Ex.No – 9 Roll No – 210701149

Implement clustering techniques - Hierarchical and K-Means

AIM:

To implement clustering techniques such as hierarchical and k-means algorithms in python.

PROCEDURES:

1. Collect and load the dataset from sources like CSV files or databases.
2. Clean and preprocess the data, including handling missing values and scaling features.
3. Determine the number of clusters (K) for K-Means, or decide on the stopping criterion for Hierarchical Clustering.
4. Choose the appropriate clustering algorithm: K-Means for partitioning, Hierarchical for nested clustering.
5. Apply the K-Means algorithm using fit_predict to assign data points to clusters.
6. Apply the Hierarchical Clustering algorithm using AgglomerativeClustering for hierarchical clusters.
7. Visualize the clusters with scatter plots for K-Means, and dendrograms for Hierarchical Clustering.
8. Evaluate clustering performance using metrics like silhouette score or inertia (for K-Means).
9. Fine-tune the clustering by adjusting the number of clusters or linkage criteria.

10. Interpret the results to understand the structure and relationships within the data.

CODE:

Hierarchical.py

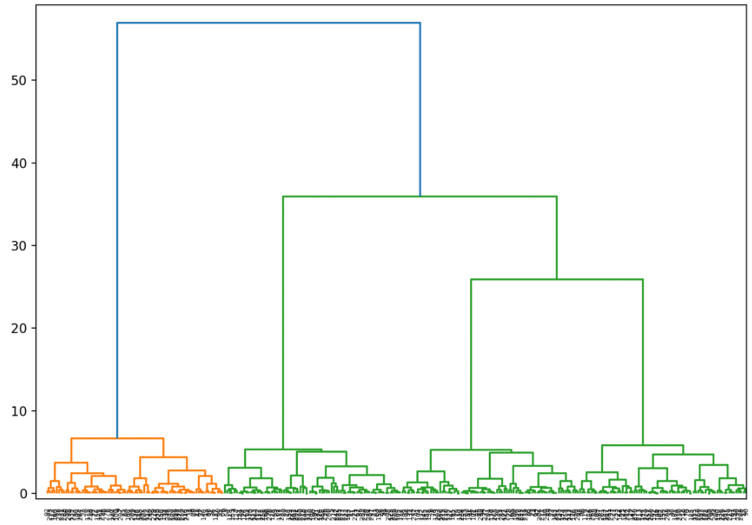
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering

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

# Create a dendrogram
linked = linkage(X, 'ward')

plt.figure(figsize=(10, 7))
dendrogram(linked)
plt.show()

# Apply AgglomerativeClustering with 4 clusters
hc = AgglomerativeClustering(n_clusters=4, metric='euclidean',
linkage='ward')
```

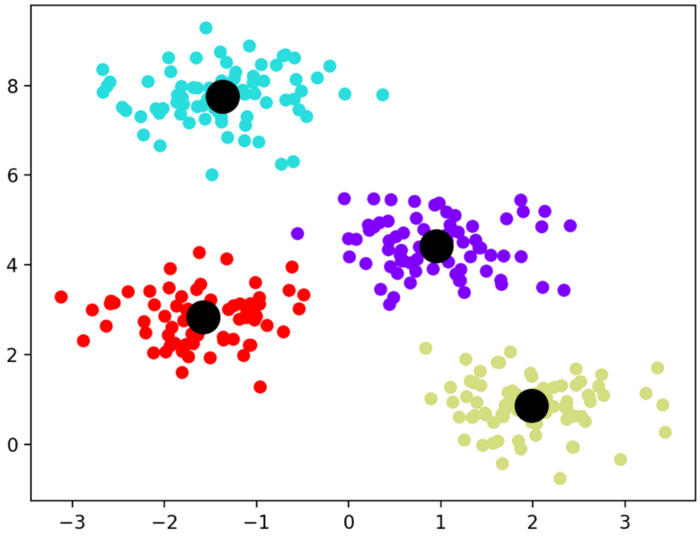
```
y_hc = hc.fit_predict(X)

# Plot the clusters
plt.scatter(X[:, 0], X[:, 1], c=y_hc, cmap='rainbow')
plt.show()
```

kmeans.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

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

# Apply KMeans with 4 clusters
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)

# Plot the clusters
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
s=300, c='black')
plt.show()
```

OUTPUT:





RESULT:

Thus, to implement hierarchical and kmeans clustering techniques are completed successfully.