EXPERIMENT NO 5

NAME: Vishal Shashikant Salvi **UID: 2019230069**

Class: TE Comps **BATCH:C**

Aim: To implement K means clustering Algorithm.

Theory:

Clustering:

Clustering is one of the most common exploratory data analysis technique used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different. In other words, we try to find homogeneous subgroups within the data such that data points in each cluster are as similar as possible according to a similarity measure such as Euclidean-based distance or correlation-based distance. The decision of which similarity measure to use is application-specific.

Types of clustering:

Broadly speaking, clustering can be divided into two subgroups-

- **Hard Clustering:** In hard clustering, each data point either belongs to a cluster completely or not.
- **Soft Clustering:** In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned.

Kmeans Algorithm

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into *K* pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The way kmeans algorithm works is as follows:

1. Specify number of clusters *K*.
2. Initialize centroids by first shuffling the dataset and then randomly selecting *K* data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

- Compute the sum of the squared distance between data points and all centroids.
- Assign each data point to the closest cluster (centroid).
- Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

Applications

kmeans algorithm is very popular and used in a variety of applications such as market segmentation, document clustering, image segmentation and image compression, etc. The goal usually when we undergo a cluster analysis is either:

1. Get a meaningful intuition of the structure of the data we're dealing with.
2. Cluster-then-predict where different models will be built for different subgroups if we believe there is a wide variation in the behaviours of different subgroups. An

example of that is clustering patients into different subgroups and build a model for each subgroup to predict the probability of the risk of having heart attack.

Advantages

1) Fast, robust and easier to understand.

2) Relatively efficient: O(tknd), where n is # objects, k is # clusters, d is # dimension of each object, and t is # iterations. Normally, k, t, d << n.

3) Gives best result when data set are distinct or well separated from each other.

Disadvantages

1) The learning algorithm requires apriori specification of the number of cluster centres.

2) The use of Exclusive Assignment - If there are two highly overlapping data then k-means will not be able to resolve that there are two clusters.

3) The learning algorithm is not invariant to non-linear transformations i.e. with different representation of data we get different results (data represented in form of cartesian co-ordinates and polar co-ordinates will give different results).

4) Euclidean distance measures can unequally weight underlying factors.

5) The learning algorithm provides the local optima of the squared error function.

