DWM Exp. 4

Name – Kshitij Jadhav
Class-Div - TE-3-B
Roll no: 24

Aim: Implementation of Clustering Algorithm (K-means / Agglomerative) using Python

Introduction:

Clustering is an unsupervised machine learning technique used to group similar data points together. K-means clustering partitions the dataset into K clusters by minimizing intra-cluster variance, whereas Agglomerative clustering follows a hierarchical approach by merging or splitting clusters based on distance metrics.

Procedure:

1. Load the dataset.
2. Preprocess the data (if necessary).
3. Apply the K-means and Agglomerative clustering algorithms.
4. Evaluate the clusters formed.
5. Visualize the results.

Program Codes:

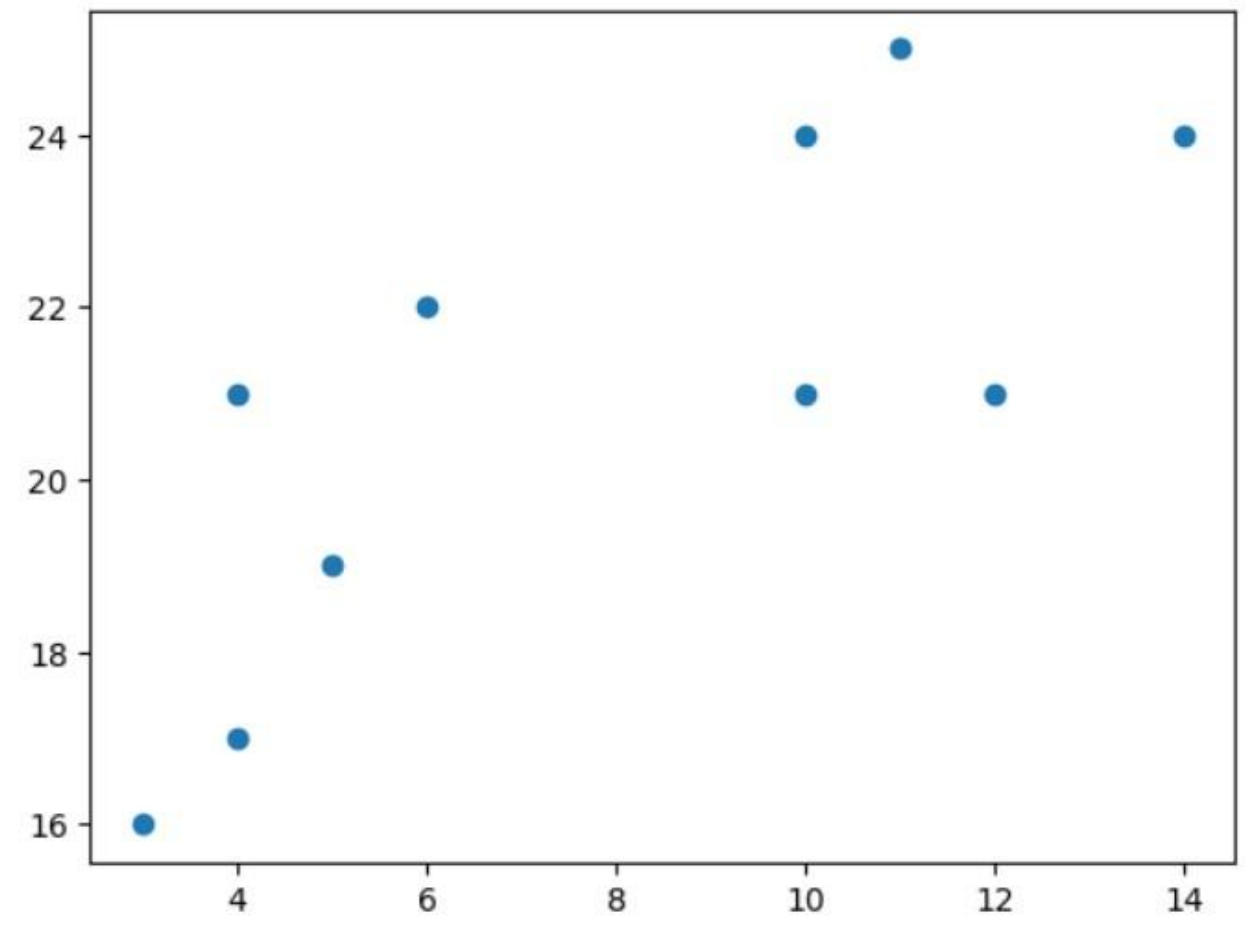
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, AgglomerativeClustering
import pandas as pd
import numpy as np

