Name-Rebecca Dias Class- TE-CMPNA Roll No-19   PID-182027

Experiment No. 05

Aim: To implement clustering algorithm K means graph.

Theory:

1. What is clustering in data mining?

Clustering is the method of converting a group of abstract objects into classes of similar objects. Clustering is a method of partitioning a set of data or objects into a set of significant subclasses called clusters. It helps users to understand the structure or natural grouping in a data set and used either as a stand-alone instrument to get a better insight into data distribution or as a pre-processing step for other algorithms.

Difference between classification and clustering-

1. Classification is used for supervised learning whereas clustering is used for unsupervised learning.
2. The process of classifying the input instances based on their corresponding class labels are known as classification whereas grouping the instances based on their similarity without the help of class labels is known as clustering.
3. As Classification have labels so there is need of training and testing dataset for verifying the model created but there is no need for training and testing dataset in clustering.
4. Classification is more complex as compared to clustering as there are many levels in classification phase whereas only grouping is done in clustering.
5. Classification examples are Logistic regression, Naive Bayes classifier, Support vector machines etc. Whereas clustering examples are k-means clustering algorithm, Fuzzy c-means clustering algorithm, Gaussian (EM) clustering algorithm etc.

2. Explain K-means Algorithm (pseudo code) and solve a clustering problem.

- **Step1:** Form *k* centroids, randomly
- **Step2:** Calculate distance between centroids and each object
  - Use Euclidean's law to determine min distance:

    $$d(A,B) = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$$
- **Step3:** Assign objects based on min distance to *k* clusters
- **Step4:** Calculate centroid of each cluster using

  $$C = \left(\frac{x_1+x_2+\ldots x_n}{n}, \frac{y_1+y_2+\ldots y_n}{n}\right)$$
  - Go to step 2.
  - Repeat until no change in centroids.

Example:

Question:

Suppose that the data mining task is to cluster points (with (x, y) representing location) **[10]** into three clusters, where the points are: A_1 (2, 10), A_2 (2, 5), A_3 (8, 4), B_1 (5, 8), B_2 (7, 5), B_3 (6, 4), C_1 (1, 2), C_2 (4, 9).
The distance function is Euclidean distance. Suppose initially we assign A_1, B_1, and C_1 as the center of each cluster, respectively. Use the *k-means* algorithm to show only (i) The three cluster centers after the first round of execution (ii) The final three clusters.

Answer:

9. $A_1(2,10)$
$A_2(2,5)$
$A_3(8,4)$
$B_1(5,8)$
$B_2(7,5)$
$B_3(6,4)$
$C_1(1,2)$
$C_2(4,9)$

Let G_1, G_2 and G_3 be the three centroids.

$G_1(2,10)$, $G_2(5,8)$, $G_3(1,2)$

Initially

$d(G_1, A_1) = \sqrt{(2-2)^2+(10-10)^2} = 0$
$d(G_1, A_2) = \sqrt{(2-2)^2+(5-10)^2} = 5$
$d(G_1, A_3) = \sqrt{(8-2)^2+(4-10)^2} = 8.485$
$d(G_1, B_1) = \sqrt{(5-2)^2+(8-10)^2} = 3.605$
$d(G_1, B_2) = \sqrt{(7-2)^2+(5-10)^2} = 7.071$
$d(G_1, B_3) = \sqrt{(6-2)^2+(4-10)^2} = 7.211$
$d(G_1, C_1) = \sqrt{(1-2)^2+(2-10)^2} = 8.062$
$d(G_1, C_2) = \sqrt{(4-2)^2+(9-10)^2} = 2.236$
$d(G_2, A_1) = \sqrt{(2-5)^2+(10-8)^2} = 3.605$
$d(G_2, A_2) = \sqrt{(2-5)^2+(\cancel{10}5-8)^2} = 4.243$
$d(G_2, A_3) = \sqrt{(8-5)^2+(4-8)^2} = 5$
$d(G_2, B_1) = \sqrt{(5-5)^2+(8-8)^2} = 0$
$d(G_2, B_2) = \sqrt{(7-5)^2+(5-8)^2} = 3.605$
$d(G_2, B_3) = \sqrt{(6-5)^2+(4-8)^2} = 4.123$
$d(G_2, C_1) = \sqrt{(1-5)^2+(2-8)^2} = 7.211$
$d(G_2, C_2) = \sqrt{(4-5)^2+(9-8)^2} = 1.414$
$d(G_3, A_1) = \sqrt{(2-1)^2+(10-2)^2} = 8.062$
$d(G_3, A_2) = \sqrt{(2-1)^2+(5-2)^2} = 3.162$