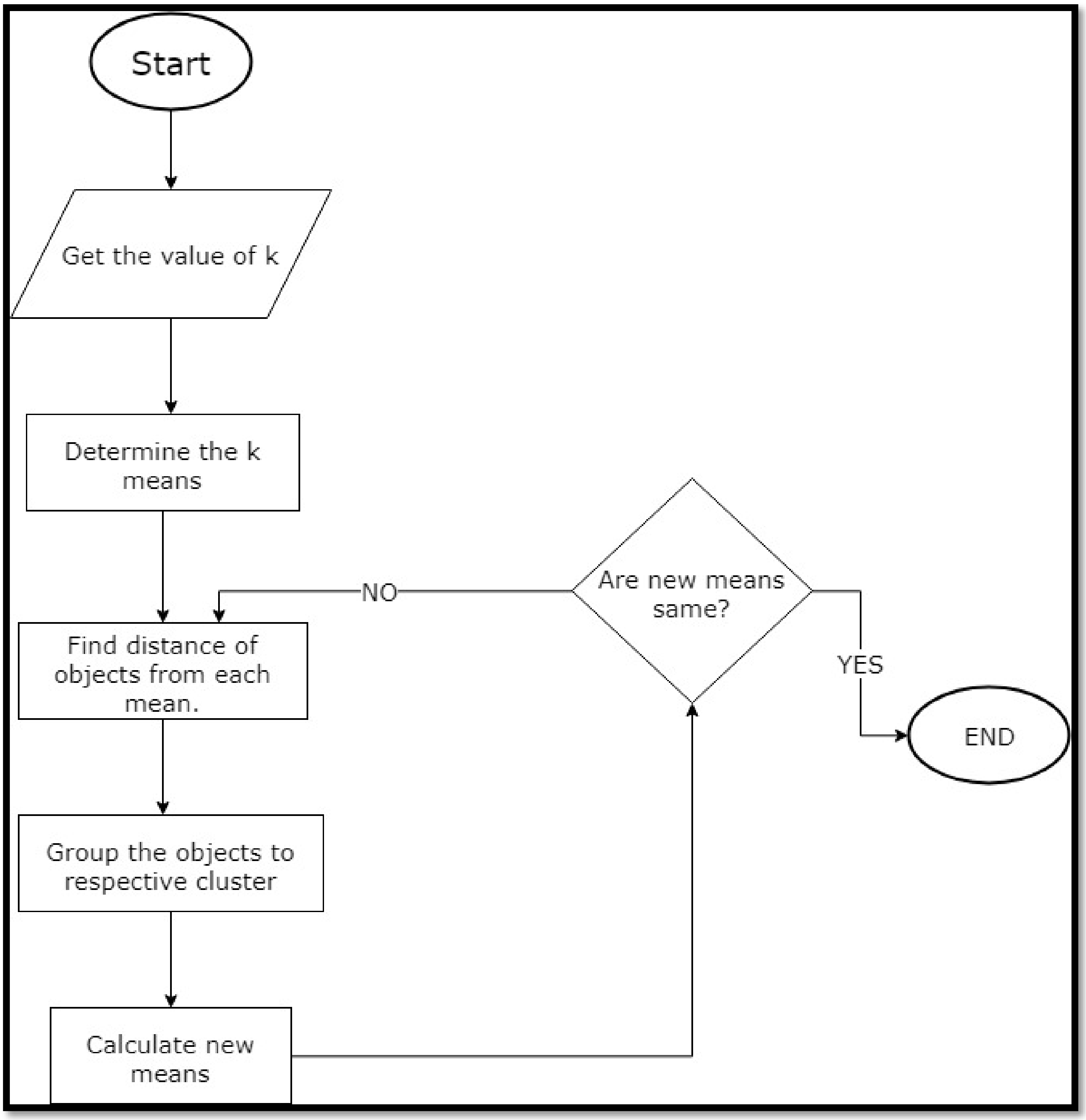
6) Randomly choosing of the cluster center cannot lead us to the fruitful result

7) Applicable only when mean is defined i.e. fails for categorical data.

8) Unable to handle noisy data and outliers.

9) Algorithm fails for non-linear data set.

K-means flowchart:



K-means algorithm:

Initialize k means with random values from the dataset.

While the means of two consecutive iterations are not the same, for each data in the dataset repeat:

Find the mean closest to the data.

1.1. Assign that data to the mean.

1.2. Update the means.

K-means example:

We will apply k-means on the following 1-dimensional data set for K=2.

Data set {2, 4, 10, 12, 3, 20, 30, 11, 25}

Iteration 1

M1, M2 are the two randomly selected centroids/means where

M1= 4, M2=11

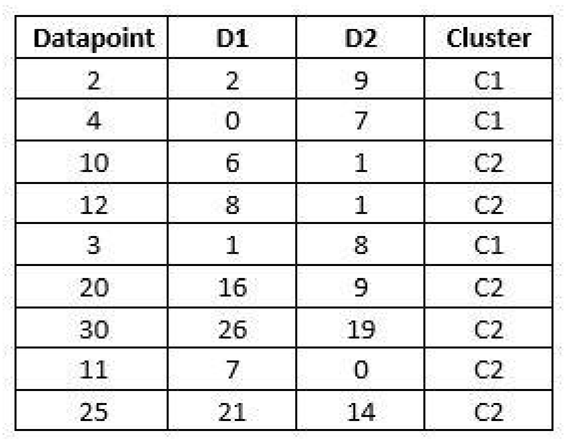
and the initial clusters are

C1= {4}, C2= {11}

Calculating the Euclidean distance for all the points

D=[x, a]=$\sqrt{(x-a)^2}$

D1 is the distance from M1 and D2 is the distance from M2.



Datapoint	D1	D2	Cluster
2	2	9	C1
4	0	7	C1
10	6	1	C2
12	8	1	C2
3	1	8	C1
20	16	9	C2
30	26	19	C2
11	7	0	C2
25	21	14	C2

<u>Iteration 1</u>

As we can see in the above table, 2 data points are added to cluster C1 and other data points added to cluster C2.

