## Clustering con pyspark.Algoritmo Kmeans

**El clustering es un problema de aprendizaje no supervisado que permite agrupar** una serie de subconjuntos de entidades relacionadas con alguna base de conocimiento.

Dentro de pyspark encontramos el algoritmo Kmeans dentro del paquete **pyspark.mllib.clustering,** que permite agrupar los datos en categorías o grupos en función de un parámetro k.

```
from pyspark.mllib.clustering import KMeans, KMeansModel
from pyspark.mllib.linalg import DenseVector
from pyspark.mllib.linalg import SparseVector
from numpy import array
```

**Entrenamiento**

```
clusters = KMeans.train(parsedData, k, maxIterations=10, runs=10, initializationMode="random")
```

**parsedData= Los puntos de entrenamiento almacenados como RDD**
**k=número de grupos o de clusters**
**runs= Número de ejecuciones en paralelo, por defecto a 1. Se devuelve el mejor modelo.**
**maxIterations = Número máximo de iteraciones**
**initializationMode = Modelo de inicialización(random o k-means)**

El código fuente de la clase Kmeans para ver lo que hace por debajo se puede ver en el respositorio de código oficial de spark

https://apache.googlesource.com/spark/+/master/python/pyspark/mllib/clustering.py

```
    :param initialModel:
      Initial cluster centers can be provided as a KMeansModel object
      rather than using the random or k-means|| initializationModel.
      (default: None)
    """
    if runs != 1:
        warnings.warn("The param `runs` has no effect since Spark 2.0.0.")
    clusterInitialModel = []
    if initialModel is not None:
        if not isinstance(initialModel, KMeansModel):
            raise Exception("initialModel is of "+str(type(initialModel))+". It needs "
                            "to be of <type 'KMeansModel'>")
        clusterInitialModel = [_convert_to_vector(c) for c in initialModel.clusterCenters]
    model = callMLlibFunc("trainKMeansModel", rdd.map(_convert_to_vector), k, maxIterations,
                          runs, initializationMode, seed, initializationSteps, epsilon,
                          clusterInitialModel)
    centers = callJavaFunc(rdd.context, model.clusterCenters)
    return KMeansModel([c.toArray() for c in centers])
```