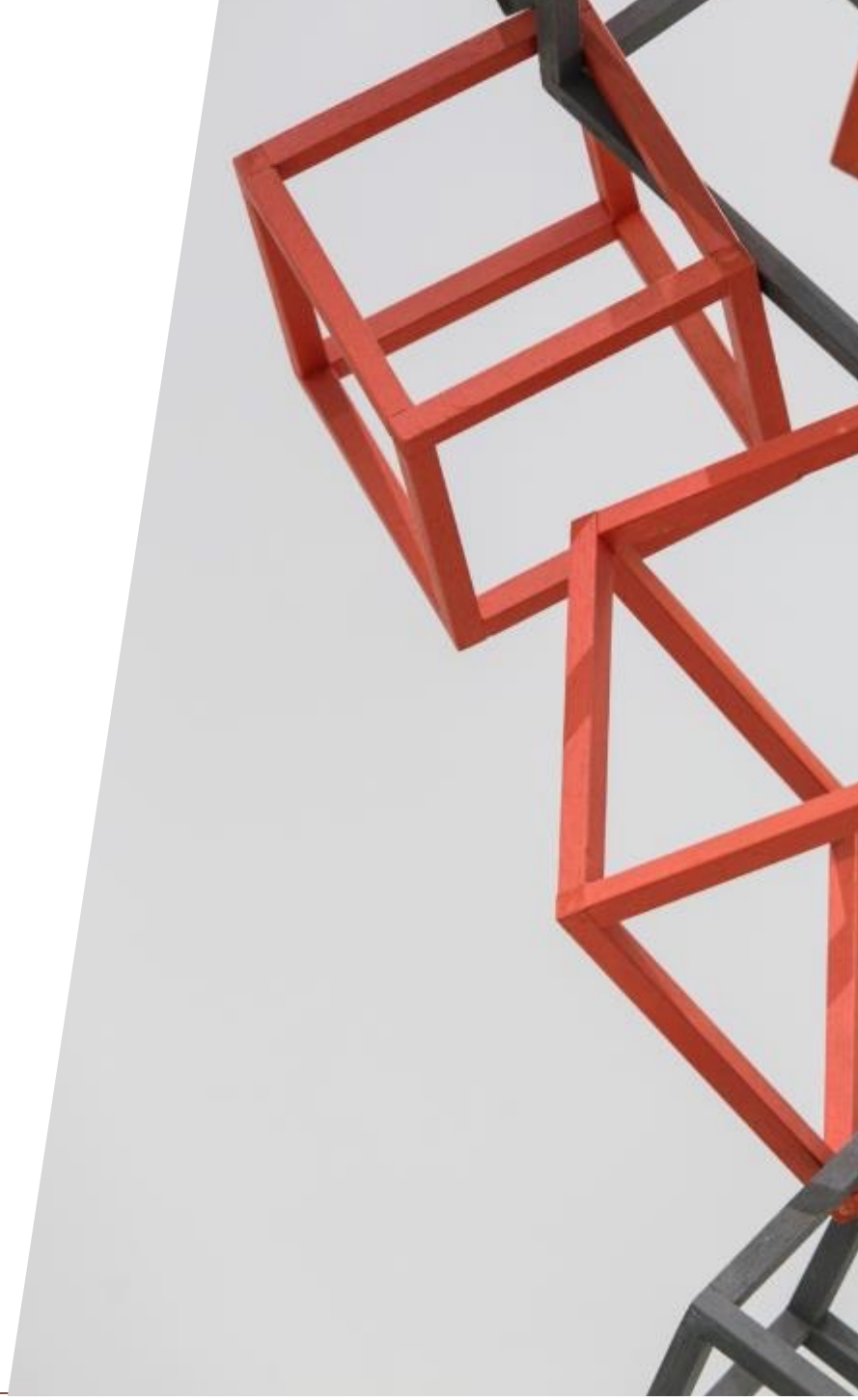
- En términos de modelos de clustering, Spark ML pone a nuestra disposición:
  - K-Means.
  - Latent Drichlet Allocation (LDA).
  - Gaussian Mixture Model (GMM).
  - …

- Y al mismo tiempo nos da una **herramienta común de evaluación** de resultados que nos da acceso al cálculo de métricas de evaluación sobre los resultados de una clasificación:
  - Silhouette…

Juguemos un poco...

Spark ML – Clustering.ipynb

Selección y tuning de modelos



Selección y tuning de modelos

- Cualquier algoritmo de machine learning incluye un conjunto (relativamente amplio) de posibles parámetros.
  - Regularización en regresión lineal y logística.
  - Profundidad en árboles de decisión.
  - Número de árboles en random forest.
  - Número de clusters en k-means.
  - …
- Spark ML nos ofrece, al igual que lo hace scikit-learn, herramientas para facilitar el proceso de encontrar el mejor modelo posible. Estas herramientas requieren:
  - Un estimador: cualquiera de los algoritmos de aprendizaje automático disponibles.
  - Un evaluador: asociado al algoritmo de aprendizaje automático seleccionado.
  - Un conjunto de parámetros y valores a testear para encontrar la mejor combinación.