Therefore,

C1= {2, 4, 3} and C2= {10, 12, 20, 30, 11, 25}

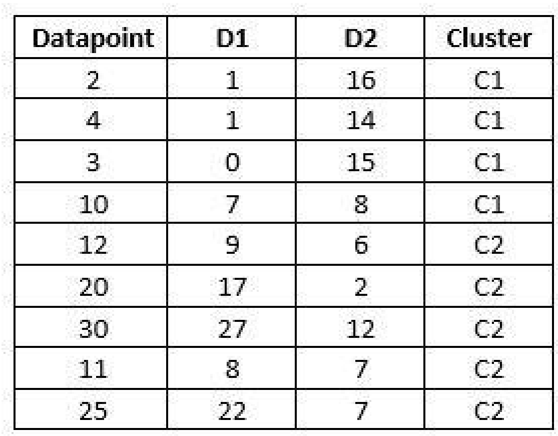
<u>Iteration 2</u>

Calculate new mean of data points in C1 and C2.

M1= (2+3+4)/3= 3

M2= (10+12+20+30+11+25)/6= 18

Calculating distance and updating clusters based on table below



Datapoint	D1	D2	Cluster
2	1	16	C1
4	1	14	C1
3	0	15	C1
10	7	8	C1
12	9	6	C2
20	17	2	C2
30	27	12	C2
11	8	7	C2
25	22	7	C2

New Clusters

C1= {2, 3, 4, 10} and C2= {12, 20, 30, 11, 25}

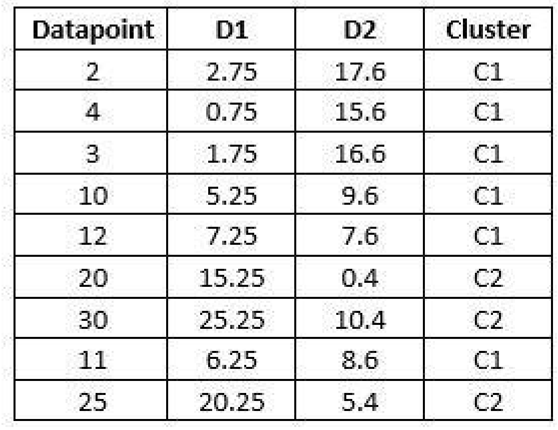
<u>Iteration 3</u>

Calculate new mean of data points in C1 and C2.

M1= (2+3+4+10)/4= 4.75

M2= (12+20+30+11+25)/5= 19.6

Calculating distance and updating clusters based on table below



Datapoint	D1	D2	Cluster
2	2.75	17.6	C1
4	0.75	15.6	C1
3	1.75	16.6	C1
10	5.25	9.6	C1
12	7.25	7.6	C1
20	15.25	0.4	C2
30	25.25	10.4	C2
11	6.25	8.6	C1
25	20.25	5.4	C2

New Clusters

C1= {2, 3, 4, 10, 12, 11} and C2= {20, 30, 25}

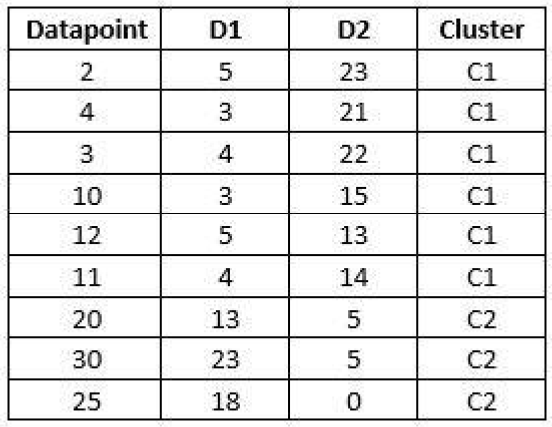
<u>Iteration 4</u>

Calculate new mean of data points in C1 and C2.

