PART A

(PART A: TO BE REFFERED BY STUDENTS)

Experiment No. 5

A.1 Aim:

To implement K-means clustering

A.2 Prerequisite:

Python Basic Concepts

A.3 Outcome:

Students will be able To implement K-means clustering.

A.4 Theory:

K-means clustering is one of the most widely used unsupervised machine learning algorithms that forms clusters of data based on the similarity between data instances. For this particular algorithm to work, the number of clusters has to be defined beforehand. The K in the K-means refers to the number of clusters.

The K-means algorithm starts by randomly choosing a centroid value for each cluster. After that the algorithm iteratively performs three steps: (i) Find the Euclidean distance between each data instance and centroids of all the clusters; (ii) Assign the data instances to the cluster of the centroid with nearest distance; (iii) Calculate new centroid values based on the mean values of the coordinates of all the data instances from the corresponding cluster**.**

Hierarchical Based Methods : The clusters formed in this method forms a tree type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two category.

Agglomerative (bottom up approach)

Divisive (top down approach) .

Agglomerative Clustering:

Agglomerative algorithms start with each individual item in its own cluster and iteratively merge clusters until all items belong in one cluster. Different agglomerative algorithms differ in how the clusters are merged at each level. Outputting the dendrogram produces a set of clusters rather than just one clustering. The user can determine which of the clusters (based on distance threshold) he or she wishes to use.

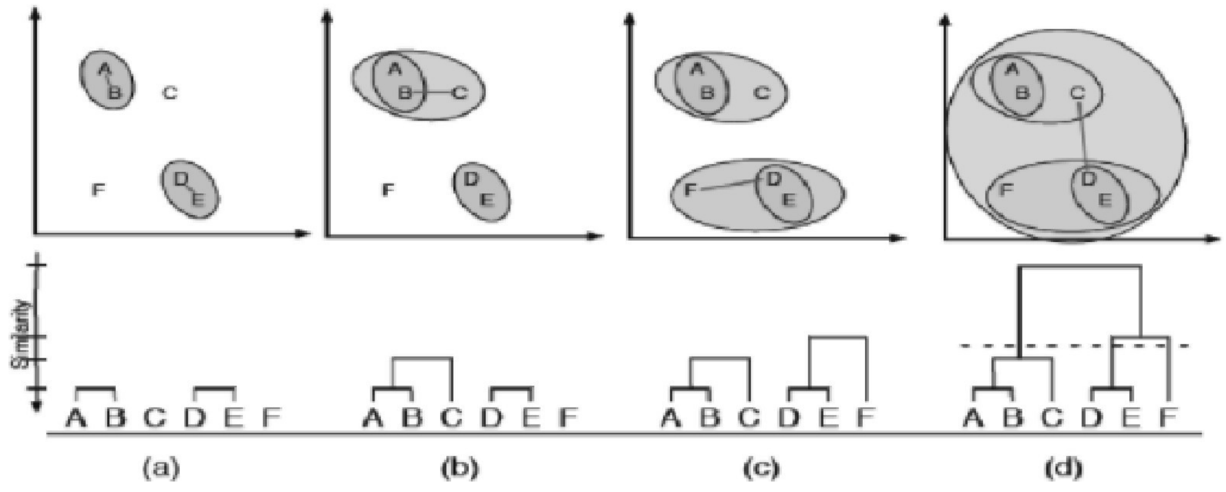
Agglomerative Algorithm

Compute the distance matrix between the input data points Let each data point be a cluster.

Repeat

Merge the two closest clusters

Update the distance matrix



Distance between two clusters Each cluster is a set of points. In following ways distance is defined in clusters.

Single Link:

Distance between clusters Ci and Cj is the minimum distance between any object in Ci and any object in Cj.

$$D_{\text{single}} = \min_{x,y}\{d(x,y) \mid x \in C_i, y \in C_j\}$$

Complete Link:

Distance between clusters Ci and Cj is the maximum distance between any object in Ci and any object in Cj

$$D_{\text{complete}} = \max_{x,y}\{d(x,y) \mid x \in C_i, y \in C_j\}$$

Average Link:

Distance between clusters Ci and Cj is the average distance between any object in Ci and any object in Cj

$$D_{\text{average}} = \frac{1}{|C_i| \times |C_j|} \sum_{x \in C_i, y \in C_j} d(x,y)$$

