Name : Sonika Kulkarni

Div : TY3

Batch : B

Roll no : 34

Subject : Data WareHouse And Mining

Experiment 4 : Implementation of Clustering Algorithm (K-means / Agglomerative) using Python

Aim :

To implement and understand the working of clustering algorithms such as K-means and Agglomerative Clustering using Python.

Introduction:

Clustering is an unsupervised machine learning technique used to group similar data points together. Two popular clustering algorithms are:

K-means Clustering: Partitions data into k clusters by minimizing the variance within each cluster.

Agglomerative Clustering: A hierarchical clustering method that builds nested clusters by merging or splitting them successively.

Procedure:

Data Preparation: Load or generate a dataset suitable for clustering.

Algorithm Implementation:

For K-means: Use the KMeans class from sklearn.cluster.

For Agglomerative Clustering: Use the AgglomerativeClustering class from sklearn.cluster.

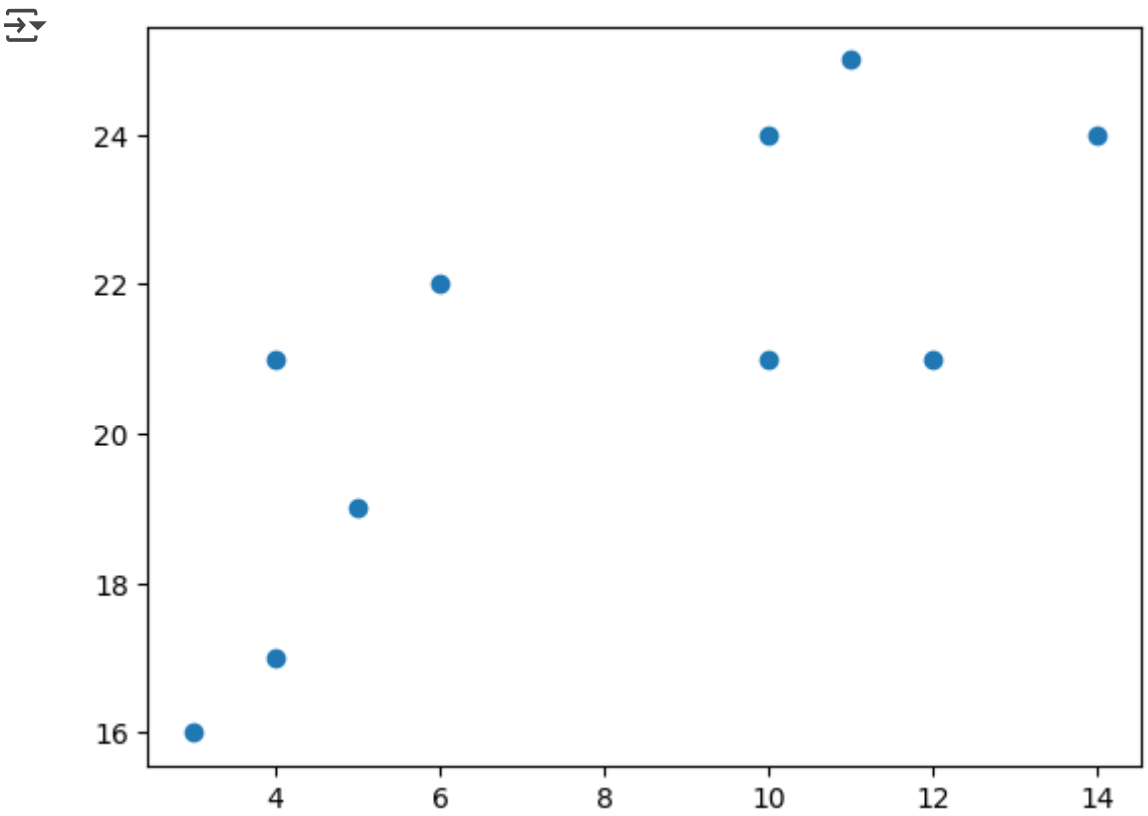
Visualization: Plot the clusters to visualize the results.

Evaluation: Use metrics like silhouette score or inertia to evaluate clustering performance.