# K means algorithm on predefined data values.
x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
```

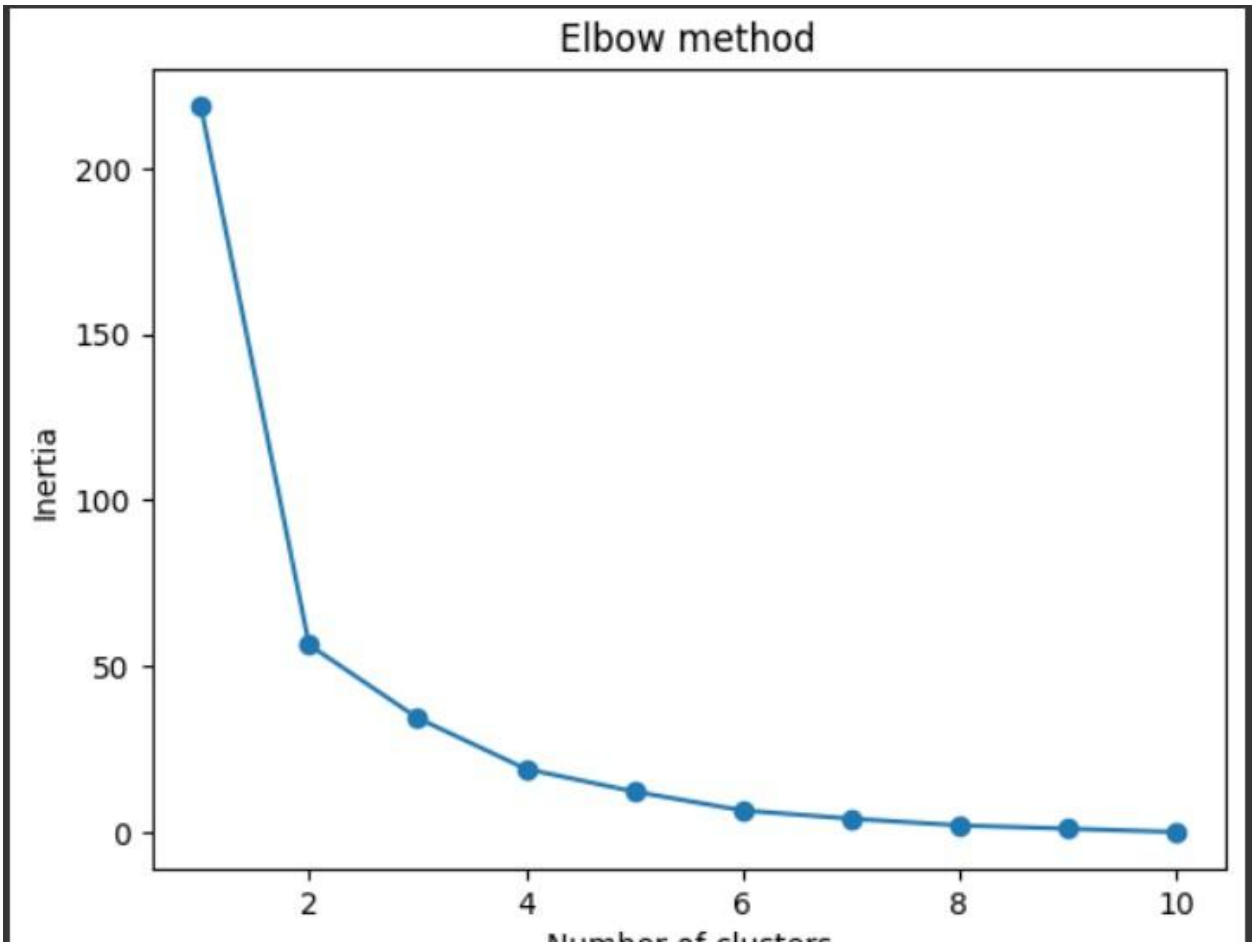
```
plt.scatter(x, y)
plt.show()

data = list(zip(x, y))
inertias = []

for i in range(1,11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(data)
    inertias.append(kmeans.inertia_)

plt.plot(range(1,11), inertias, marker='o')
plt.title('Elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()

kmeans = KMeans(n_clusters=2)
kmeans.fit(data)

plt.scatter(x, y, c=kmeans.labels_)
plt.show()