M1= (2+3+4+10+12+11)/6=7

M2= (20+30+25)/3= 25

Calculating distance and updating clusters based on table below



Datapoint	D1	D2	Cluster
2	5	23	C1
4	3	21	C1
3	4	22	C1
10	3	15	C1
12	5	13	C1
11	4	14	C1
20	13	5	C2
30	23	5	C2
25	18	0	C2

New Clusters

C1= {2, 3, 4, 10, 12, 11} and C2= {20, 30, 25}

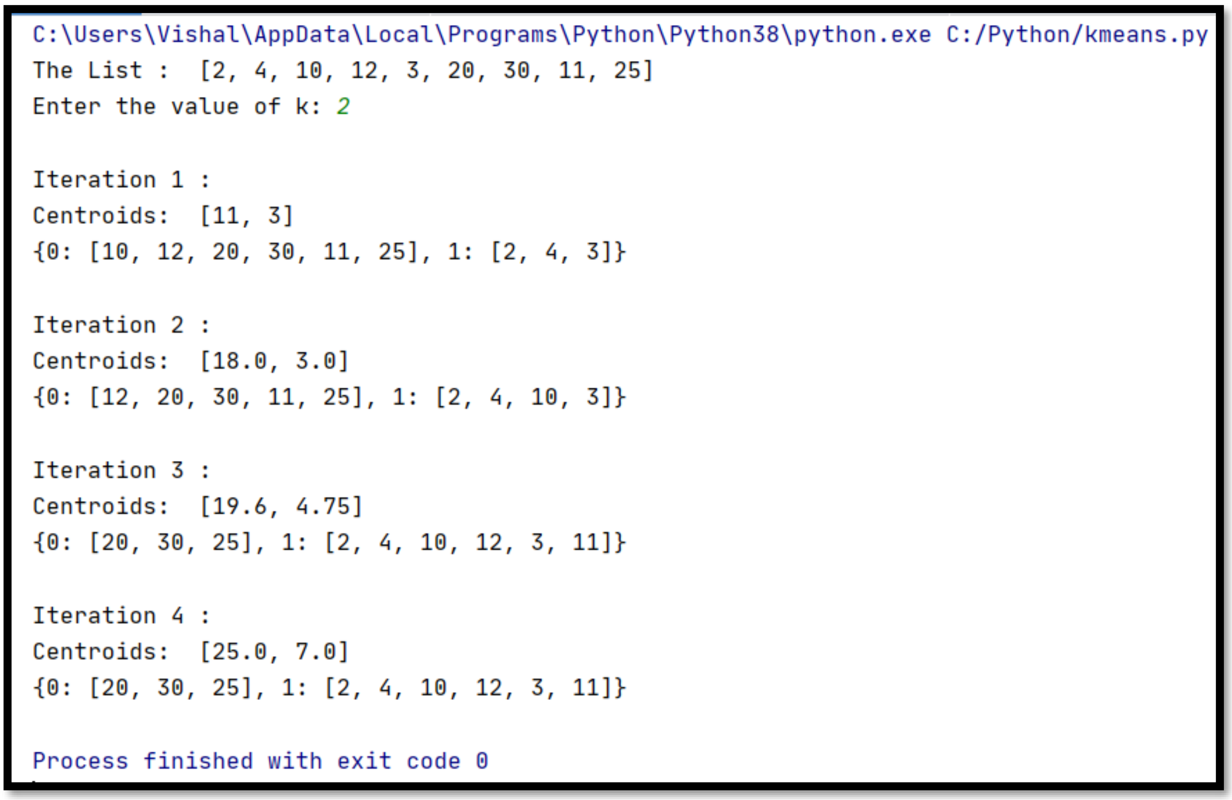
As we can see that the data points in the cluster C1 and C2 in iteration 3 are same as the data points of the cluster C1 and C2 of iteration 2.

It means that none of the data points has moved to another cluster. Also, the means/centroid of these clusters is constant. So, this becomes the stopping condition for our algorithm.

