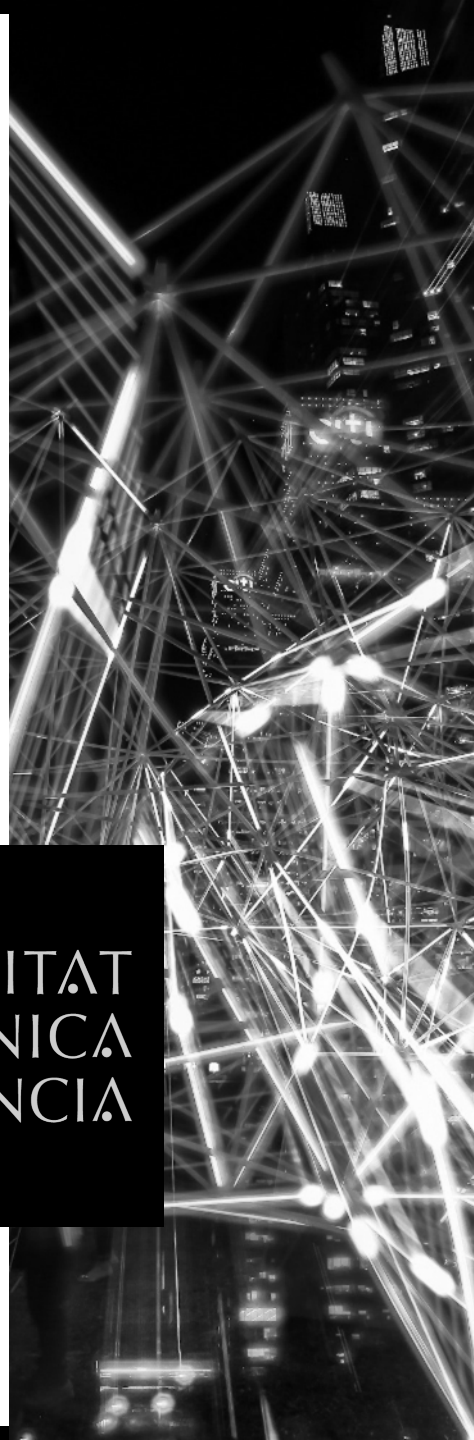
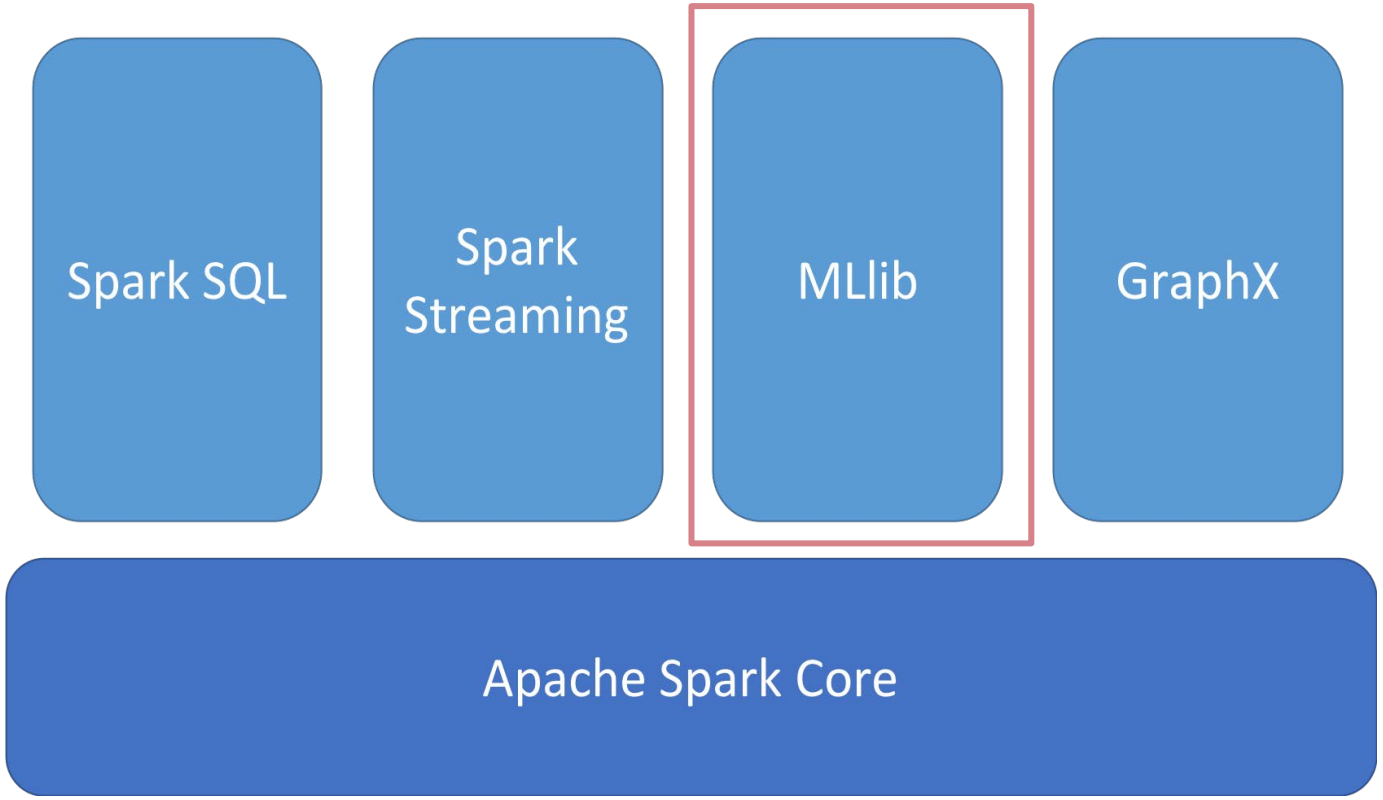


