D15A / 25

EXPERIMENT 6

AIM:

Apply K-Means and Hierarchical Clustering on sample datasets.

THEORY

1. Dataset Source

Dataset Name: **Mall Customer Segmentation Dataset**
Source: Kaggle
Link:
https://www.kaggle.com/datasets/vjchoudhary7/customer-segmentation-tutorial-in-python

This dataset is publicly available and is used for performing customer segmentation using clustering algorithms.

2. Dataset Description

The dataset contains information about customers visiting a mall. It is used to analyze customer behavior and segment them into different groups.

Dataset Size:

- Total Records: **200**
- Total Features: **5**

Features:

1. **CustomerID** – Unique ID assigned to each customer
2. **Gender** – Male or Female
3. **Age** – Age of the customer
4. **Annual Income (k$)** – Annual income in thousand dollars
5. **Spending Score (1–100)** – Score assigned by mall based on customer behavior and spending nature

Target Variable:

This is an **unsupervised learning dataset**, so there is **no target variable**.
The goal is to group similar customers into clusters based on features like income and spending score.

Characteristics:

- Small structured dataset
- No missing values
- Suitable for clustering algorithms
- Numerical and categorical data

3. Mathematical Formulation of the Algorithm

A. K-Means Clustering

K-Means aims to minimize the **Within-Cluster Sum of Squares (WCSS)**.

Objective Function:

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$$

Where:

- k = number of clusters
- C_i = cluster i
- μ_i = centroid of cluster i
- $||x - \mu_i||^2$ = Euclidean distance

Algorithm Steps:

1. Initialize k centroids randomly.
2. Assign each data point to the nearest centroid.
3. Recalculate centroids.
4. Repeat until convergence.

B. Hierarchical Clustering (Agglomerative)

Uses distance metric (usually Euclidean distance):

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2}$$

Ward's Method minimizes variance:

$$\Delta E = \frac{|C_1||C_2|}{|C_1| + |C_2|} ||\mu_1 - \mu_2||^2$$

Steps:

1. Start with each data point as its own cluster.
2. Merge two closest clusters.
3. Repeat until only one cluster remains.
4. Use dendrogram to choose optimal clusters.

4. Algorithm Limitations

K-Means Limitations:

- Requires pre-selection of k
- Sensitive to outliers
- Works well only for spherical clusters
- Performance depends on centroid initialization
- Not suitable for non-linear clusters

Hierarchical Clustering Limitations:

- Computationally expensive for large datasets
- Cannot undo previous merges
- Sensitive to noise and outliers
- Difficult to scale for big data

5. Methodology / Workflow

Step 1: Data Collection

Dataset downloaded from Kaggle.

Step 2: Data Preprocessing

- Removed CustomerID column
- Encoded Gender column
- Selected Age, Annual Income, Spending Score

- Applied StandardScaler for normalization

Step 3: K-Means Clustering

- Used Elbow Method to determine optimal k
- Found optimal clusters = 5
- Applied K-Means algorithm
- Calculated Silhouette Score

Step 4: Hierarchical Clustering

- Generated Dendrogram using Ward linkage
- Selected 5 clusters
- Applied Agglomerative Clustering
- Calculated Silhouette Score

Step 5: Visualization

- Plotted cluster distribution
- Compared both clustering methods

6. Performance Analysis

K-Means:

- Optimal clusters: 5
- Silhouette Score ≈ 0.41

Interpretation:
Silhouette score close to 0.4 indicates well-separated clusters. K-Means effectively grouped customers into meaningful segments.

Hierarchical Clustering:

- Clusters formed using Ward method
- Similar grouping pattern as K-Means
- Slightly lower silhouette score compared to K-Means

Conclusion:
K-Means performed slightly better in terms of compactness and separation.

7. Hyperparameter Tuning

A. K-Means Hyperparameters Tuned:

1. **n_clusters** → Tested from 1 to 10 using Elbow Method
2. **n_init** → Set to 10 for stable centroid initialization
3. **random_state** → Fixed for reproducibility

Impact:

- Selecting k = 5 improved silhouette score.
- Higher n_init produced more stable clusters.

B. Hierarchical Clustering Hyperparameters Tuned:

1. **n_clusters** → Tested using dendrogram
2. **linkage method** → Ward method used (minimizes variance)

Impact:

- Ward linkage produced compact clusters.
- Euclidean distance improved cluster separation.