X = np.random.rand(100, 2)

kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
labels = kmeans.labels_
centroids = kmeans.cluster_centers_

# Visualize the clusters
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.scatter(centroids[:, 0], centroids[:, 1], marker='x', s=200,
linewidths=3, color='r')
plt.title('K-means Clustering')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()



# Agglomerative Clustering
```

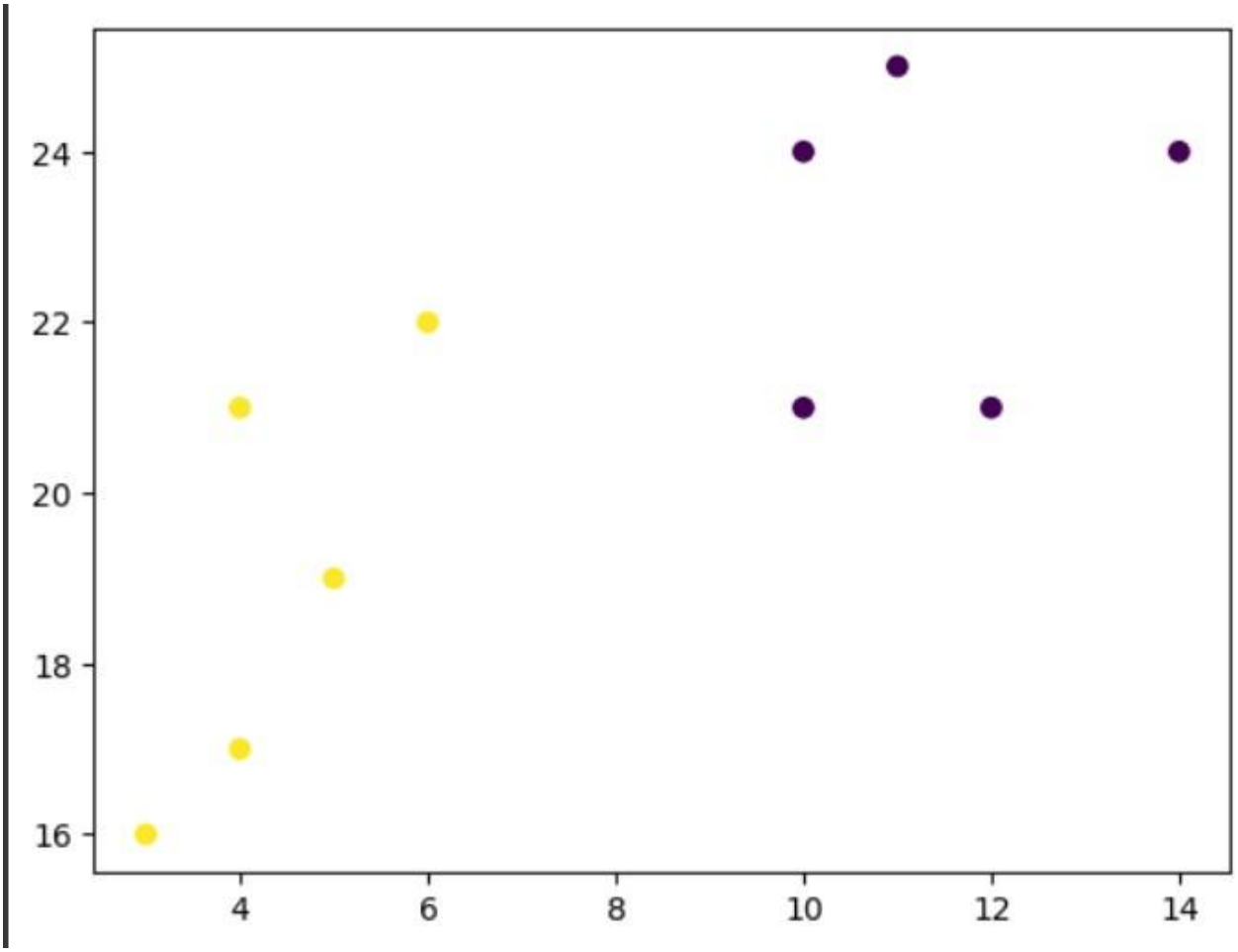
```
agg_clustering = AgglomerativeClustering(n_clusters=3)
agg_labels = agg_clustering.fit_predict(X)