MLLIB

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

APIS DISPONIBLES EN SPARK

APIs generales disponibles

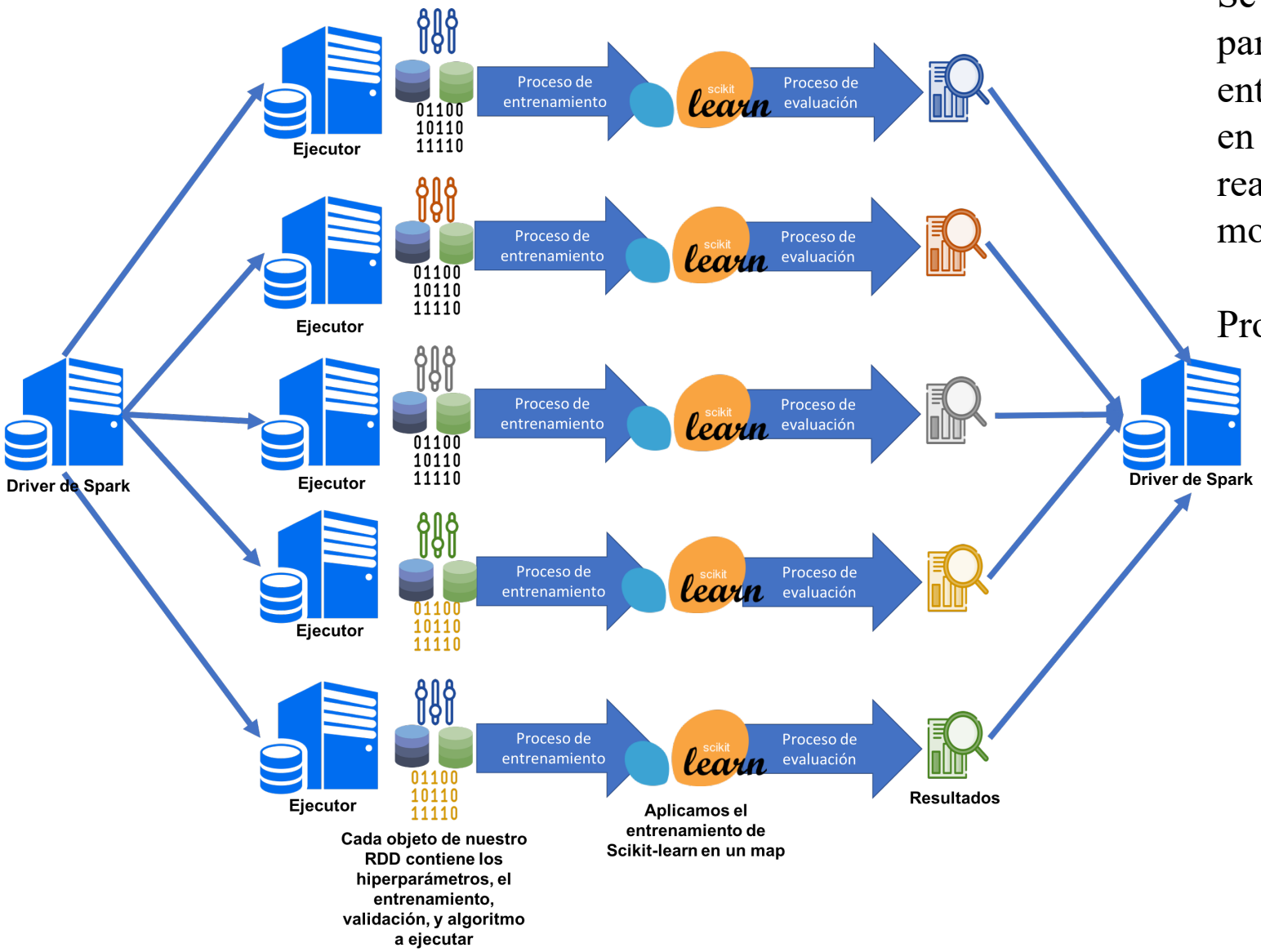


- MLLib es la API centrada en tareas de aprendizaje automático
- Permite la construcción de grandes modelos en base a colecciones muy grandes de datos
- Se encuentra en Java, Scala y Python

APRENDIZAJE AUTOMÁTICO EN SPARK

