**PySpark and MLLib**

**Step 1: Run pyspark**

Navigate to spark dir -> bin

./pyspark

**Step 2: Working with Dataframes**

**Creating a DataFrame:**

Df = spark.read.csv(“file, Header = true);

(.csv Read as a csv / .text etc..)

**df.schema()**  : structure view

**df.printSchema()** : more readable structure view

**Df.columns** : get Columns

**df.take(5)**  : get first 5 rows

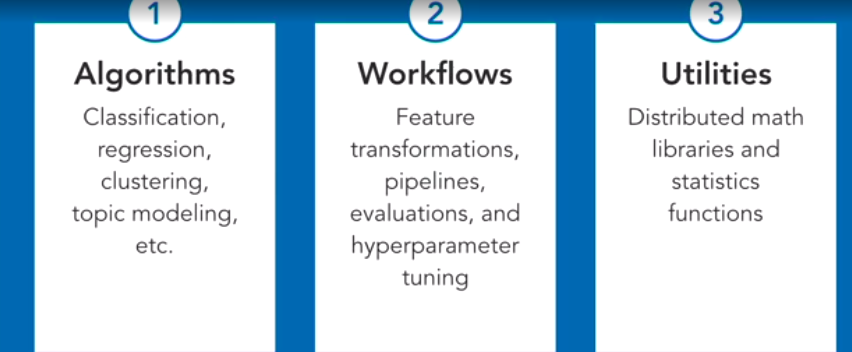
**df.count()**  : no of rows

Get a sample:

**New\_sample = df.sample(False, 0.1)**

: 10% randomly without replacement

**Step 3: Components of MLLib**

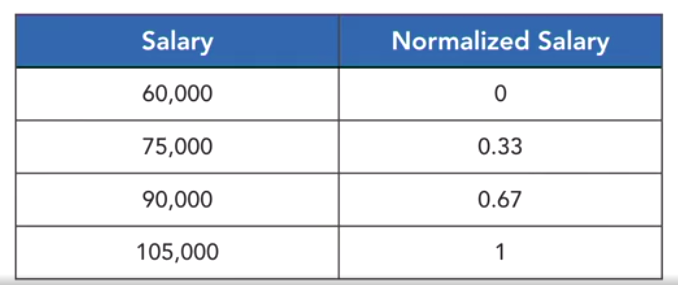


**Standardizing to aNormal Distribution**

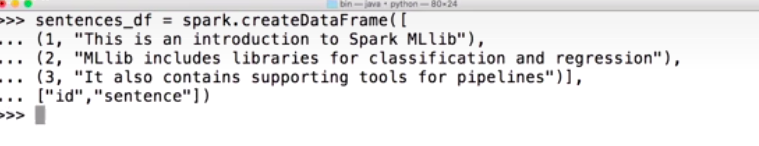
What is Normalizing?

Mapping in between 0-1 (0 as the minimum of range and 1 as the maximum of range)

ex:

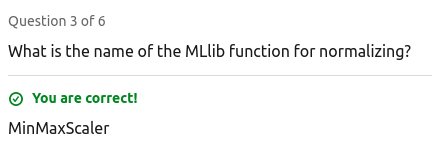


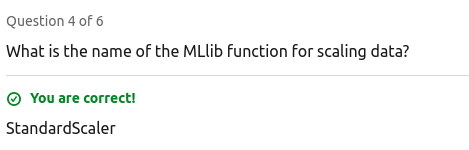
**Create Dataframe:**

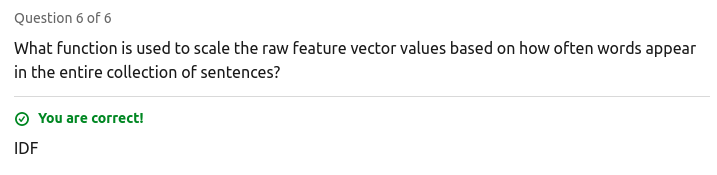


MLLib tokenizer

Used to generate tokens (ex: divide words from a given text files to a list)

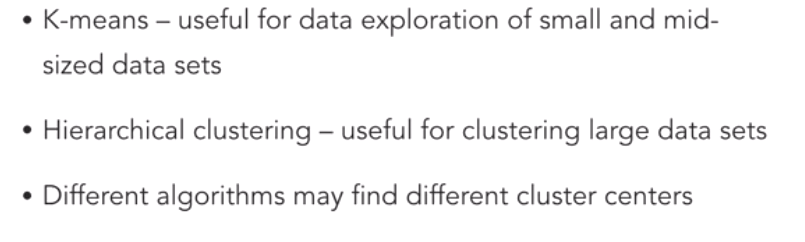






K-Means Clustering (vcluster/ hcluster)

From pyspark.ml.linalg import Vectors ← Vectors package



Classification Algorithms :

1) Naive Byes

**Building Recommendation system with pySpark, Spark MLLib**

* numBlocks: Preset to -1 in an auto-configuration setting. This parameter is meant to parallelize computation.
* custRank: The number of features, otherwise known as latent factors.
* iterations: This parameter represents the number of iterations for ALS to execute. For a reasonable solution to converge on, this algorithm needs roughly 20 iterations or less.
* regParam: The regularization parameter.
* implicitPrefs: This hyperparameter is a specifier. It lets us use either of the following:
  + Explicit feedback
  + Implicit feedback
* alpha: This is a hyperparameter connected to an implicit feedback variant of the ALS algorithm. Its role is to govern the baseline confidence in preference observations.

**Collaborative Filtering: For recommendation systems**

