Ex. No. 06	AGGLOMERATIVE CLUSTERING ALGORITHM
06.07.2024	

Aim

Develop agglomerative clustering models to cluster the given dataset using the scikit-learn.

Description

Hierarchical clustering is a type of clustering algorithm used in unsupervised machine learning to group similar data points together in a hierarchical tree-like structure, also known as a dendrogram. The algorithm starts by treating each data point as a separate cluster, and then iteratively merges the closest pair of clusters until a stopping criterion is met.

Hierarchical clustering can be performed using two approaches:

Agglomerative hierarchical clustering: In this approach, each data point is initially considered as a separate cluster and then, at each step, the two closest clusters are merged into a larger cluster. This process is continued until all data points belong to a single cluster. This is a bottom-up approach.

Divisive hierarchical clustering: In this approach, all data points are initially considered as belonging to a single cluster, and then at each step, the cluster is recursively split into smaller clusters based on some distance metric, until each cluster contains only one data point. This is a top-down approach.

Dendrogram: A Useful Tool for Summarizing Similarity Measurements

- Min Distance: Minimum distance of two points.
- Max Distance: Maximum distance of two points.
- Group Average: Average of distance between every two points of the cluster.
- Ward’s method: Similarity is based on the increase in square when two clusters are merged.

Source Code

1. Pre-process the data and fill the missing values and apply normalization

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScale
data=pd.read_csv(hopping-data.csv')
data.fillna(data.mean(), inplace=True)
scaler = StandardScaler()

data_normalized = scaler.fit_transform(data.select_dtypes(include=[np.number]))

data_normalized_df = pd.DataFrame(data_normalized,
columns=data.select_dtypes(include=[np.number]).columns)

non_numeric_cols = data.select_dtypes(exclude=[np.number]).columns

data_normalized_df = pd.concat([data_normalized_df,
data[non_numeric_cols].reset_index(drop=True)], axis=1)

print(data_normalized_df.head())
```

2. Apply label encoding to convert the categorical values to numerical values

```
from sklearn.preprocessing import LabelEncoder
encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    encoders[column] = LabelEncoder()
    data[column] = encoders[column].fit_transform(data[column])
```

OUTPUT:

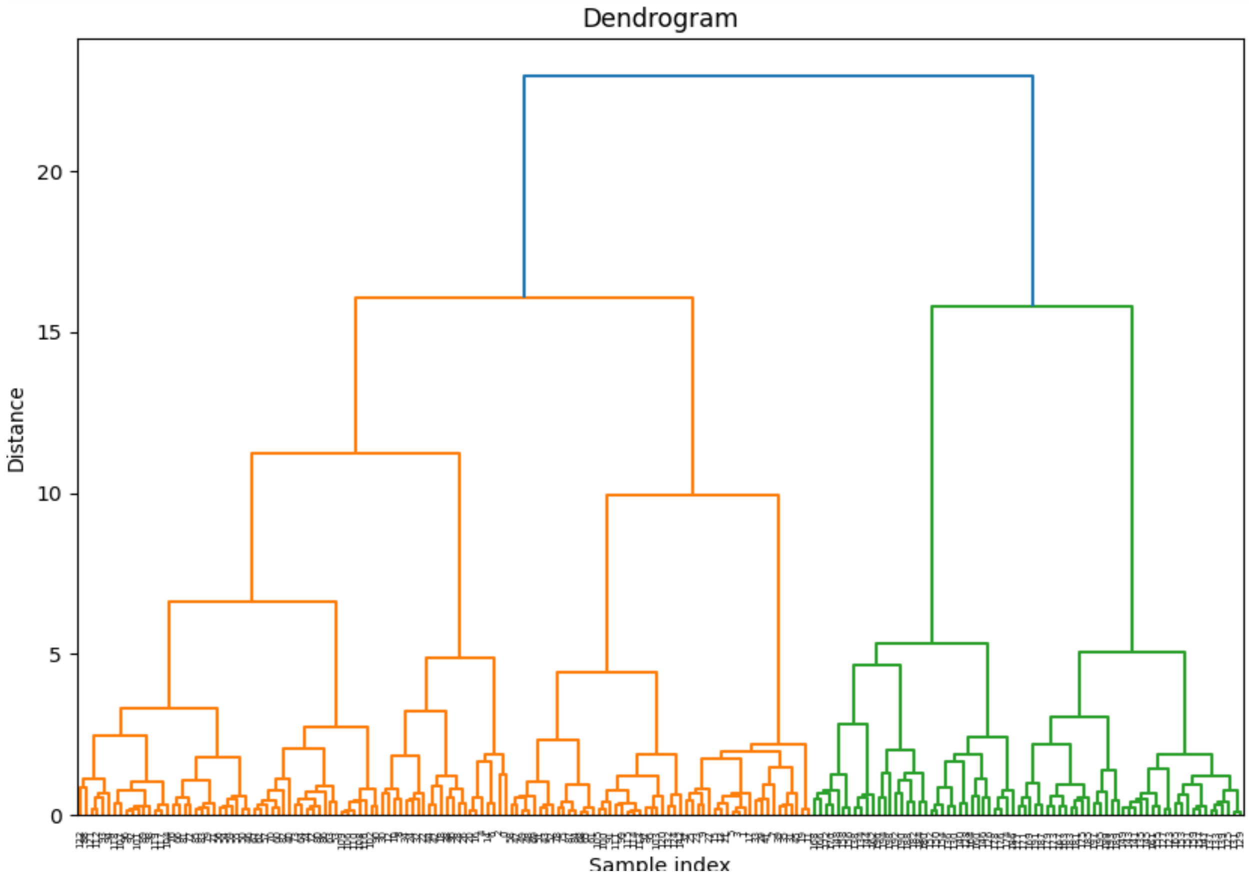
	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19	15	39
1	2	21	15	81
2	3	20	16	6
3	4	23	16	77
4	5	31	17	40

3. Plot the dendrogram