¿Cómo se puede llevar a cabo?

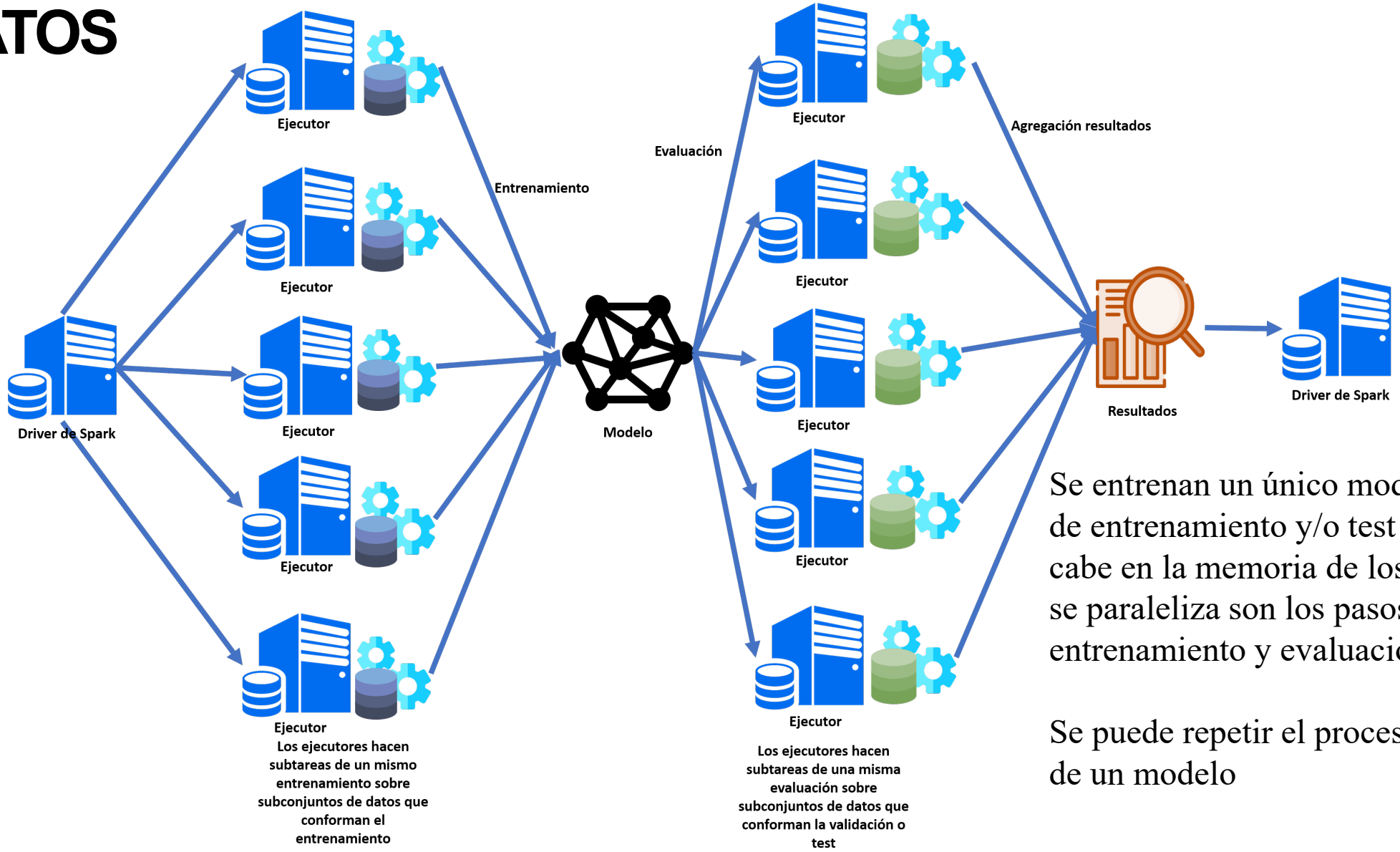
MÉTODO 1: PARALELIZACIÓN DE MÚLTIPLES ENTRENAMIENTOS



Se entrenan y evalúan muchos modelos en paralelo, ya que el conjunto de datos de entrenamiento y validación es pequeño, y cabe en la memoria de los ejecutores. Cada ejecutor realiza el entrenamiento de uno o varios modelos.