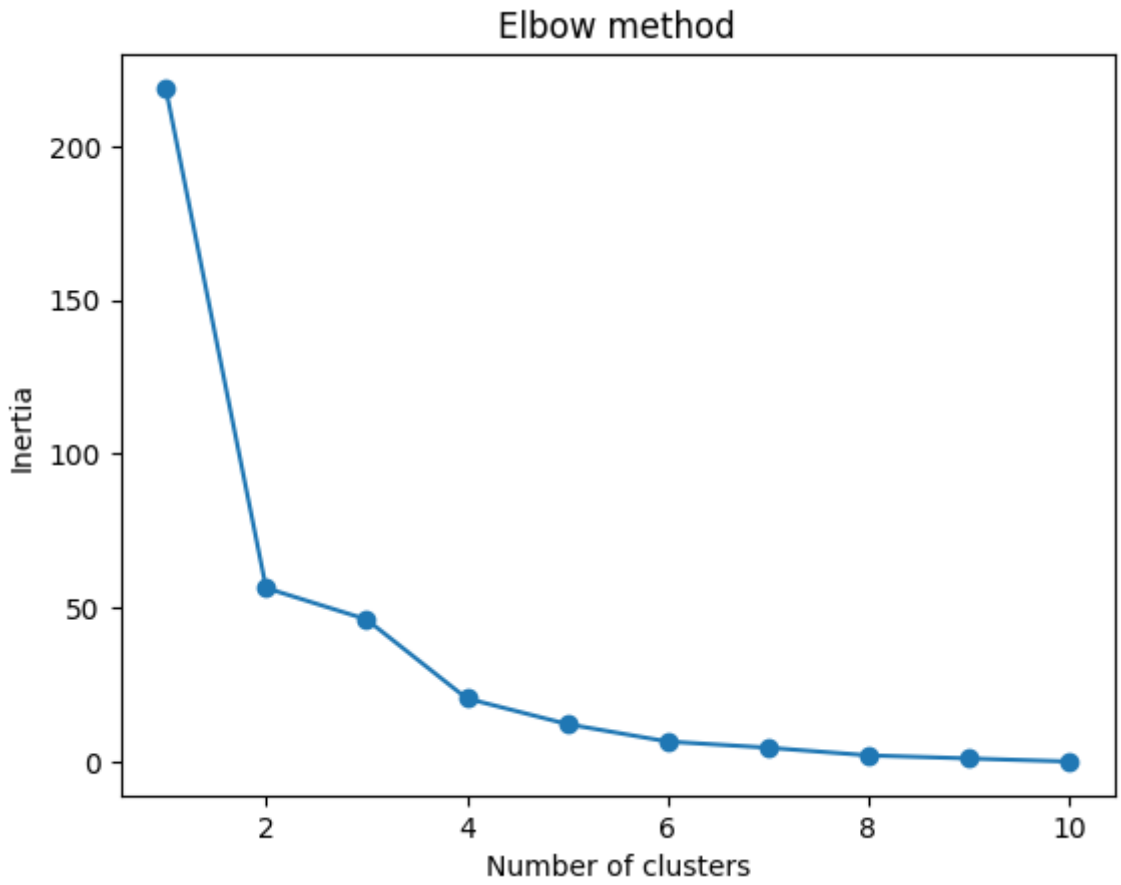
Program Codes:

K-means Clustering:

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt


x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]