Selección y tuning de modelos

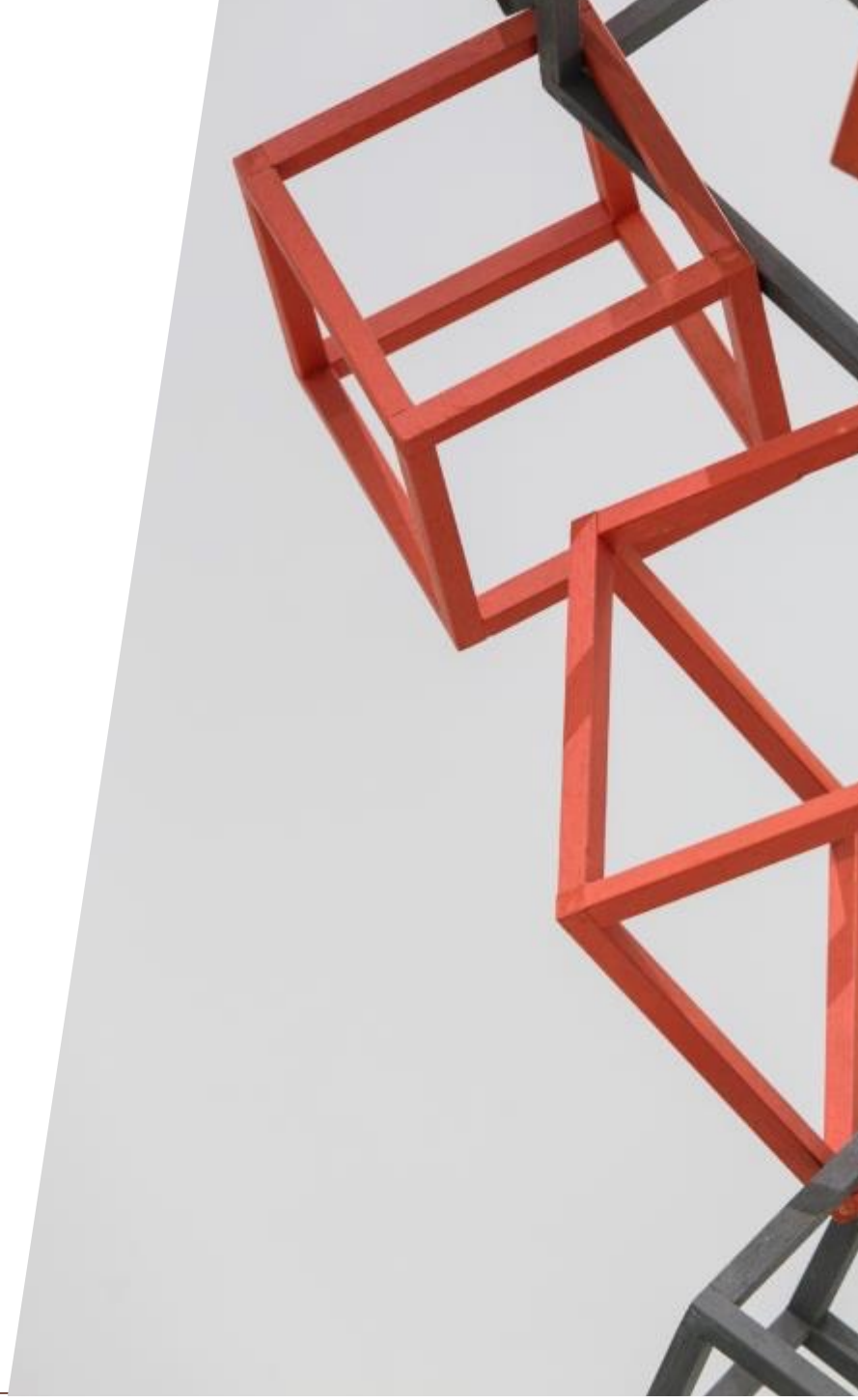
- El proceso que se seguirá para la optimización de parámetros será el siguiente:
  - Realizar un split entre train y test.
  - Crear un estimador con una combinación de parámetros (del espacio configurado).
  - Entrenar el estimador con el split de train y predecir sobre el de test.
  - Evaluar los resultados con el evaluador configurado.
  - Seleccionar el mejor modelo posible una vez probadas todas las opciones (del espacio configurado).

- En función de la parametrización del split train-test que queramos hacer (y el tiempo que queramos esperar a los resultados) tenemos dos posibilidades:
  - TrainValidationSplit: split único → Más rápido, menos seguridad sobre generalización.
  - CrossValidator: k splits → Más lento, más seguridad sobre generalización.

Juguemos un poco...

Spark ML – Tuning.ipynb

Pipelines



Pipelines

- Hasta ahora, hemos visto un conjunto de elementos que pueden ser configurados de forma individual y combinados para alcanzar una solución óptima.