Programable con la API de RDDs.

MÉTODO 2: PARALELIZACIÓN DE UN ENTRENAMIENTO CON GRAN VOLUMEN DE DATOS



Se entrenan un único modelo con un conjunto de entrenamiento y/o test muy grande que no cabe en la memoria de los ejecutores. Lo que se paraleliza son los pasos a realizar para el entrenamiento y evaluación de dicho modelo.

Se puede repetir el proceso para entrenar más de un modelo

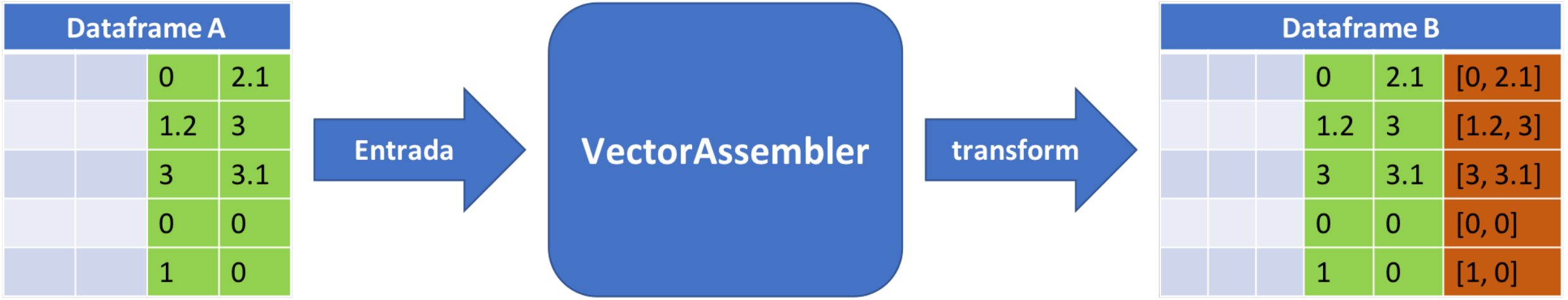
ENTRENAMIENTO DE GRANDES MODELOS EN MLLIB

- MLLib se especializa en el entrenamiento de modelos que requieren grandes cantidades de entrenamiento. La paralelización es a nivel de los pasos que se realizan para entrenar un modelo (método 2).
- Aunque existe la API para trabajar con MLLib usando RDDs, desde hace unas cuantas versiones la abstracción de datos recomendada es la de **Dataframe**.
- En MLLib hay dos componentes principales con los que podemos trabajar:
  - **Transformador**
  - **Estimador**

TRANSFORMADORES

Un transformador es una componente que toma un **Dataframe como entrada** y realiza una transformación, generando un **nuevo Dataframe como salida** con la transformación realizada.

TRANSFORMADORES: EJEMPLO VECTOR ASSEMBLER



TRANSFORMADORES: EJEMPLO VECTOR ASSEMBLER

```
from pyspark.ml.feature import VectorAssembler


assembler = VectorAssembler(inputCols=["sepal_length","sepal_width",
"petal_length","petal_width"], outputCol="features")


vectorized_df = assembler.transform(train_df)

vectorized_df.show()
```