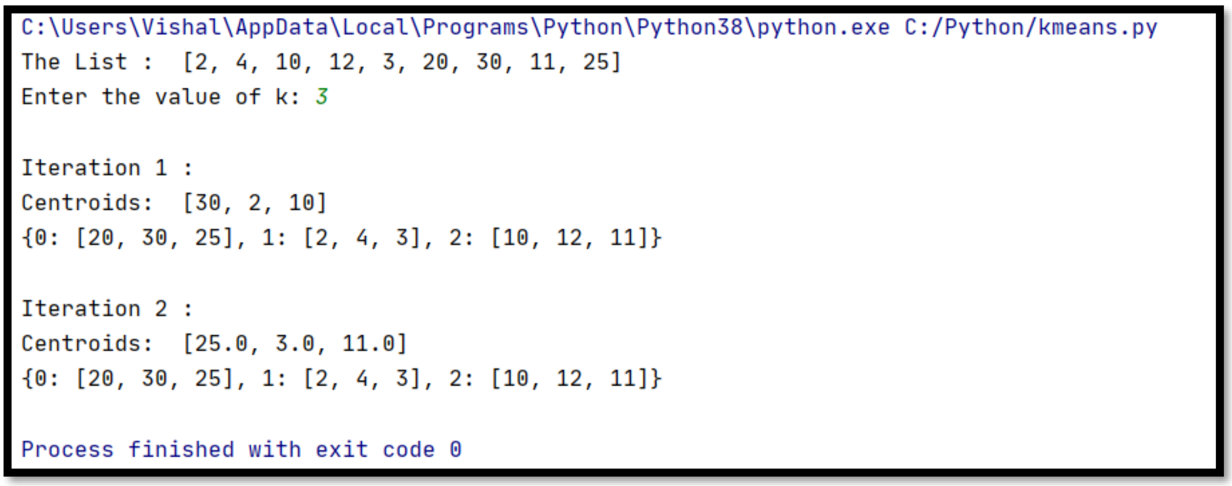
Code:

```
import random
mylist = [2,4,10,12,3,20,30,11,25]
print("The List : ",mylist)
k = int(input("Enter the value of k: "))
centroids = random.sample(mylist,k)
dit = {}
iterationNo = 0
while True:
 iterationNo += 1
 print("\nIteration",iterationNo,":")
 print("Centroids: ", centroids)
 for x in range(len(centroids)):
     dit[x] = []
 for i in mylist:
    minDiff = 999
    for j in range(len(centroids)):
        if minDiff > abs(i - centroids[j]):
            minDiff = abs(i - centroids[j])
            clusterNo = j
    dit[clusterNo].append(i)
 print(dit)
 newCentroids = centroids[:]
 for i in range(len(centroids)):
    centroids[i] = 0
    for j in range(len(dit[i])):
        centroids[i] += dit[i][j]
    centroids[i] = centroids[i] / len(dit[i])
 if centroids == newCentroids:
     break
```

Output:



```
C:\Users\Vishal\AppData\Local\Programs\Python\Python38\python.exe C:/Python/kmeans.py
The List :  [2, 4, 10, 12, 3, 20, 30, 11, 25]
Enter the value of k: 2

Iteration 1 :
Centroids:  [11, 3]
{0: [10, 12, 20, 30, 11, 25], 1: [2, 4, 3]}

Iteration 2 :
Centroids:  [18.0, 3.0]
{0: [12, 20, 30, 11, 25], 1: [2, 4, 10, 3]}

Iteration 3 :
Centroids:  [19.6, 4.75]
{0: [20, 30, 25], 1: [2, 4, 10, 12, 3, 11]}

Iteration 4 :
Centroids:  [25.0, 7.0]
{0: [20, 30, 25], 1: [2, 4, 10, 12, 3, 11]}

Process finished with exit code 0
```