```
import matplotlib.pyplot as plt

from scipy.cluster.hierarchy import dendrogram, linkage
def plot_dendrogram(data):
    linked = linkage(data, method='ward')  # You can change the method here
    plt.figure(figsize=(10, 7))
    dendrogram(linked, orientation='top', distance_sort='descending', show_leaf_counts=True)
    plt.title('Dendrogram')
    plt.xlabel('Sample index')
    plt.ylabel('Distance')
    plt.show()
plot_dendrogram(data_normalized)
```

OUTPUT:



4. Implement agglomerative clustering algorithms to cluster the given data for different distance metrics (Euclidean, Manhattan, Cosine, L1, L2) and linkage functions (single, complete, average, wards).

```
from sklearn.cluster import AgglomerativeClustering
distance_metrics = ['euclidean', 'manhattan', 'cosine']
linkage_methods = ['ward', 'single', 'complete', 'average']
results = {}
for metric in distance_metrics:

    for linkage_method in linkage_methods:
```

```
        if metric == 'cosine' and linkage_method == 'ward':
            continue # Skip incompatible combinations
        try:

            clustering = AgglomerativeClustering(n_clusters=3, affinity=metric,
linkage=linkage_method)

            cluster_labels = clustering.fit_predict(data_normalized)
            results[(metric, linkage_method)] = cluster_labels
        except ValueError as e:

            print(f"Error occurred for metric: {metric}, linkage: {linkage_method}")
            print(e)
```

OUTPUT:

```
Error occurred for metric: manhattan, linkage: ward
manhattan was provided as metric. Ward can only work with euclidean distances.
```

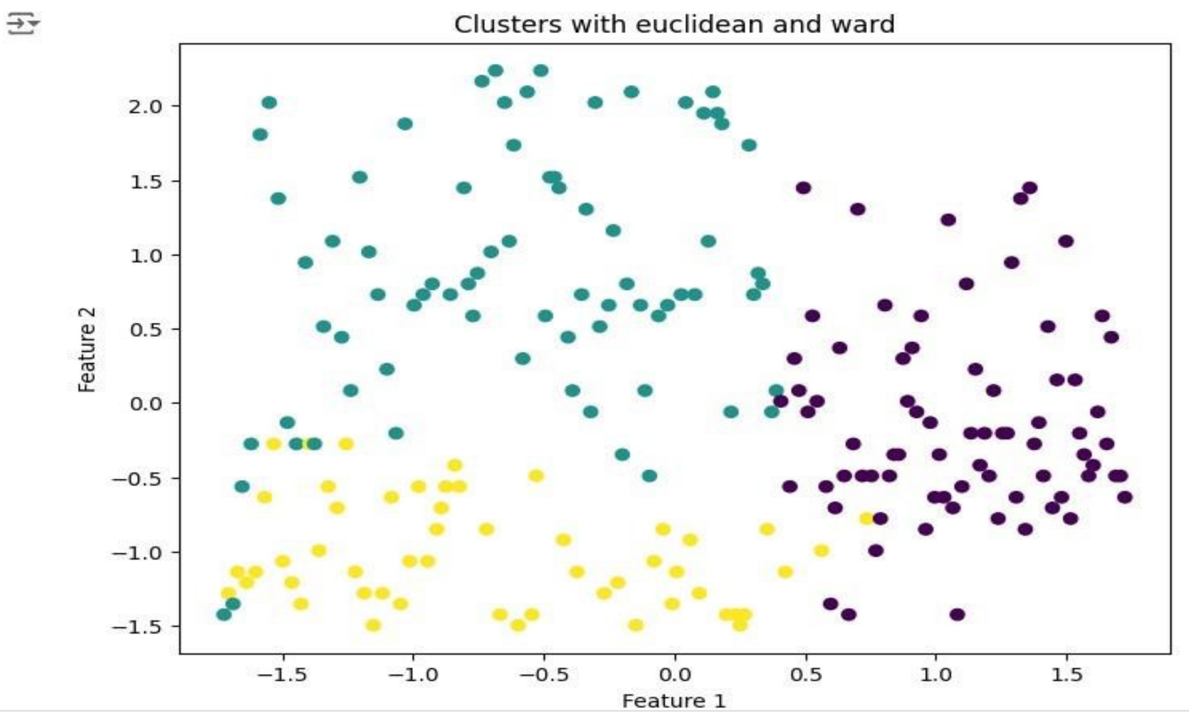
5. Try with the whole dataset (except the label) and with any 2 attributes

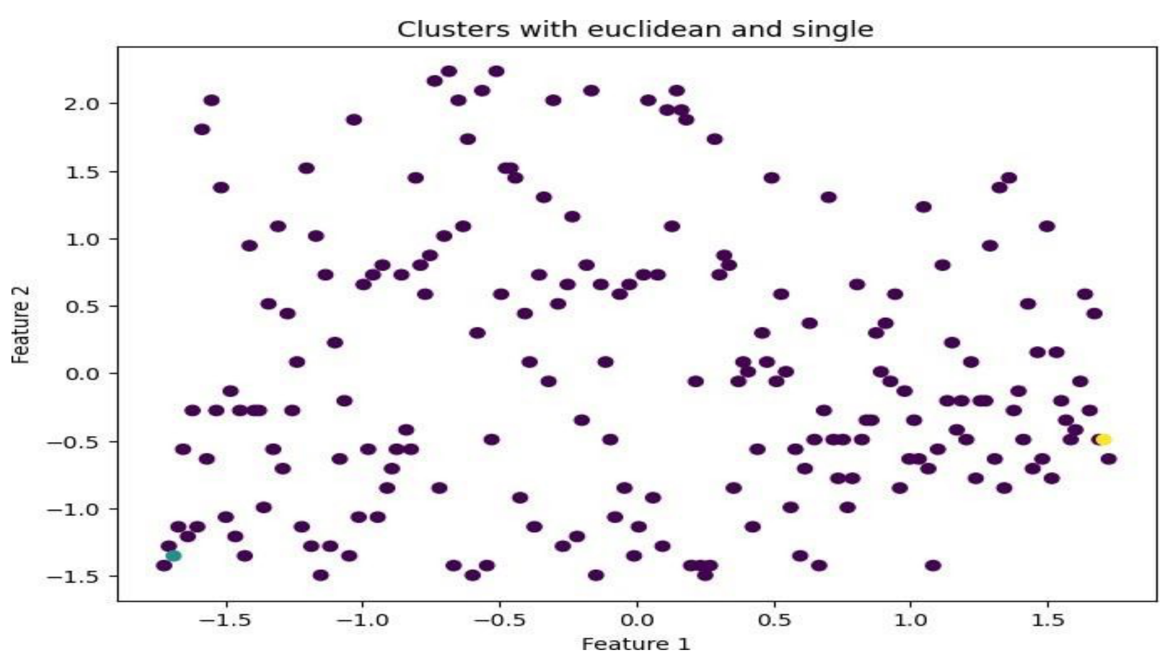
```
def plot_scatter(data, cluster_labels, title):
    plt.figure(figsize=(8, 6))
    plt.scatter(data[:, 0], data[:, 1], c=cluster_labels, cmap='viridis', marker='o')
    plt.title(title)
    plt.xlabel('Feature 1')

