**20CS2032L – MACHINE LEARNING TECHNIQUES**

**URK22AI1048**

|  |  |
| --- | --- |
| **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.

Ex. 06| Agglomerative Clustering Algorithm

1

**20CS2032L – Machine Learning Techniques**

**URK22AI1048**

# Source Code

### 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())

### 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])

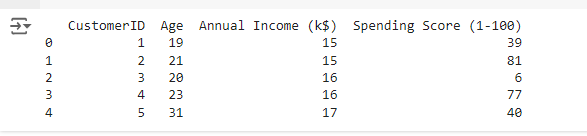
Ex. 06| Agglomerative Clustering Algorithm

2

**20CS2032L – Machine Learning Techniques**

**URK22AI1048**

## OUTPUT:



### 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)

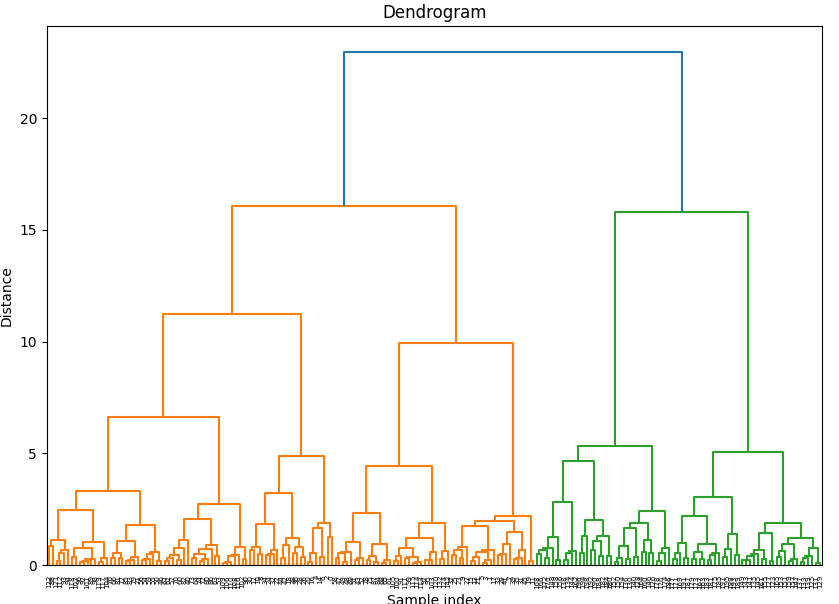
## OUTPUT:

Ex. 06| Agglomerative Clustering Algorithm

3

**20CS2032L – Machine Learning Techniques**

**URK22AI1048**



### 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:

Ex. 06| Agglomerative Clustering Algorithm

4

**20CS2032L – Machine Learning Techniques**

**URK22AI1048**

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:



### 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():

Ex. 06| Agglomerative Clustering Algorithm

5

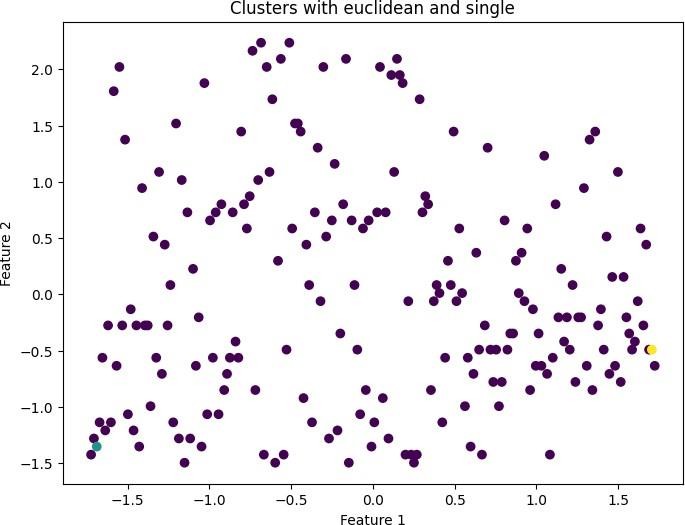
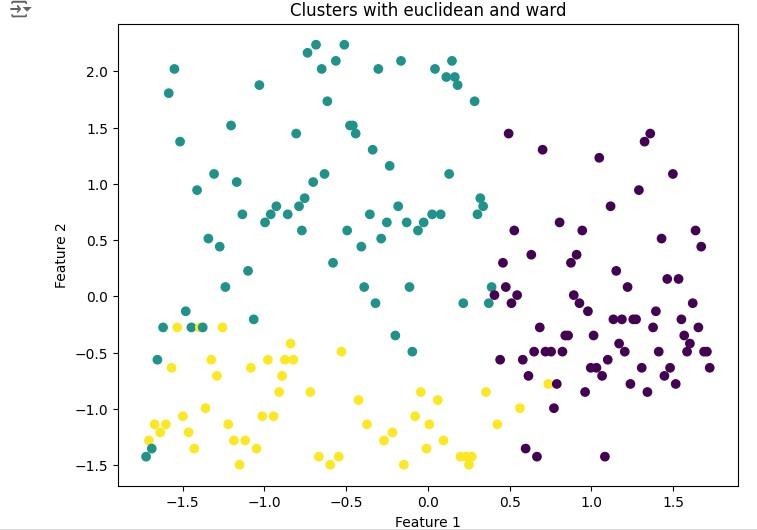
**20CS2032L – Machine Learning Techniques**

**URK22AI1048**

plot\_scatter(data\_normalized[:, :2], cluster\_labels, f'Clusters with {metric} and

{linkage\_method}')

## OUTPUT:



Ex. 06| Agglomerative Clustering Algorithm

6

**20CS2032L – Machine Learning Techniques**

**URK22AI1048**

### Analyse the results using scatter plot

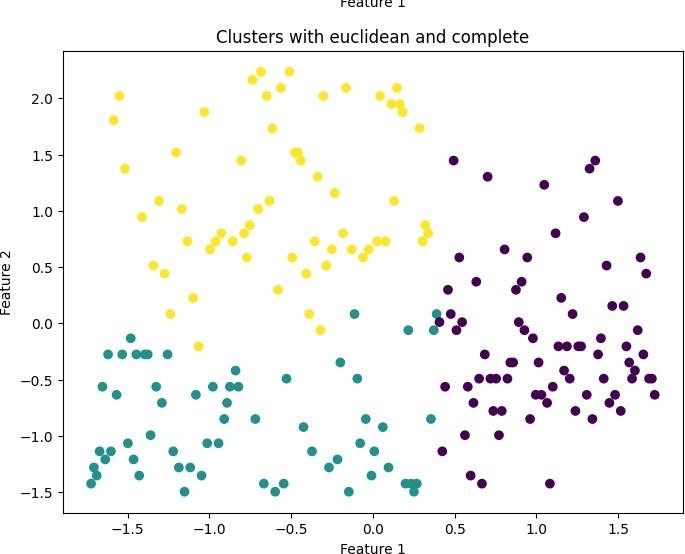
def plot\_scatter(data, cluster\_labels, title): plt.figure(figsize=(8, 6))

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:



Ex. 06| Agglomerative Clustering Algorithm

7

**20CS2032L – Machine Learning Techniques**

**URK22AI1048**

### 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']

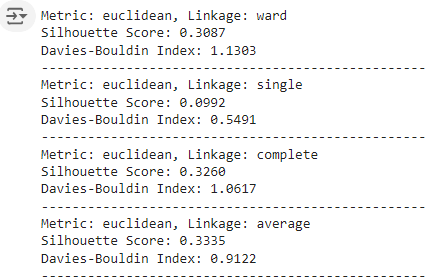
for (metric, linkage\_method), cluster\_labels in results.items(): silhouette = silhouette\_score(data\_normalized, cluster\_labels)

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:**



# Result

Agglomerative Clustering Algorithm are executed and verified successfully.

Ex. 06| Agglomerative Clustering Algorithm

8