$d(G_3, A_3) = \sqrt{(8-1)^2 + (4-2)^2} = 7.280$

$d(G_3, B_1) = \sqrt{(5-1)^2 + (8-2)^2} = 7.211$

$d(G_3, B_2) = \sqrt{(7-1)^2 + (5-2)^2} = 6.708$

$d(G_3, B_3) = \sqrt{(6-1)^2 + (4-2)^2} = 5.385$

$d(G_3, C_1) = \sqrt{(1-1)^2 + (2-2)^2} = 0$

$d(G_3, C_2) = \sqrt{(4-1)^2 + (9-2)^2} = 7.616.$

Distance Matrix :-

$$
\begin{array}{c}
\\ G_1 \\ G_2 \\ G_3
\end{array}
\begin{array}{c}
\begin{array}{cccccccc} A_1 & A_2 & A_3 & B_1 & B_2 & B_3 & C_1 & C_2 \end{array} \\
\begin{bmatrix}
0 & 5 & 8.485 & 3.605 & 7.071 & 7.211 & 8.062 & 2.236 \\
3.605 & 4.243 & 5 & 0 & 3.605 & 4.123 & 7.211 & 1.414 \\
8.062 & 3.162 & 7.280 & 7.211 & 6.708 & 5.385 & 0 & 7.616
\end{bmatrix}
\end{array}
$$

Group Matrix.

$$
\begin{array}{c}
\\ G_1 \\ G_2 \\ G_3
\end{array}
\begin{array}{c}
\begin{array}{cccccccc} A_1 & A_2 & A_3 & B_1 & B_2 & B_3 & C_1 & C_2 \end{array} \\
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\end{array}
$$

for G_1, $x = 2$
$y = 10$

For G_2,

$$x = \frac{8+5+7+6+4}{5}, \quad y = \frac{4+8+5+4+9}{5}$$

$x = 6$, $y = 6$.

for G_3,

$$x = \frac{2+1}{2}, \quad y = \frac{5+2}{2}$$

$x = 1.5$, $y = 3.5$

$G_1(2,10)$, $G_2(6,6)$, $G_3(1.5, 3.5)$

Iteration 1:-

	A_1	A_2	A_3	B_1	B_2	B_3	C_1	C_2
G_1	0	5	8.485	3.605	7.071	7.211	8.062	2.236
G_2	5.657	4.123	2.828	2.236	1.414	2	6.403	3.605
G_3	6.519	1.581	6.519	5.7	5.7	4.528	1.581	6.042

	A_1	A_2	A_3	B_1	B_2	B_3	C_1	C_2
G_1	1	0	0	0	0	0	0	1
G_2	0	0	1	1	1	1	0	0
G_3	0	1	0	0	0	0	1	0

$$G_1\left(\frac{2+4}{2}, \frac{10+9}{2}\right) = G_1(3, 9.5)$$

$$G_2\left(\frac{8+5+7+6}{4}, \frac{4+8+5+4}{4}\right) = G_2(6.5, 5.25)$$

$$G_3\left(\frac{2+1}{2}, \frac{5+2}{2}\right) = G_3(1.5, 3.5)$$

Iteration 2:

	A_1	A_2	A_3	B_1	B_2	B_3	C_1	C_2
G_1	1.118	4.609	7.433	2.5	6.021	6.264	7.762	1.118
G_2	6.543	4.507	1.953	3.132	0.559	1.346	6.388	4.507
G_3	6.519	1.587	6.519	5.7	5.7	4.528	1.581	6.042

	A_1	A_2	A_3	B_1	B_2	B_3	C_1	C_2
G_1	1	0	0	1	0	0	0	1
G_2	0	0	1	0	1	1	0	0
G_3	0	1	0	0	0	0	1	0

G1 (3.667, 9) , G2 (7, 4.333) , G3 (1.5, 3.5)

Iteration 3:-

	A_1	A_2	A_3	B_1	B_2	B_3	C_1	C_2
G1	1.944	4.333	6.616	1.666	5.206	5.717	7.491	0.333
G2	7.557	5.049	1.054	4.177	0.667	1.539	6.437	5.548
G3	6.519	1.581	6.519	5.7	5.7	4.527	1.581	6.041

	A_1	A_2	A_3	B_1	B_2	B_3	C_1	C_2
G1	1	0	0	1	0	0	0	1
G2	0	0	1	0	1	1	0	0
G3	0	1	0	0	0	0	1	0.

~~Since~~ the current group matrix is equal to the previous group matrix.

The final grouping is given below.

A_1, B_1, C_2 assigned G1
A_3, B_2, B_3 assigned G2
A_2, C_1 assigned G3.

Implementation:

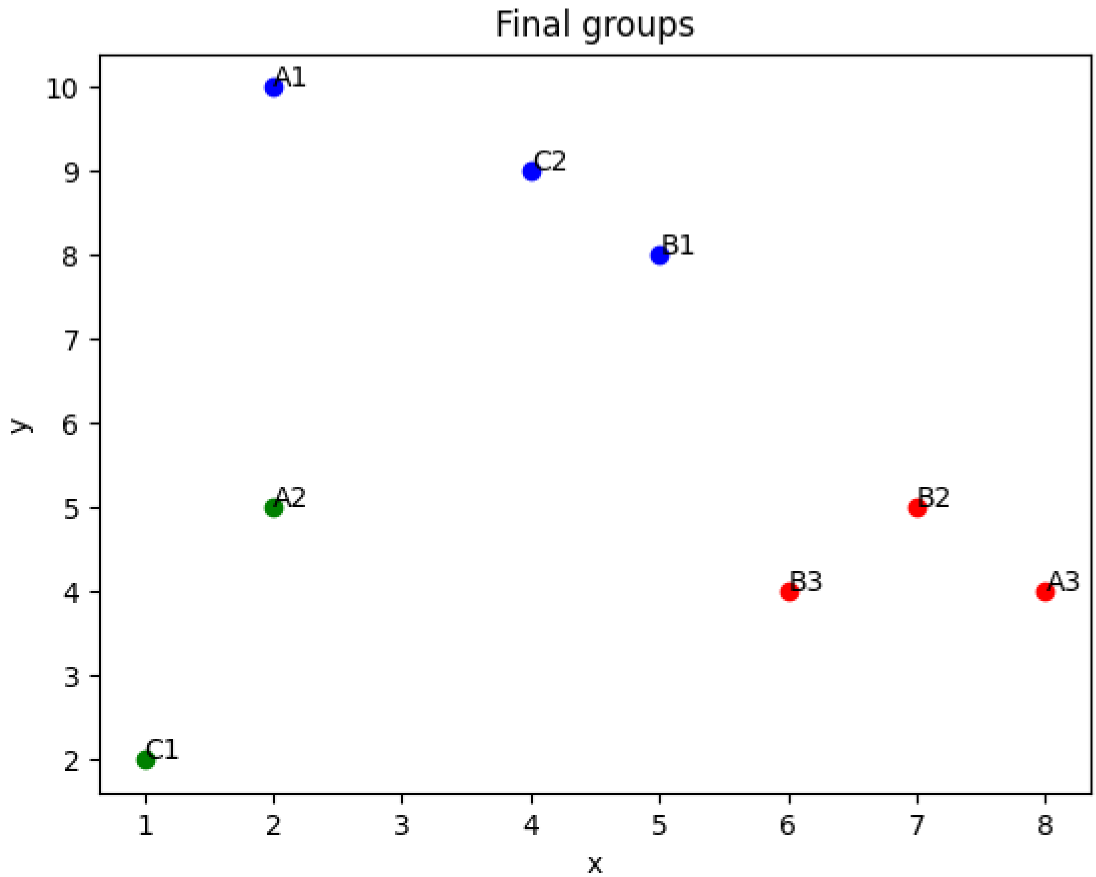
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

