



| | | |
|------------------------------|---------------------------------|---------------|
| Name: Subrato Tapaswi | Class/Roll No.: D16AD/60 | Grade: |
|------------------------------|---------------------------------|---------------|

Title of Experiment: Implement Anyone Clustering Algorithm in Hadoop.

Objective of Experiment: The objective of this practical exercise is to implement a clustering algorithm using Hadoop, a distributed computing framework, to efficiently analyze datasets.

Outcome of Experiment: Thus, we implemented the K-means clustering Algorithm Using PySpark In Hadoop

Problem Statement: Implement K-Means Clustering Algorithm using PySpark.

Description / Theory:

Clustering: Clustering is an unsupervised machine learning technique that groups similar data points in unlabeled datasets, assigning each group a cluster ID based on shared characteristics.

Types Of Clustering Methods:

- Partitioning Clustering
- Density-Based Clustering
- Distribution Model-Based Clustering
- Hierarchical Clustering
- Fuzzy Clustering



K-Means Clustering:

The k-means algorithm is one of the most popular clustering algorithms. It classifies the dataset by dividing the samples into different clusters of equal variances. The number of clusters must be specified in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Algorithm/Pseudocode:

1. Choose the number of clusters(K) and obtain the data points
2. Place the centroids c_1, c_2, \dots, c_k randomly
3. Repeat steps 4 and 5 until convergence or until the end of a fixed number of iterations
4. for each data point x_i :
 - find the nearest centroid($c_1, c_2 \dots c_k$)
 - assign the point to that cluster
5. for each cluster $j = 1..k$
 - new centroid = mean of all points assigned to that cluster
6. End



Program & Output:

```
[cloudera@quickstart ~]$ pyspark
```

```
>>> df = sqlContext.createDataFrame([[0,33.3,-17.5],[1,40.4,-20.5],[2,28.6,-23.9],[3,29.5,-19.0],[4,32.8,-18.84]],["other","lat","long"])
23/10/04 07:12:13 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
```

```
>>> df.show()
```

| other | lat | long |
|-------|------|--------|
| 0 | 33.3 | -17.5 |
| 1 | 40.4 | -20.5 |
| 2 | 28.6 | -23.9 |
| 3 | 29.5 | -19.0 |
| 4 | 32.8 | -18.84 |

```
>>> from pyspark.ml.feature import VectorAssembler
```

```
>>> vecAssembler = VectorAssembler(inputCols = ["lat", "long"], outputCol = "features")
```

```
>>> new_df = vecAssembler.transform(df)
```

```
>>> new_df.show()
```

| other | lat | long | features |
|-------|------|--------|---------------|
| 0 | 33.3 | -17.5 | [33.3,-17.5] |
| 1 | 40.4 | -20.5 | [40.4,-20.5] |
| 2 | 28.6 | -23.9 | [28.6,-23.9] |
| 3 | 29.5 | -19.0 | [29.5,-19.0] |
| 4 | 32.8 | -18.84 | [32.8,-18.84] |

```
>>> from pyspark.ml.clustering import KMeans
```

```
>>> kmeans = KMeans(k=2, seed=1)
```

```
>>> model = kmeans.fit(new_df.select('features'))
```

```
>>> transformed = model.transform(new_df)
```

```
>>> transformed.show()
```

| other | lat | long | features | prediction |
|-------|------|--------|---------------|------------|
| 0 | 33.3 | -17.5 | [33.3,-17.5] | 0 |
| 1 | 40.4 | -20.5 | [40.4,-20.5] | 1 |
| 2 | 28.6 | -23.9 | [28.6,-23.9] | 0 |
| 3 | 29.5 | -19.0 | [29.5,-19.0] | 0 |
| 4 | 32.8 | -18.84 | [32.8,-18.84] | 0 |

Results and Discussions: K-means clustering, along with the use of VectorAssembler, was applied to a small dataset of geographical coordinates.