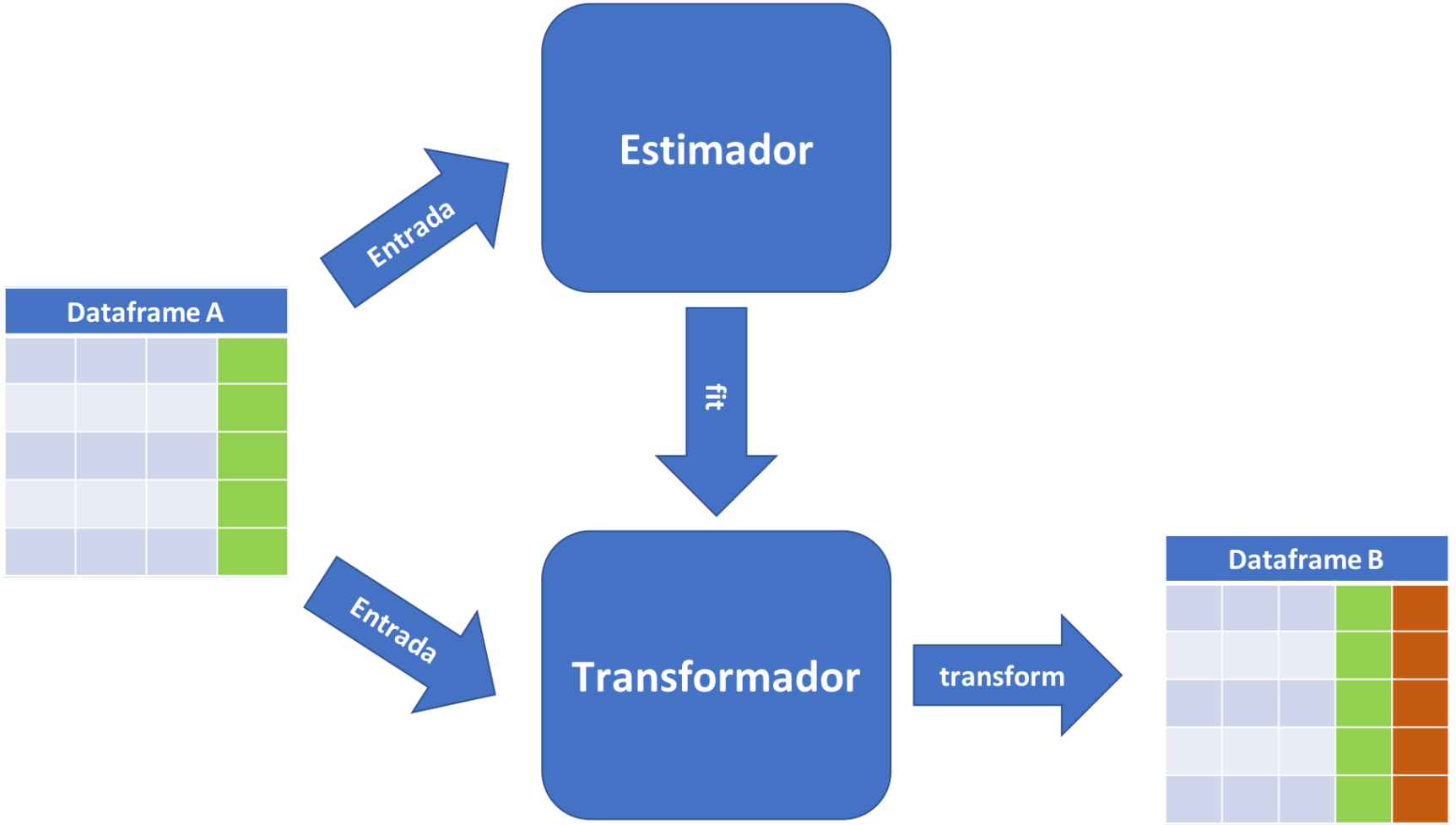
TRANSFORMADORES EN MLLIB

Nombre	Descripción	Especializado en...
Tokenizer	Toma un texto y lo separa, típicamente en palabras de acuerdo a alguna expresión regular	Texto
StopWordRemover	Toma un vector de palabras o términos y elimina todos aquellos que no tienen un significado especial (lista pasada por parámetro)	Texto
Binarizer	Toma una columna Double y la transforma a binaria de acuerdo a un umbral (x>umbral → cierto)	Discretizar
Interaction	Toma dos columnas de tipo vector numérico y genera un nuevo Dataframe con una columna por cada multiplicación de elementos de ambos vectores	Numérico

TRANSFORMADORES EN MLLIB (II)

Nombre	Descripción	Especializado en...
Normalizer	Toma una columna de tipo vector numérico y genera un Dataframe con una nueva columna donde los vectores han sido normalizados para tener norma unitaria	Numérico
VectorAssembler	Toma varias columnas de tipo numérico (vector o no) y genera un Dataframe con una nueva columna de tipo vector numérico resultado de juntar las columnas especificadas.	Numérico
Bucketizer	Toma una columna numérica y unas divisiones del dominio (e.g., -inf, -3, 45, 120, inf) y genera un nuevo Dataframe con una columna que es el resultado de indicar el índice en las divisiones donde caen los valores de la columna	Discretizar
OneHotEncoderEstimator	Toma una columna con valores discretos y genera un nuevo Dataframe con una columna de tipo vector numérico donde hay tantas componentes como posibles valores en la columna original. Cada componente es binaria y vale 1 cuando el valor en la columna original es el especificado.	Discretizar

