Name:Shreya Singh	Class/RollNo.:D16AD/55	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 or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group." It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.It is an unsupervised learning method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.After applying this clustering technique,
each cluster or group is provided with a cluster-ID
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. It is fast with fewer computations required, with the linear complexity of O(n). It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters. The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. 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/ Pseudo Code / Flowchart (whichever is applicable)

1. Choose the number of clusters(K) and obtain the data points
2. Place the centroids c_1, c_2, ... 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 .. 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 :

```
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]],["ot
her","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'))
```


```
>>> 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:

BDA/Odd Sem 2023-23/Experiment 6

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

Clustering Results:

K=2 was determined to be the optimal number of clusters.
Cluster 1 represents a unique location.
Cluster 2 and Cluster 3 contain coordinates that are geographically closer to each other.

Application of VectorAssembler:

VectorAssembler was employed to assemble the latitude and longitude into a feature vector, streamlining the input for the K-means algorithm.

Insights:

The integration of VectorAssembler simplified the input structure, enhancing the efficiency of the K-means algorithm.
The clustering successfully grouped similar geographic points based on proximity, illustrating the practical use of feature engineering.