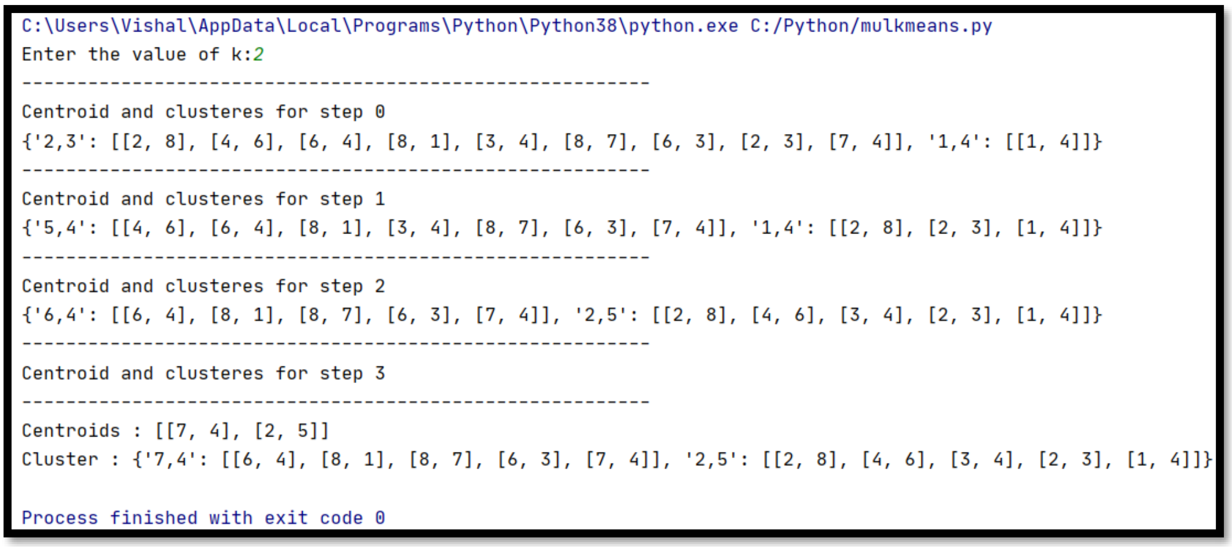


```
C:\Users\Vishal\AppData\Local\Programs\Python\Python38\python.exe C:/Python/kmeans.py
The List :  [2, 4, 10, 12, 3, 20, 30, 11, 25]
Enter the value of k: 3

Iteration 1 :
Centroids:  [30, 2, 10]
{0: [20, 30, 25], 1: [2, 4, 3], 2: [10, 12, 11]}

Iteration 2 :
Centroids:  [25.0, 3.0, 11.0]
{0: [20, 30, 25], 1: [2, 4, 3], 2: [10, 12, 11]}

Process finished with exit code 0
```

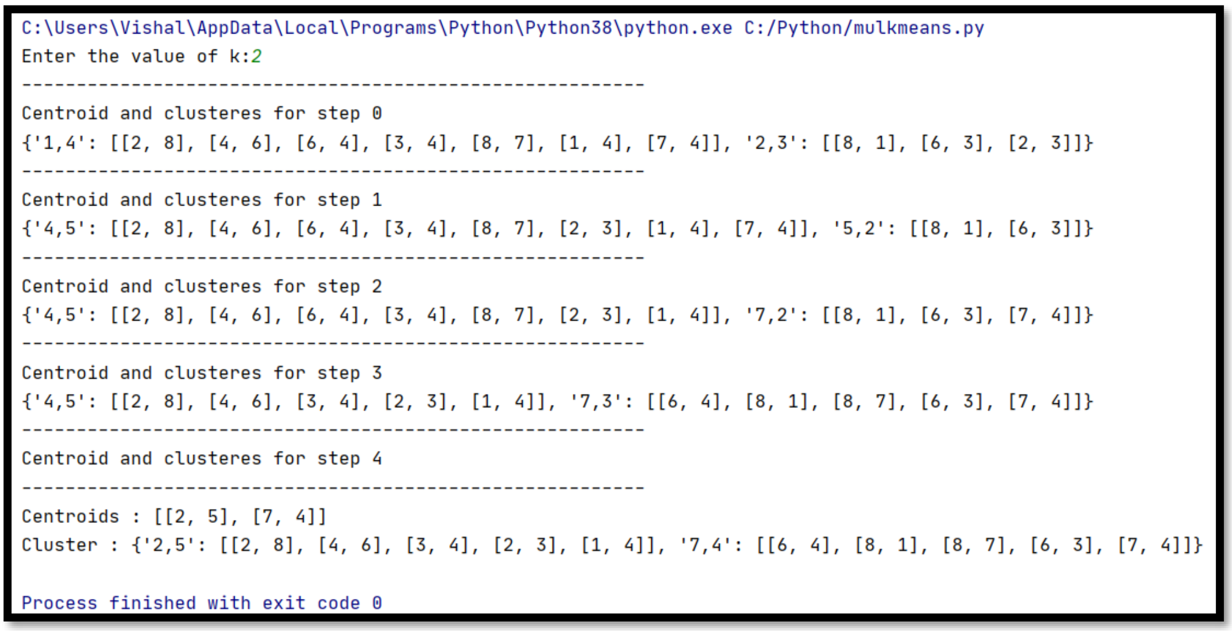
Code:

```
import random
dataset = [[2,8],[4,6],[6,4],[8,1],[3,4],[8,7],[6,3],[2,3],[1,4],[7,4]]
k= int(input("Enter the value of k:"))
centroids=[]
i=0
while i<k:
    centroid = random.choice(dataset)
    if centroid not in centroids:
        centroids.append(centroid)
        i+=1
j=0;
while True:
    cluster = dict()
    print("----------------------------------------------------------")
    print("Centroid and clusteres for step", j)
    j = j+1;
    for centroid in centroids:
        cluster [str(centroid[0])+","+str(centroid[1])] = []
    for d in dataset:
        mi = [abs(d[0] - centroid[0])+abs(d[1] - centroid[1])for centroid
in centroids]
        m_index = mi.index(min(mi))

cluster[str(centroids[m_index][0])+","+str(centroids[m_index][1])].append(d
)
    new_centroid = []
    for c_list in cluster.values():
        a = b = 0
        for i_list in c_list:
            a += i_list[0]
            b += i_list[1]
        new_centroid.append([round(a/len(c_list)),round(b/len(c_list))])
    if new_centroid == centroids:
        print("----------------------------------------------------------")
        print("Centroids :",new_centroid)
        print("Cluster :",cluster)
        break
    else:
        centroids = new_centroid
        print(cluster)
```