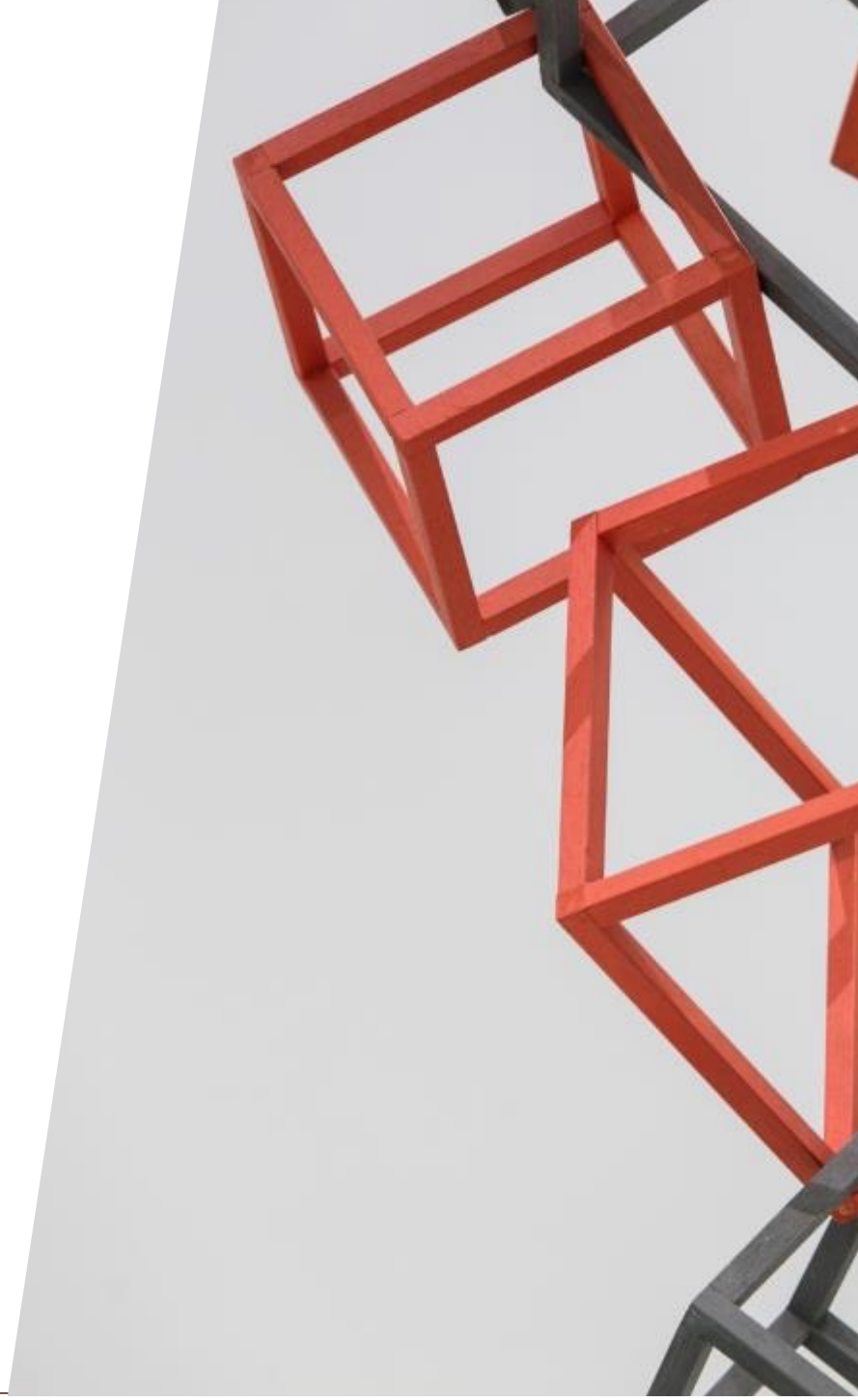
- Sin embargo, tratarlos de forma individual tiene ciertos problemas:
  - Configuración "manual" de combinaciones de parámetros.
  - Establecimiento de orden/prioridad a la hora de optimizar.
  - Data leakage a la hora de hacer Cross Validation.
  - ...

- Spark ML, igual que *scikit-learn*, nos permite definir un proceso completo de datos y modelización (conjunto de pasos) y su utilización como si de un modelo más se tratase:
  - Un único método fit para todos los pasos.
  - Un único método transform para todos los pasos.
  - Un único conjunto de hiperparámetros (unión de los parámetros de cada paso).

Juguemos un poco...

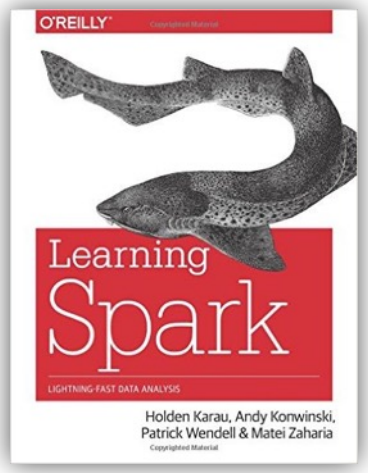
Spark ML – Pipelines.ipynb

Referencias



Referencias

- Documentación oficial:
  - https://spark.apache.org
- Tutoriales online:
  - https://www.tutorialspoint.com/apache_spark/
- Libros:



Afi Escuela