# Visualize the agglomerative clustering results
plt.scatter(X[:, 0], X[:, 1], c=agg_labels)
plt.title('Agglomerative Clustering')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

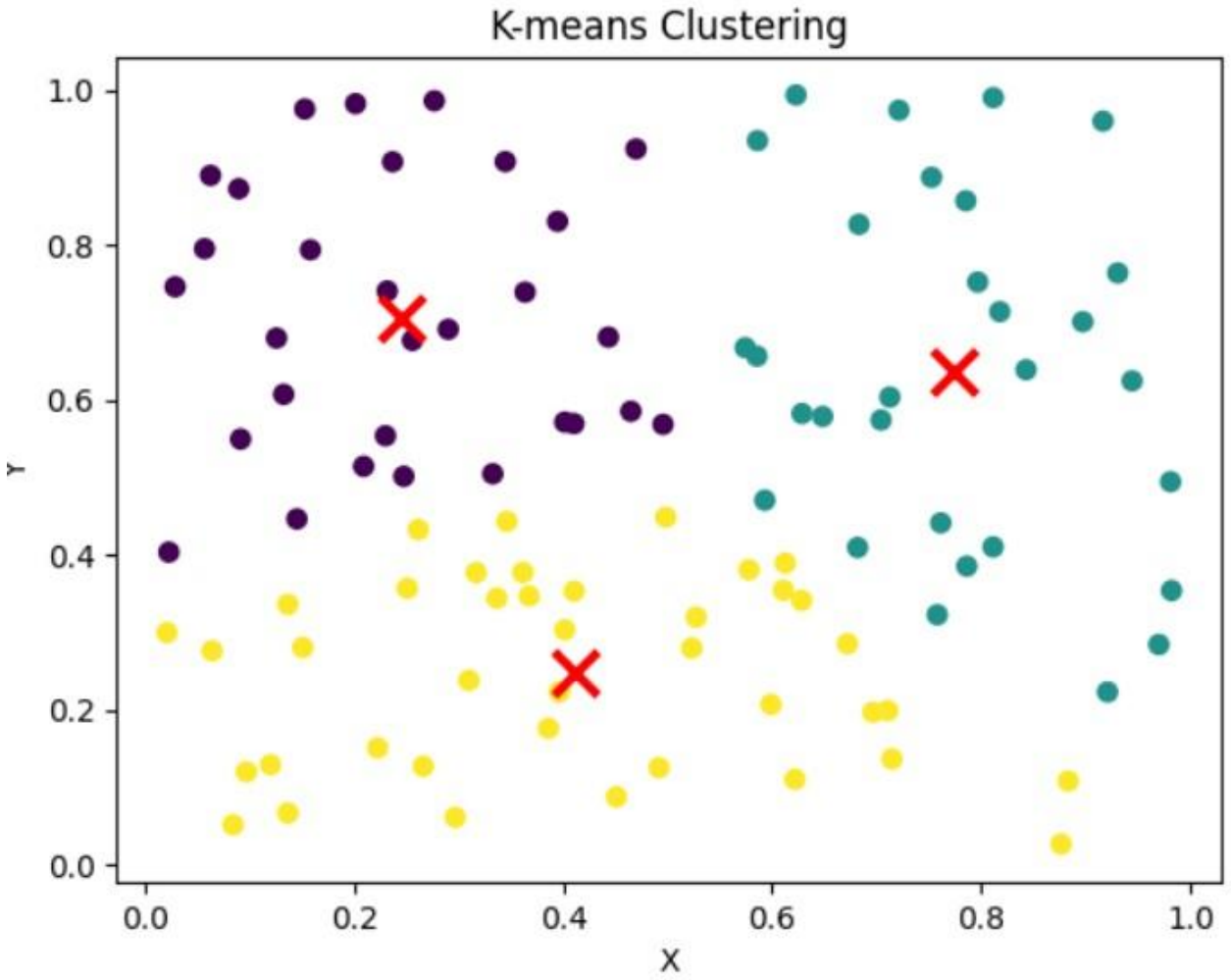
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.preprocessing import StandardScaler