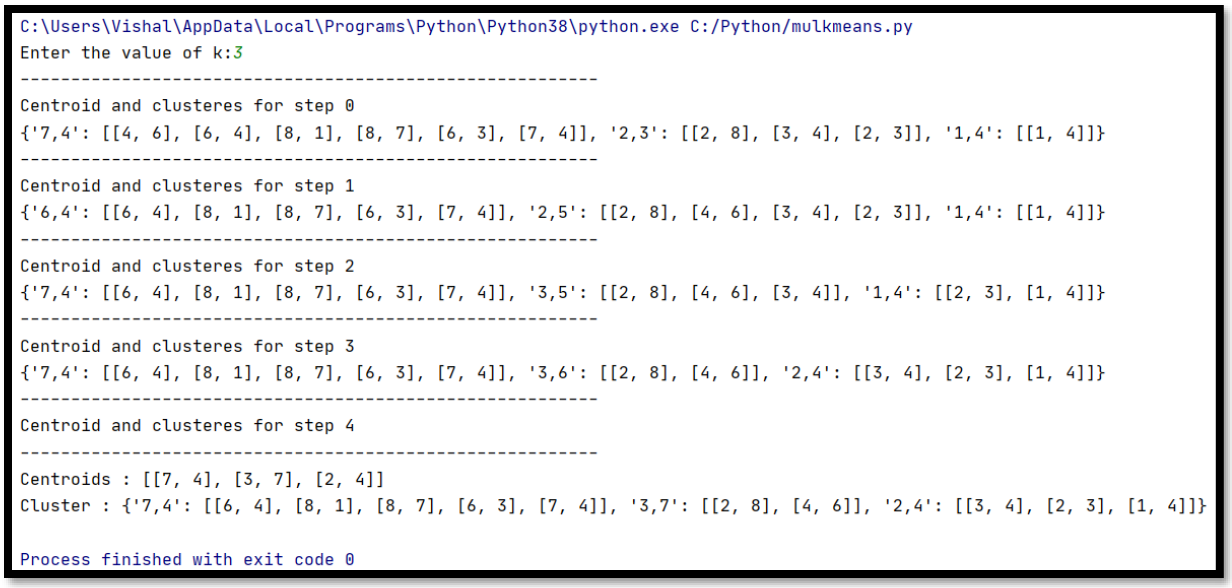
Output:



```
C:\Users\Vishal\AppData\Local\Programs\Python\Python38\python.exe C:/Python/mulkmeans.py
Enter the value of k:2
---------------------------------------------------------
Centroid and clusteres for step 0
{'2,3': [[2, 8], [4, 6], [6, 4], [8, 1], [3, 4], [8, 7], [6, 3], [2, 3], [7, 4]], '1,4': [[1, 4]]}
---------------------------------------------------------
Centroid and clusteres for step 1
{'5,4': [[4, 6], [6, 4], [8, 1], [3, 4], [8, 7], [6, 3], [7, 4]], '1,4': [[2, 8], [2, 3], [1, 4]]}
---------------------------------------------------------
Centroid and clusteres for step 2
{'6,4': [[6, 4], [8, 1], [8, 7], [6, 3], [7, 4]], '2,5': [[2, 8], [4, 6], [3, 4], [2, 3], [1, 4]]}
---------------------------------------------------------
Centroid and clusteres for step 3
---------------------------------------------------------
Centroids : [[7, 4], [2, 5]]
Cluster : {'7,4': [[6, 4], [8, 1], [8, 7], [6, 3], [7, 4]], '2,5': [[2, 8], [4, 6], [3, 4], [2, 3], [1, 4]]}

Process finished with exit code 0
```



```
C:\Users\Vishal\AppData\Local\Programs\Python\Python38\python.exe C:/Python/mulkmeans.py
Enter the value of k:2
---------------------------------------------------------
Centroid and clusteres for step 0
{'1,4': [[2, 8], [4, 6], [6, 4], [3, 4], [8, 7], [1, 4], [7, 4]], '2,3': [[8, 1], [6, 3], [2, 3]]}
---------------------------------------------------------
Centroid and clusteres for step 1
{'4,5': [[2, 8], [4, 6], [6, 4], [3, 4], [8, 7], [2, 3], [1, 4], [7, 4]], '5,2': [[8, 1], [6, 3]]}
---------------------------------------------------------
Centroid and clusteres for step 2
{'4,5': [[2, 8], [4, 6], [6, 4], [3, 4], [8, 7], [2, 3], [1, 4]], '7,2': [[8, 1], [6, 3], [7, 4]]}
---------------------------------------------------------
Centroid and clusteres for step 3
{'4,5': [[2, 8], [4, 6], [3, 4], [2, 3], [1, 4]], '7,3': [[6, 4], [8, 1], [8, 7], [6, 3], [7, 4]]}