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



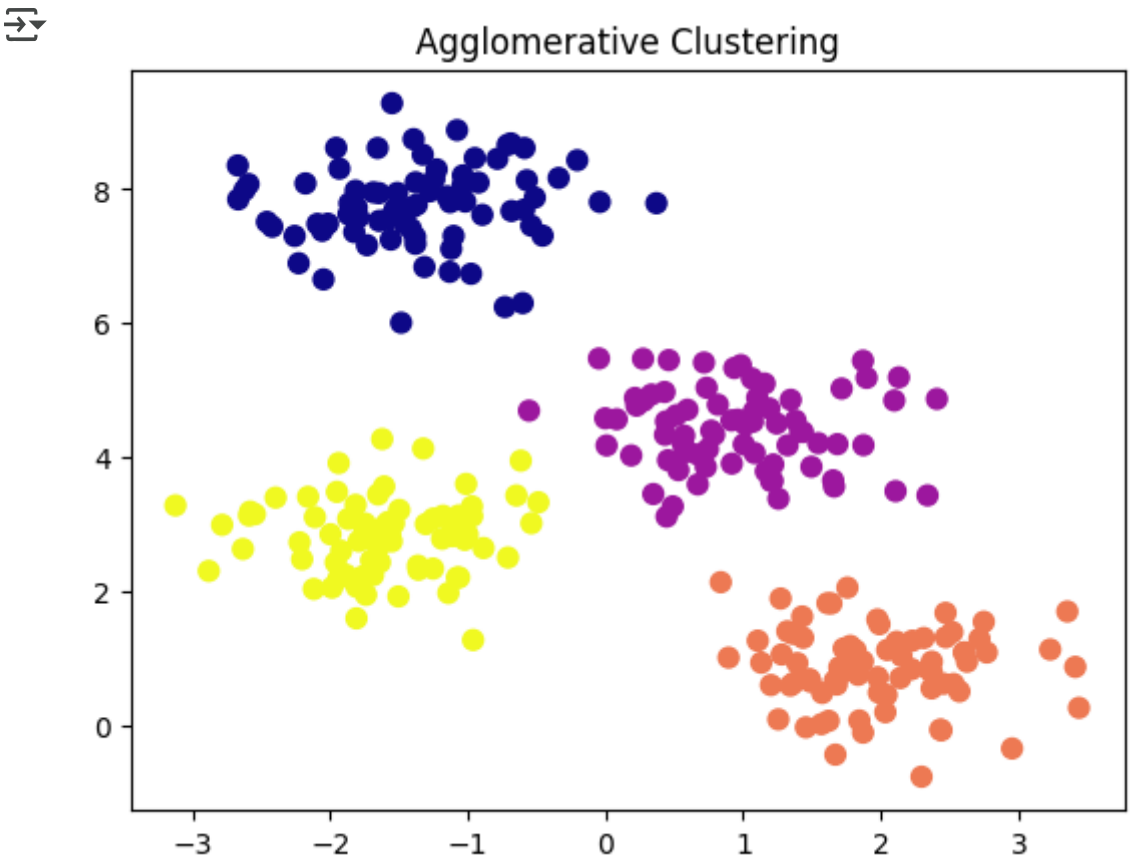

```
plt.plot(range(1,11), inertias, marker='o')
plt.title('Elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```



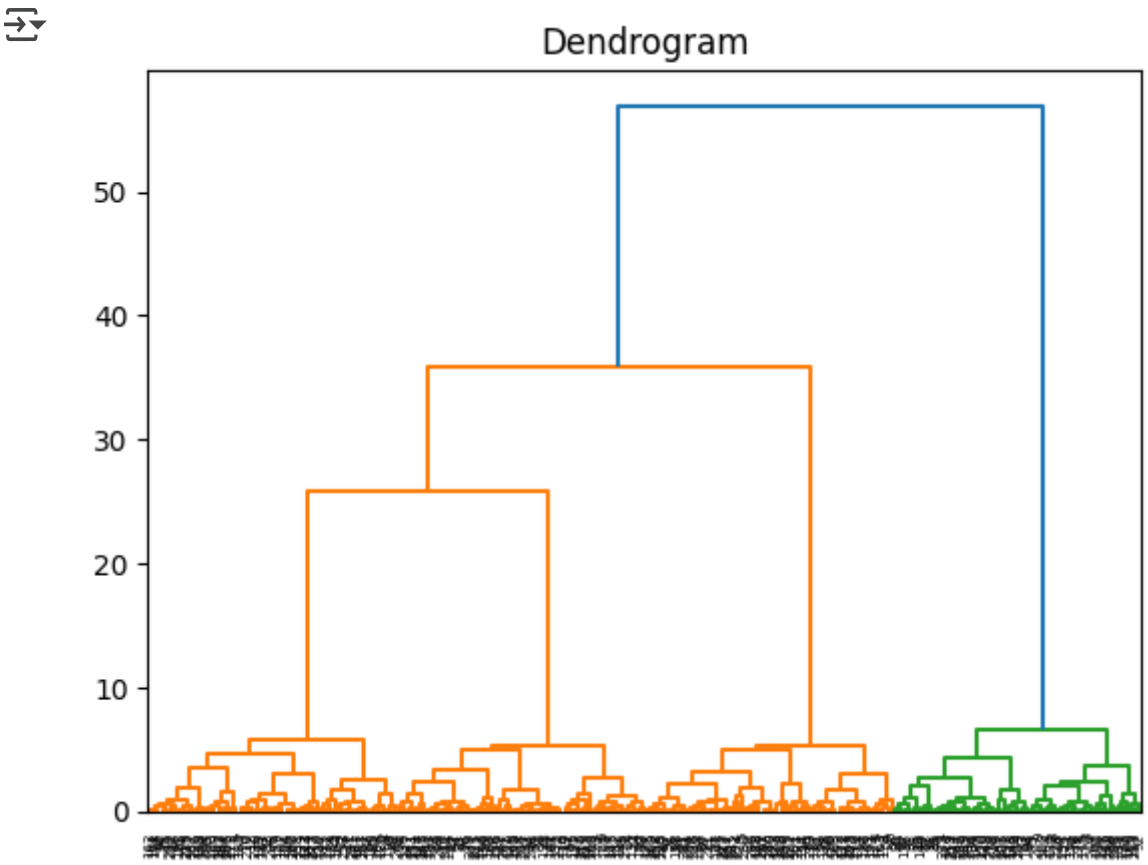
Agglomerative Clustering:

```
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.datasets import make_blobs


# Generate sample data
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)
```

```
# Generate sample data
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)


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


# Plot the clusters
plt.scatter(X[:, 0], X[:, 1], c=y_agglo, s=50, cmap='plasma')
plt.title("Agglomerative Clustering")
plt.show()
```




```
# Dendrogram
linked = linkage(X, 'ward')
dendrogram(linked, orientation='top', distance_sort='descending', show_leaf_counts=True)
plt.title("Dendrogram")
plt.show()
```



Conclusion:

- K-means is efficient for large datasets and works well when clusters are spherical.
- Agglomerative Clustering provides a hierarchical structure and is useful for understanding data relationships.
- Both algorithms are powerful tools for unsupervised learning and pattern recognition.

Review Questions:

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

The K-means clustering algorithm is an unsupervised machine learning technique used to partition a dataset into k distinct, non-overlapping clusters. The goal of the algorithm is to group similar data points together while keeping different clusters as distinct as possible.

How it works:

Initialization:

Choose the number of clusters k.

Randomly initialize k cluster centroids (points in the feature space).

Assignment Step:

Assign each data point to the nearest centroid based on a distance metric (usually Euclidean distance).

This forms k clusters.

Update Step:

Recalculate the centroids of the clusters by taking the mean of all data points assigned to each cluster.

Iteration:

Repeat the assignment and update steps until convergence is reached (i.e., when the centroids no longer change significantly or a maximum number of iterations is reached).

Key Points:

- K-means minimizes the within-cluster sum of squares (WCSS), which is the sum of squared distances between data points and their assigned centroids.
- It is sensitive to the initial placement of centroids, which can lead to suboptimal solutions. Techniques like K-means++ are used to improve initialization.
- K-means works best when clusters are spherical and evenly sized.

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

Determining the optimal number of clusters (k) is a critical step in K-means clustering. Two common methods are used:

Elbow Method:

Plot the within-cluster sum of squares (WCSS) or inertia against the number of clusters (k).

Inertia measures how tightly the data points are grouped within clusters. As k increases, inertia decreases.

The "elbow point" in the plot (where the rate of decrease in inertia slows down significantly) is considered the optimal number of clusters.

Limitation: The elbow point is not always clear, making it subjective.

Silhouette Score:

The silhouette score measures how similar a data point is to its own cluster compared to other clusters.

It ranges from -1 to 1, where:

A score close to 1 indicates that the data point is well-matched to its own cluster and poorly matched to neighboring clusters.

A score close to 0 indicates overlapping clusters.

A negative score suggests incorrect clustering.

The optimal k is the one that maximizes the average silhouette score across all data points.

Advantage: Provides a more objective measure compared to the elbow method.

Gap Statistic:

Compares the total within intra-cluster variation for different numbers of clusters with their expected values under null reference distribution (random data).

The optimal k is the smallest value where the gap statistic is maximized.

Domain Knowledge:

Sometimes, the choice of k is guided by prior knowledge of the dataset or the problem domain.

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

Common metrics include:

- Euclidean distance
- Manhattan distance
- Cosine similarity
- Ward's method (minimizes variance when merging clusters).

Github

https://github.com/sonikak19/DWM.git