X, _ = make_blobs(n_samples=300, centers=4, cluster_std=1.0,
random_state=42)
X = StandardScaler().fit_transform(X)

# K-Means Clustering
kmeans = KMeans(n_clusters=4, random_state=42)
kmeans_labels = kmeans.fit_predict(X)

# Agglomerative Clustering
agglo = AgglomerativeClustering(n_clusters=4)
agglo_labels = agglo.fit_predict(X)

# Plot results
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
ax1.scatter(X[:, 0], X[:, 1], c=kmeans_labels, cmap='viridis', marker='o')
ax1.set_title("K-Means Clustering")
ax2.scatter(X[:, 0], X[:, 1], c=agglo_labels, cmap='plasma', marker='o')
ax2.set_title("Agglomerative Clustering")
plt.show()
```

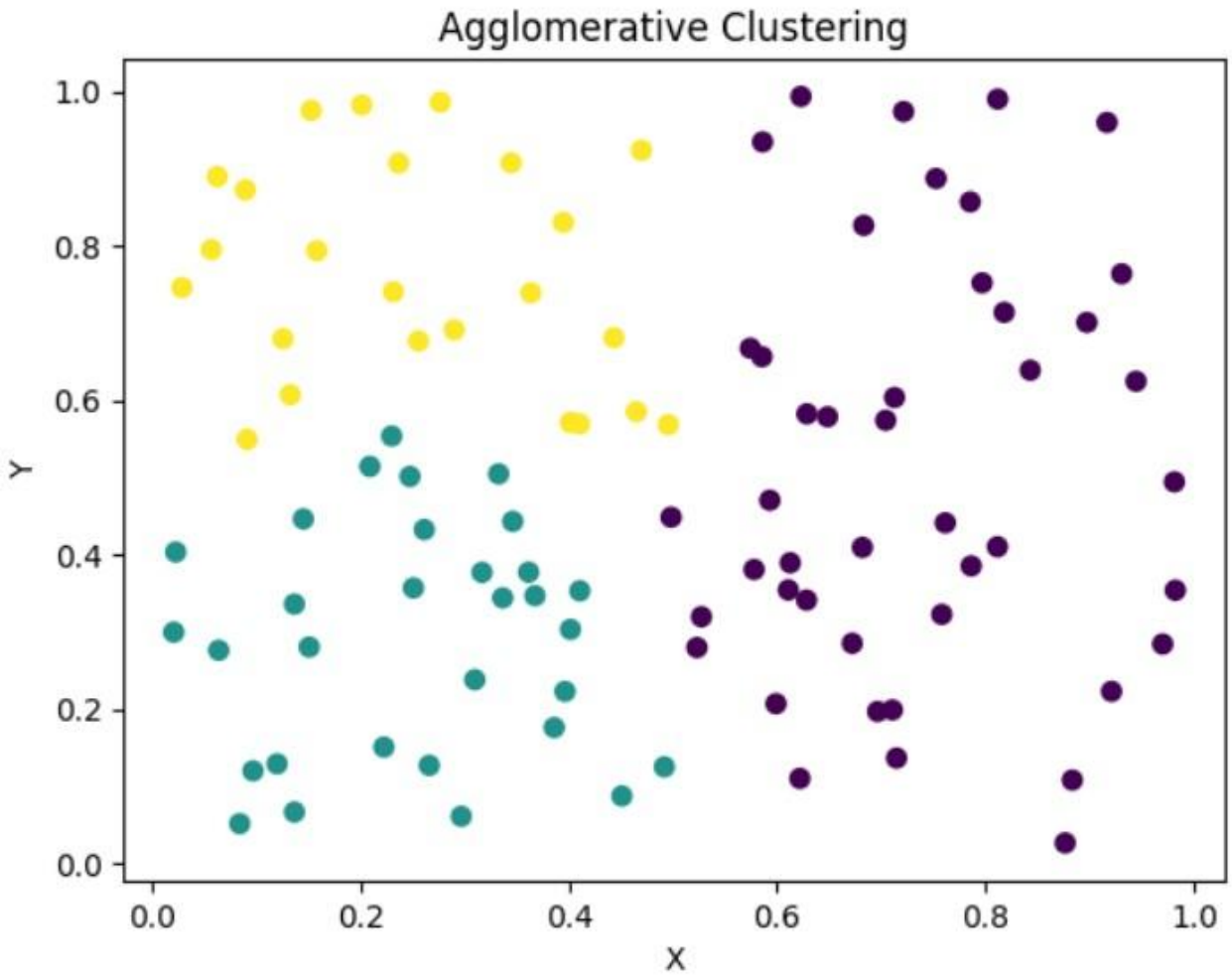
Implementation/Output snapshot:

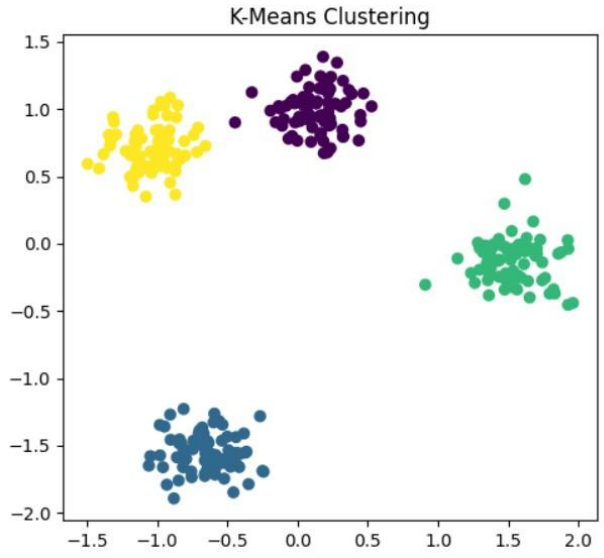

24
22
20
18
16
4
6
8
10
12
14

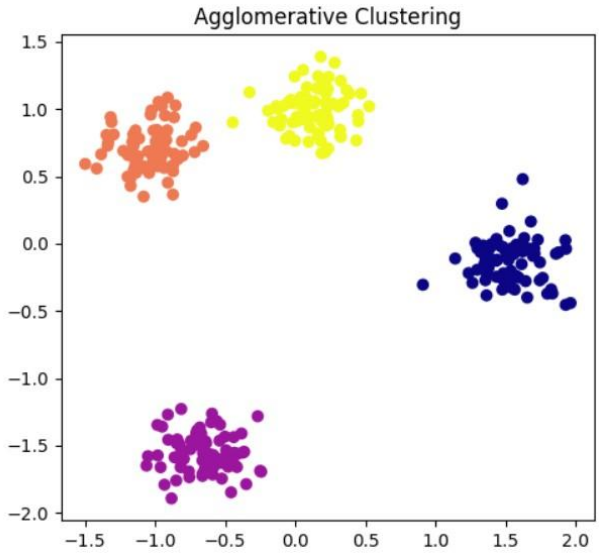

Elbow method
Inertia
200
150
100
50
0
2
4
6
8
10


24
22
20
18
16
4
6
8
10
12
14


K-means Clustering
1.0
0.8
0.6
0.4
0.2
0.0
Y
0.0
0.2
0.4
0.6
0.8
1.0
X







Conclusion: K-means and Agglomerative clustering effectively group data points into clusters. K-means is computationally efficient but requires specifying the number of clusters, while Agglomerative clustering is

hierarchical and does not require pre-specifying cluster numbers but can be computationally expensive.

Review Questions:

1. What is the K-means clustering algorithm, and how does it work?

Ans. K-means is an unsupervised clustering algorithm that partitions a dataset into K clusters. It works as follows:

- Initialize K cluster centroids randomly.
- Assign each data point to the nearest centroid.
- Compute the new centroids as the mean of all points in each cluster.
- Repeat the assignment and centroid update steps until centroids stabilize or a stopping criterion is met.

2. How do you determine the optimal number of clusters in K-means?

Ans. The optimal number of clusters can be determined using:

- **Elbow Method**: Plot the within-cluster sum of squares (WCSS) against the number of clusters and look for the "elbow point."
- **Silhouette Score**: Measures how well-separated the clusters are, with higher values indicating better clustering.
- **Gap Statistic**: Compares clustering performance with a random reference distribution.

3. What are the common distance metrics used in Agglomerative Clustering?

Ans. Common distance metrics include:

- Euclidean Distance (default) – Measures straight-line distance.
- Manhattan Distance – Measures distance along grid paths.
- Cosine Similarity – Measures the cosine of the angle between vectors.