19/4/23

EXPERIMENT - 7

AIM

To implement the K-Means clustering algorithm in Python

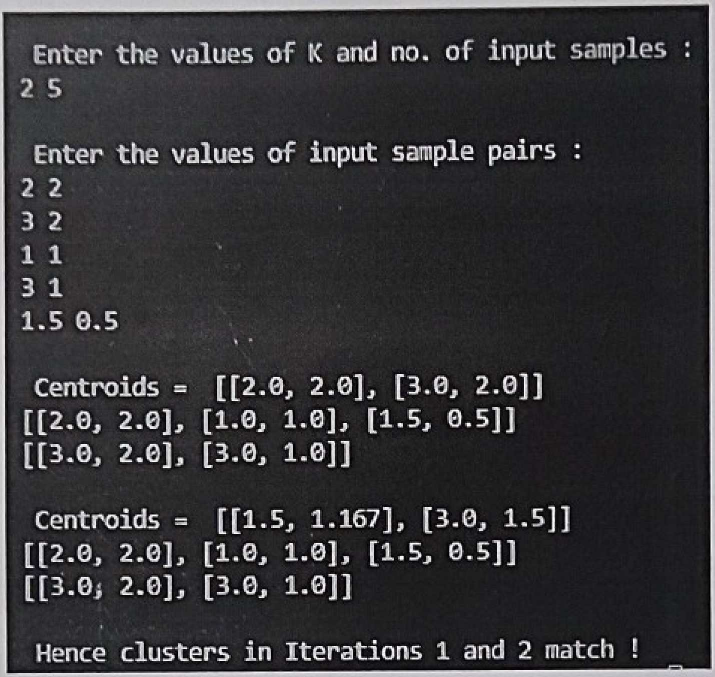
ALGORITHM

* Select K (number of clusters) and K random points or centroids from the input dataset
* For each data point, calculate Euclidean distance to all centroids and assign it to the cluster of the closest centroid
* Calculate average of all data points in each cluster to get new centroids
* Repeat steps 2-3 until the clusters in any 2 consecutive iterations match.
* Model is ready with new centroids & K clusters

THEORY

K-Means Clustering is an unsupervised learning algorithm which groups the unlabeled dataset into different clusters. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters

OUTPUT



```
 Enter the values of K and no. of input samples :
2 5

 Enter the values of input sample pairs :
2 2
3 2
1 1
3 1
1.5 0.5

 Centroids =  [[2.0, 2.0], [3.0, 2.0]]
[[2.0, 2.0], [1.0, 1.0], [1.5, 0.5]]
[[3.0, 2.0], [3.0, 1.0]]

 Centroids =  [[1.5, 1.167], [3.0, 1.5]]
[[2.0, 2.0], [1.0, 1.0], [1.5, 0.5]]
[[3.0, 2.0], [3.0, 1.0]]

 Hence clusters in Iterations 1 and 2 match !
```

CODE

```
import math

def calculateEuclideanDistance ( point , center ) :
   return round ( math.sqrt ( ( point [ 0 ] - center [ 0 ] ) ** 2 + ( point [ 1 ] - center [ 1 ] ) ** 2 ) , 3 )

def executeIteration ( call , prev_clusters ) :
   centroids , distances , clusters , start = [] , [] , [ [] for i in range ( K ) ] , K if call == 1 else 0
   if call == 1 :
      for cind in range ( K ) :
         centroids.append ( inputs [ cind ] )
         clusters [ cind ].append ( inputs [ cind ] )
   else :
      for lst in prev_clusters :
         x , y = 0 , 0
         lng = len ( lst )
         for point in lst :
            x , y = x + point [ 0 ] , y + point [ 1 ]
         centroids.append ( [ round ( x / lng , 3 ) , round ( y / lng , 3 ) ] )
   print ( "\n Centroids = " , centroids )
   for ind in range ( start , len ( inputs ) ) :
      for center in range ( K ) :
         distances.append ( calculateEuclideanDistance ( inputs [ ind ] , centroids [ center ] ) )
      id = distances.index ( min ( distances ) )
      clusters [ id ].append ( inputs [ ind ] )
      distances.clear ()
   for cl in clusters :
      print ( cl )
   if call != 1 and prev_clusters == clusters :
      return False , prev_clusters
   prev_clusters = clusters
   return True , prev_clusters

def KmeansCluster () :
   itr , cont , prev_clusters = 1 , True , []
   while cont == True :
      cont , prev_clusters = executeIteration ( itr , prev_clusters )
      itr += 1
   print ( "Hence clusters in Iterations" , itr - 2 , "and" , itr - 1 , "match !")

print ( "\n Enter the values of K and no. of input samples : " )
K , n_input = map ( int , input ().split () )
inputs = []
print ( "\n Enter the values of input sample pairs : " )
for el in range ( n_input ) :
   inputs.append ( list ( map ( float , input ().split () ) ) )
KmeansCluster ()
```

RESULT

Hence the K-means clustering algorithm has been implemented successfully