file_path=input("Enter the file path of the data: ")
df=pd.read_csv(file_path,delimiter=",")
print(df)
column_names=df.columns
dropped_numpy_df=df.drop(column_names[0],axis=1).to_numpy()
```

```
np.set_printoptions(formatter={'float':lambda x:"{0:0.3f}".format(x)})

k=int(input("Enter the number of groups to be formed: "))

assigned_coord={}
for grp_num in range(k):
    t=[]
    temp=input("G"+str(grp_num+1)+" should be assigned to "+str(column_names[0]).
lower()+": ")
    temp_df=df[df[column_names[0]]==temp]
    for i in range(1,len(column_names)):
        t.append(temp_df[column_names[i]].item())
    assigned_coord["G"+str(grp_num+1)]=t

print("NOTE: For distance matrix & group matrix, each row represents a group and
each column represents a point.")
previous_group_matrix=np.zeros((k,len(df)),dtype=np.int)
distance_matrix=np.zeros((k,len(df)))
count_iterations=0
while (True):
    group_idx=0
    for _,coord_list in assigned_coord.items():
        for p_idx in range(dropped_numpy_df.shape[0]):
            distance_matrix[group_idx][p_idx]=np.sqrt((coord_list[0]-
dropped_numpy_df[p_idx][0])**2+(coord_list[1]-dropped_numpy_df[p_idx][1])**2)

        group_idx+=1
    maximum_indices=np.argmin(distance_matrix,axis=0)
    counter=0
    next_group_matrix=np.zeros((k,len(df)),dtype=np.int)
    for column_idx in range(next_group_matrix.shape[1]):
        next_group_matrix[maximum_indices[counter]][column_idx]=1
        counter+=1
    print("-"*80)
    print("Iteration number "+str(count_iterations)+"- ")
    count_iterations+=1
    print("Distance matrix:")
    print(distance_matrix)
    print("Group matrix:")
    print(next_group_matrix)
    for i in range(next_group_matrix.shape[0]):
        num=0
        accumulator=[0,0]
        for j in range(next_group_matrix.shape[1]):
```

```
            if (next_group_matrix[i][j]==1):
                accumulator[0]+=dropped_numpy_df[j][0]
                accumulator[1]+=dropped_numpy_df[j][1]
                num+=1
            assigned_coord["G"+str(i+1)]=[val/num for val in accumulator]
    if (np.array_equal(previous_group_matrix,next_group_matrix)):
        print("Final distance and group matrix are displayed above.")
        break
    previous_group_matrix=next_group_matrix

group_assigned=np.argmax(next_group_matrix,axis=0)
uniques=np.unique(group_assigned)
colors=['bo','ro','go','co','mo','yo']
color_names=['blue','red','green','cyan','magenta','yellow']
plt.xlabel(column_names[1])
plt.ylabel(column_names[2])
plt.title("Final groups")
group_idx=0
for group_idx in range(len(uniques)):
    for idx in range(len(group_assigned)):
        if (uniques[group_idx]==group_assigned[idx]):
            plt.plot(dropped_numpy_df[idx][0],dropped_numpy_df[idx][1],colors[gro
up_idx])

plt.annotate(df.iloc[idx,0],(dropped_numpy_df[idx][0],dropped_numpy_df[idx][1]))
print("Group "+str(uniques[group_idx]+1)+" coloured:"+color_names[group_idx])
plt.show()
```

Output:

```
Enter the file path of the data: Exp5_dataset.csv
  Point  x   y
0    A1  2  10
1    A2  2   5
2    A3  8   4
3    B1  5   8
4    B2  7   5
5    B3  6   4
6    C1  1   2
7    C2  4   9
Enter the number of groups to be formed: 3
G1 should be assigned to point: A1
G2 should be assigned to point: B1
G3 should be assigned to point: C1
NOTE: For distance matrix & group matrix, each row represents a group and each column represents a point.
--------------------------------------------------------------------------------
Iteration number 0-
Distance matrix:
[[0.000 5.000 8.485 3.606 7.071 7.211 8.062 2.236]
 [3.606 4.243 5.000 0.000 3.606 4.123 7.211 1.414]
 [8.062 3.162 7.280 7.211 6.708 5.385 0.000 7.616]]
Group matrix:
[[1 0 0 0 0 0 0 0]
 [0 0 1 1 1 1 0 1]
 [0 1 0 0 0 0 1 0]]
--------------------------------------------------------------------------------


Iteration number 1-
Distance matrix:
[[0.000 5.000 8.485 3.606 7.071 7.211 8.062 2.236]
 [5.657 4.123 2.828 2.236 1.414 2.000 6.403 3.606]
 [6.519 1.581 6.519 5.701 5.701 4.528 1.581 6.042]]
Group matrix:
[[1 0 0 0 0 0 0 1]
 [0 0 1 1 1 1 0 0]
 [0 1 0 0 0 0 1 0]]
--------------------------------------------------------
Iteration number 2-
Distance matrix:
[[1.118 4.610 7.433 2.500 6.021 6.265 7.762 1.118]
 [6.543 4.507 1.953 3.132 0.559 1.346 6.388 4.507]
 [6.519 1.581 6.519 5.701 5.701 4.528 1.581 6.042]]
Group matrix:
[[1 0 0 1 0 0 0 1]
 [0 0 1 0 1 1 0 0]
 [0 1 0 0 0 0 1 0]]
--------------------------------------------------------
Iteration number 3-
Distance matrix:
[[1.944 4.333 6.616 1.667 5.207 5.518 7.491 0.333]
 [7.557 5.044 1.054 4.177 0.667 1.054 6.438 5.548]
 [6.519 1.581 6.519 5.701 5.701 4.528 1.581 6.042]]
Group matrix:
[[1 0 0 1 0 0 0 1]
 [0 0 1 0 1 1 0 0]
 [0 1 0 0 0 0 1 0]]
Final distance and group matrix are displayed above.
Group 1 coloured: blue
Group 2 coloured: red
Group 3 coloured: green
```



Conclusion:

1) Summary:
In this experiment, we learned about clustering in data mining, difference between clustering classification and K-means algorithm. We have implemented K-means graph in colab notebook in which we implemented the whole problem and also solved the sum and got the same output. We have also shown the graph which shows the clustering.

2) Importance:

i It is a simpler algorithm to implement.
ii It helps to scale large data set.
iii It guarantees convergence.
iv) It helps to warm start the positions of centroids.
v Easily adapts to new examples.
vi It generalizes to clusters of different shapes and sizes, such as elliptical clusters.

By implementing this algorithm, we understood how it works and we were able to plot a graph to visualize the cluster.

3) Applications -

i Get a meaningful intution of the structure of data we are dealing with.
ii It has various applications like market segmentation, document clustering, image

segmentation, image compression etc.

9 clusters then predict where different models will be built for different subgroups. If we believe there is a wide variation in the behaviour of different subgroups.