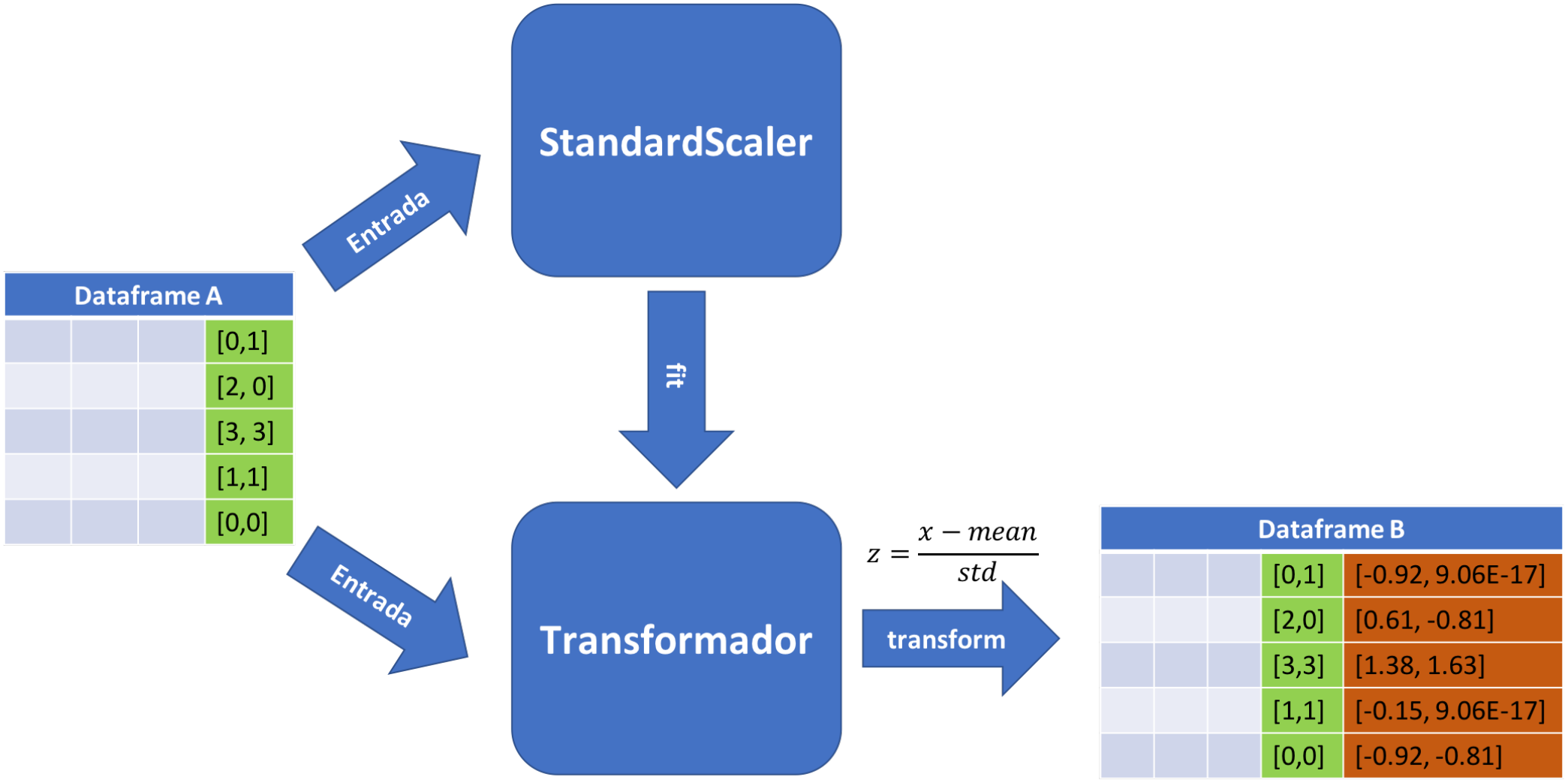
ESTIMADORES



Un estimador funciona en dos pasos. Primero **toma un Dataframe como entrada**, y en base a un proceso de ajuste, **genera un nuevo transformador**.

Posteriormente, el transformador puede emplearse para transformar el mismo u otros Dataframes.