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

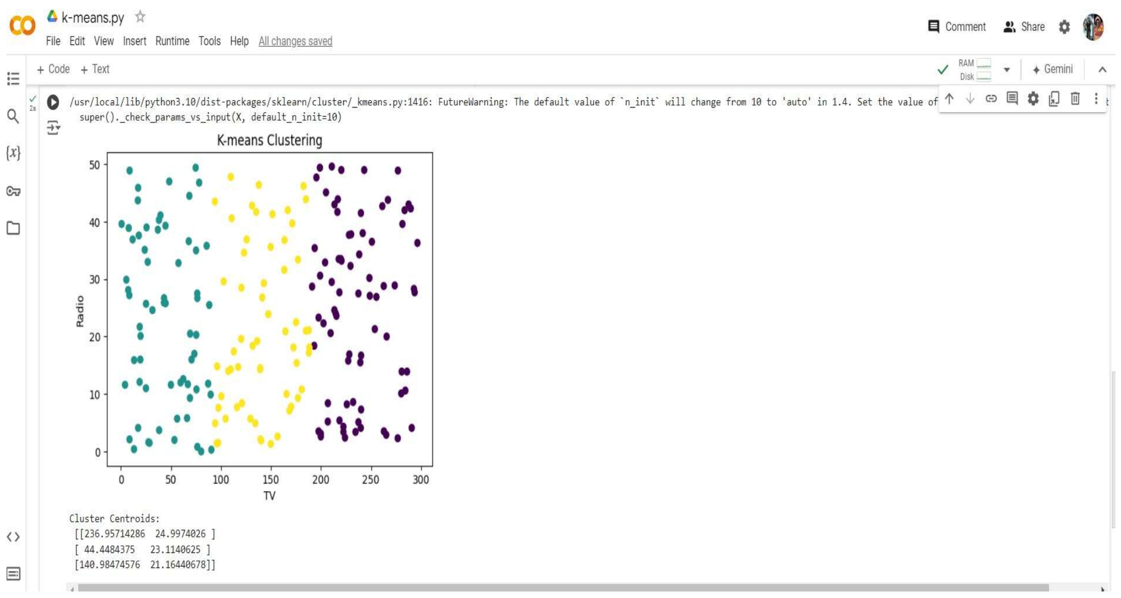
Roll. No. A15	Name: Khan Mohammad TAQI
Class: BE-Comps	Batch: A1
Date of Experiment	Date of Submission
Grade:	

B.1 Software Code written by student:

```
import pandas as pd
from sklearn.cluster import KMeans      #Harshad Chopra
import matplotlib.pyplot as plt

# Step 1: Load the dataset from a CSV file
# Replace 'your_file.csv' with the actual file path or name
data          =          pd.read_csv('/content/drive/MyDrive/Classroom/AI
class/Advertising.csv')

# Step 2: Explore the data (Optional)
# Uncomment the following line to see the first few rows of the data
# print(data.head())

# Step 3: Select features for clustering
#  Replace  'feature1'  and  'feature2'  with  actual  column  names  in  your
CSV file
X = data[['TV', 'Radio']]

# Step 4: Apply K-means clustering
kmeans = KMeans(n_clusters=3)  # You can change the number of clusters
as needed
kmeans.fit(X)

# Step 5: Assign cluster labels to the dataset
data['Cluster'] = kmeans.labels_
```

```python
# Step 6: Visualize the clusters (2D visualization)
plt.scatter(X.iloc[:,     0],     X.iloc[:,     1],     c=data['Cluster'],
cmap='viridis')
plt.xlabel('TV')
plt.ylabel('Radio')
plt.title('K-means Clustering')
plt.show()

# Step 7: Save the data with cluster labels (Optional)
# Replace 'output_file.csv' with the desired output file name
data.to_csv('output_file.csv', index=False)

# Observations: Comment on the cluster centroids and distribution
centroids = kmeans.cluster_centers_
print("Cluster Centroids:\n", centroids)
```

B.2 Input and Output:



```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
  super()._check_params_vs_input(X, default_n_init=10)
```

```
Cluster Centroids:
 [[236.95714286  24.9974026 ]
 [ 44.4484375   23.1140625 ]
 [140.98474576  21.16440678]]
```

B.3 Observations and learning:

The data points were divided into 3 clusters, each assigned to the nearest cluster centroid. The clusters represent distinct groupings based on the input features.

K-Means clustering is an unsupervised learning technique that effectively groups similar data points.

The choice of the number of clusters (K) is crucial and can significantly influence the results.

Standardizing the data before clustering can improve the algorithm's performance.

B.4 Conclusion:

The K-Means clustering algorithm successfully divided the data into distinct clusters, demonstrating its ability to group similar data points based on feature similarity. The experiment highlights the importance of proper data preprocessing and the selection of an appropriate number of clusters to achieve meaningful clustering results.

B.5 Question of Curiosity

(To be answered by student based on the practical performed and learning/observations)