---------------------------------------------------------
Centroid and clusteres for step 4
---------------------------------------------------------
Centroids : [[2, 5], [7, 4]]
Cluster : {'2,5': [[2, 8], [4, 6], [3, 4], [2, 3], [1, 4]], '7,4': [[6, 4], [8, 1], [8, 7], [6, 3], [7, 4]]}

Process finished with exit code 0
```



```
C:\Users\Vishal\AppData\Local\Programs\Python\Python38\python.exe C:/Python/mulkmeans.py
Enter the value of k:3
---------------------------------------------------------
Centroid and clusteres for step 0
{'7,4': [[4, 6], [6, 4], [8, 1], [8, 7], [6, 3], [7, 4]], '2,3': [[2, 8], [3, 4], [2, 3]], '1,4': [[1, 4]]}
---------------------------------------------------------
Centroid and clusteres for step 1
{'6,4': [[6, 4], [8, 1], [8, 7], [6, 3], [7, 4]], '2,5': [[2, 8], [4, 6], [3, 4], [2, 3]], '1,4': [[1, 4]]}
---------------------------------------------------------
Centroid and clusteres for step 2
{'7,4': [[6, 4], [8, 1], [8, 7], [6, 3], [7, 4]], '3,5': [[2, 8], [4, 6], [3, 4]], '1,4': [[2, 3], [1, 4]]}
---------------------------------------------------------
Centroid and clusteres for step 3
{'7,4': [[6, 4], [8, 1], [8, 7], [6, 3], [7, 4]], '3,6': [[2, 8], [4, 6]], '2,4': [[3, 4], [2, 3], [1, 4]]}
---------------------------------------------------------
Centroid and clusteres for step 4
---------------------------------------------------------
Centroids : [[7, 4], [3, 7], [2, 4]]
Cluster : {'7,4': [[6, 4], [8, 1], [8, 7], [6, 3], [7, 4]], '3,7': [[2, 8], [4, 6]], '2,4': [[3, 4], [2, 3], [1, 4]]}

Process finished with exit code 0
```

Conclusion:

Thus, I studied and understood the concept of clustering and its various types. I also studied and implemented K-means clustering algorithm, which is a type of unsupervised learning. The major advantage of k-means is that it is easy to implement. The major difficulty is to predict the number of clusters (value of k).