    plt.ylabel('Feature 2')
    plt.show()
for (metric, linkage_method), cluster_labels in results.items():
```

plot_scatter(data_normalized[:, :2], cluster_labels, f'Clusters with {metric} and {linkage_method}')

OUTPUT:



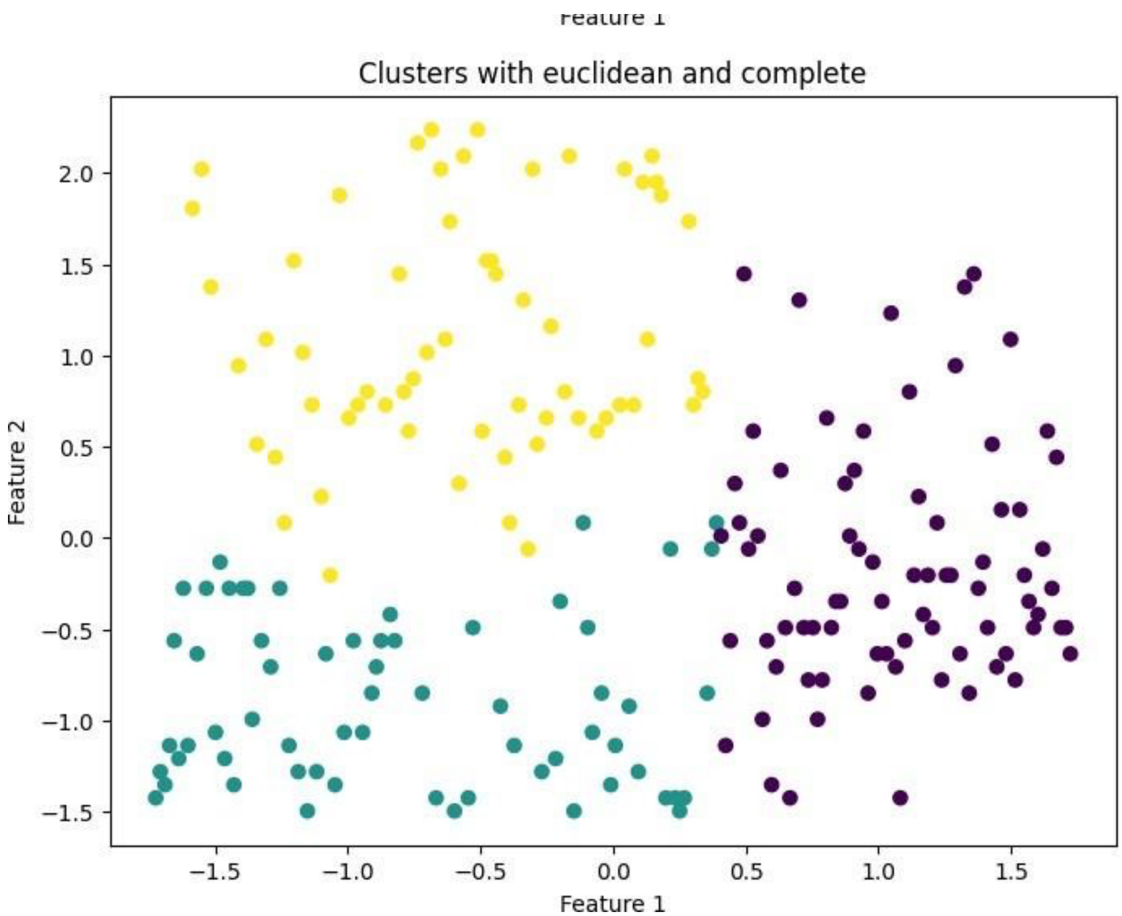


6. Analyse the results using scatter plot

```
def plot_scatter(data, cluster_labels, title):
    plt.figure(figsize=(8, 6))
    plt.scatter(data[:, 0], data[:, 1], c=cluster_labels, cmap='viridis', marker='o')
    plt.title(title)
    plt.xlabel('Feature 1')

    plt.ylabel('Feature 2')
    plt.show()
```

OUTPUT:



7. Compare the results using Mutual information, Silhouette Score (Silhouette Coefficient), Davies-Bouldin Index.

```
From sklearn.metrics import silhouette_score, davies_bouldin_score, mutual_info_score
 true_labels = data['true_label_column_name']
for (metric, linkage_method), cluster_labels in results.items():
    silhouette = silhouette_score(data_normalized, cluster_labels)
    davies_bouldin = davies_bouldin_score(data_normalized, cluster_labels)
    mutual_info = mutual_info_score(true_labels, cluster_labels)
    print(f"Metric: {metric}, Linkage: {linkage_method}")
    print(f"Silhouette Score: {silhouette:.4f}")
    print(f"Davies-Bouldin Index: {davies_bouldin:.4f}")

    print(f"Mutual Information: {mutual_info:.4f}") # Uncomment if true labels are available
    print("-" * 50)
```

OUTPUT:

```
Metric: euclidean, Linkage: ward
Silhouette Score: 0.3087
Davies-Bouldin Index: 1.1303
--------------------------------------------------
Metric: euclidean, Linkage: single
Silhouette Score: 0.0992
Davies-Bouldin Index: 0.5491
--------------------------------------------------
Metric: euclidean, Linkage: complete
Silhouette Score: 0.3260
Davies-Bouldin Index: 1.0617
--------------------------------------------------
Metric: euclidean, Linkage: average
Silhouette Score: 0.3335
Davies-Bouldin Index: 0.9122
--------------------------------------------------
```

Result

Agglomerative Clustering Algorithm are executed and verified successfully.