ESTIMADORES: EJEMPLO STANDARDSCALER



ESTIMADORES: EJEMPLO STANDARD SCALER

```
from pyspark.ml.feature import VectorAssembler


assembler = VectorAssembler(inputCols=["sepal_length","sepal_width",
"petal_length","petal_width"], outputCol="features")


vectorized_df = assembler.transform(train_df)

vectorized_df.show()
```

TRANSFORMADORES: EJEMPLO VECTOR ASSEMBLER

```
from pyspark.ml.feature import StandardScaler


scaler = StandardScaler(withMean=True, withStd=True, inputCol="features",
outputCol="std_features")


fitted_scaler = scaler.fit(vectorized_df)


scaled_df = fitted_scaler.transform(vectorized_df)

scaled_df.show()
```

ESTIMADORES EN MLLIB

Nombre	Descripción	Especializado en...
TF-IDF	Transformación empleada en el procesado de textos. Construye un vector de importancias por cada palabra de un vocabulario en la descripción de un documento.	Texto
Word2Vec	Transforma un documento de texto en un vector de características definido por los términos que se utilizan en el mismo.	Texto
CountVectorizer	Cuenta el número de veces que aparece cada término en un texto.	Texto
PCA	Toma una columna de tipo vector numérico y realiza transformaciones ortogonales para transformar el vector en uno con menos componentes.	Numérico

ESTIMADORES EN MLLIB (II)

Nombre	Descripción	Especializado en...
StandardScaler	Toma una columna de tipo vector numérico y genera un Dataframe donde a cada componente se le ha sustraído su media y se le ha dividido por su desviación típica. Transformación a N(0,1)	Numérico
MinMaxScaler	Toma una columna de tipo vector numérico y genera un Dataframe con una nueva columna de tipo vector numérico donde cada componente del vector ha sido transformada normalmente a una escala [0,1]	Numérico
MaxAbsScaler	Toma una columna de tipo vector numérico y genera un Dataframe con una nueva columna de tipo vector numérico donde cada componente del vector ha sido transformada normalmente a una escala [-1,1]	Numérico

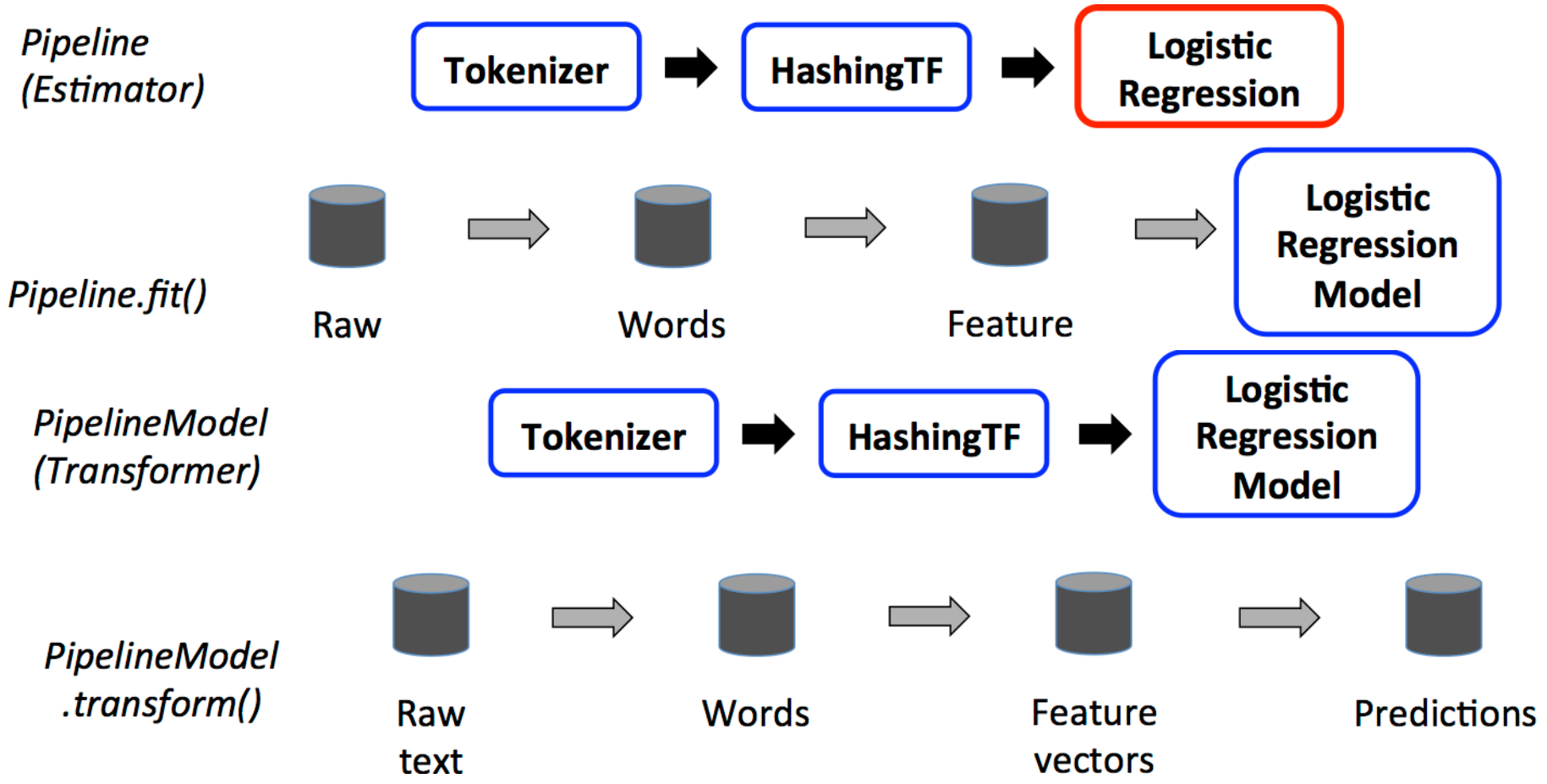
ESTIMADORES EN MLLIB (III)

Nombre	Descripción	Especializado en...
StringIndexer	Toma una columna que contiene valores de tipo cadena de texto y genera un Dataframe con una nueva columna de tipo numérico donde a cada cadena de texto diferente se le ha asignado un identificador	Discretizar

Más transformadores y estimadores en:
https://spark.apache.org/docs/latest/ml-features.html

PIPELINES DE ESTIMADORES Y TRANSFORMADORES

- En los procesos de aprendizaje automático es muy frecuente que tengamos que hacer una serie de transformaciones sobre el conjunto de entrenamiento y test para facilitar el aprendizaje de los algoritmos. Estas transformaciones suelen ser secuenciales (i.e., primero una, sobre el resultado de esta otra, y así) hasta alcanzar el resultado deseado. En MLLib, los algoritmos de aprendizaje son estimadores, que se ajustan con el conjunto de datos y crean un transformador que añade la predicción al conjunto de datos original.



EJEMPLO PIPELINE

```
from pyspark.ml import Pipeline

assembler =
VectorAssembler(inputCols=["sepal_length","sepal_width",
"petal_length","petal_width"], outputCol="features")

scaler = StandardScaler(withMean=True, withStd=True,
inputCol="features", outputCol="std_features")

indexer = StringIndexer(inputCol="variety",outputCol="class")

network =
MultilayerPerceptronClassifier(featuresCol="std_features",labelCol=
"class", predictionCol="prediction", layers=[4, 4, 3])


pipeline = Pipeline(stages=[assembler, scaler, indexer, network])

fitted_model = pipeline.fit(train_df)


test_predictions_df = fitted_model.transform(test_df)

test_predictions_df.show()
```

ALGORITMOS DISPONIBLES

Tipo	Algoritmos	Tipo	Algoritmos
Regresión	Regresión lineal Modelos de regresión generalizados Regresión basada en árboles de decisión Regresión basada en Random Forest Regresión basada en árboles gradient boosted Regresión de supervivencia Regresión isotónica	Recomendación	Filtrado colaborativo basado en descomposición en matrices
Clasificación	Regresión logística Árboles de decisión Random Forest Árbol Gradient Boosted Máquina de Soporte Vectorial Naïve Bayes Perceptron multicapa	Reglas de asociación	FP-Growth PrefixSpan
Clustering	K-Means Latent Dirichlet Allocation (LDA) Bissecting K-Means Gaussian Mixture Model Power Iteration Clustering		

EJEMPLO VALIDACIÓN CRUZADA Y GRID SEARCH

```
assembler = VectorAssembler(inputCols=["sepal_length","sepal_width", "petal_length","petal_width"],
outputCol="features")

scaler = StandardScaler(withMean=True, withStd=True, inputCol="features", outputCol="std_features")

indexer = StringIndexer(inputCol="variety",outputCol="class")

network = MultilayerPerceptronClassifier(featuresCol="std_features",labelCol="class",
predictionCol="prediction")


pipeline = Pipeline(stages=[assembler, scaler, indexer, network])


parameters = ParamGridBuilder().addGrid(network.layers, [ [4,4,3], [4,4,4,3], [4,3]  ] ).\

    addGrid( network.maxIter, [100, 200, 300] ).build()


evaluator = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol="class",
metricName="accuracy")

cross_validator =
CrossValidator().setEstimator(pipeline).setEvaluator(evaluator).setEstimatorParamMaps(parameters).\

    setNumFolds(5).setSeed(2021)

model = cross_validator.fit(train_df)
```

COP
UPV

CONTINUARÁ …

Víctor Sánchez Anguix ✉ vicsana1@upv.es

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

DEIOAC
VALENCIA

DEPARTAMENTO DE ESTADÍSTICA E INV. OPERAT. APLICADAS Y CALIDAD

UNIVERSITAT POLITÈCNICA DE VALÈNCIA