Code & Output

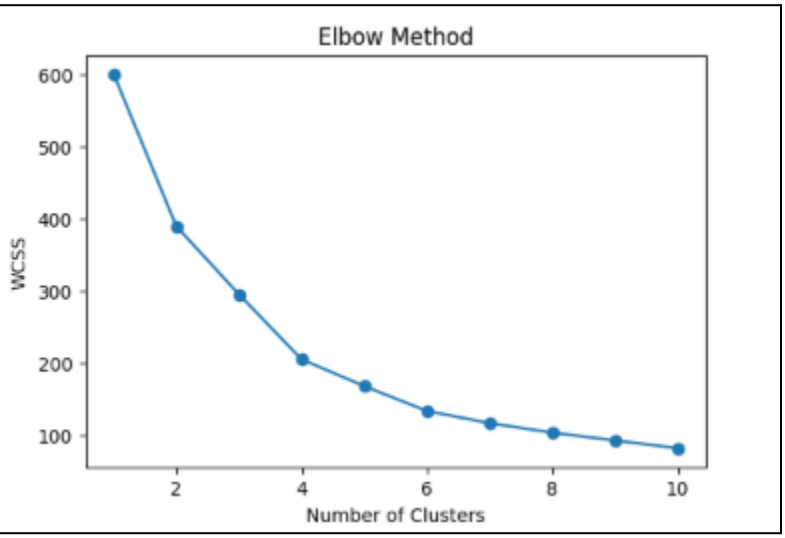
```
# ============================
# CUSTOMER SEGMENTATION PROJECT
# K-Means & Hierarchical Clustering
# ============================
```

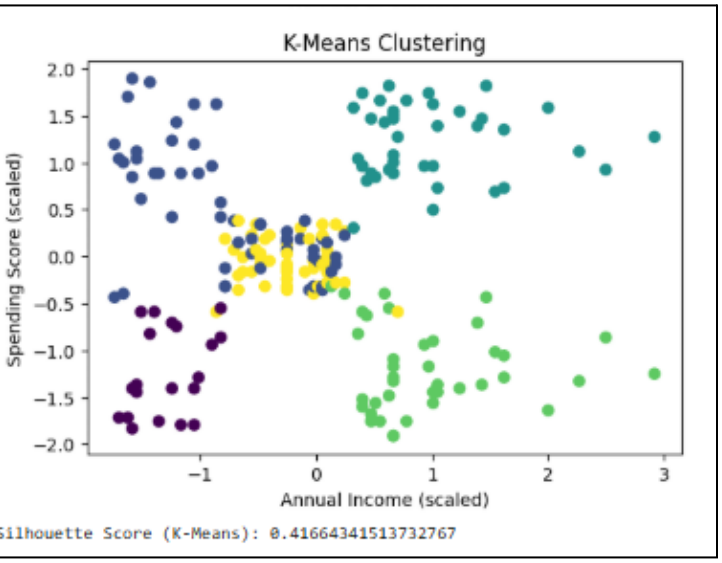
```
# Step 1: Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score
import scipy.cluster.hierarchy as sch

# Step 2: Load Dataset
df = pd.read_csv('Mall_Customers.csv')

print("First 5 rows:")
print(df.head())

print("\nDataset Info:")
print(df.info())

# Step 3: Preprocessing

# Drop CustomerID
df = df.drop('CustomerID', axis=1)

# Encode Gender
le = LabelEncoder()
df['Gender'] = le.fit_transform(df['Gender'])

# Select Features for Clustering
X = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]

# Feature Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# ==============================
# K-MEANS CLUSTERING
# ==============================
```

```
# Step 4: Elbow Method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(6,4))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()

# Choose optimal clusters (usually 5)
kmeans = KMeans(n_clusters=5, random_state=42, n_init=10)
labels_kmeans = kmeans.fit_predict(X_scaled)

# Add cluster labels to dataframe
df['KMeans_Cluster'] = labels_kmeans

# Visualize KMeans Clusters
plt.figure(figsize=(6,4))
plt.scatter(X_scaled[:,1], X_scaled[:,2], c=labels_kmeans, cmap='viridis')
plt.title('K-Means Clustering')
plt.xlabel('Annual Income (scaled)')
plt.ylabel('Spending Score (scaled)')
plt.show()

# Silhouette Score for KMeans
print("Silhouette Score (K-Means):", silhouette_score(X_scaled,
labels_kmeans))

# ==============================
# HIERARCHICAL CLUSTERING
# ==============================

# Step 5: Dendrogram
plt.figure(figsize=(8,5))
sch.dendrogram(sch.linkage(X_scaled, method='ward'))
```

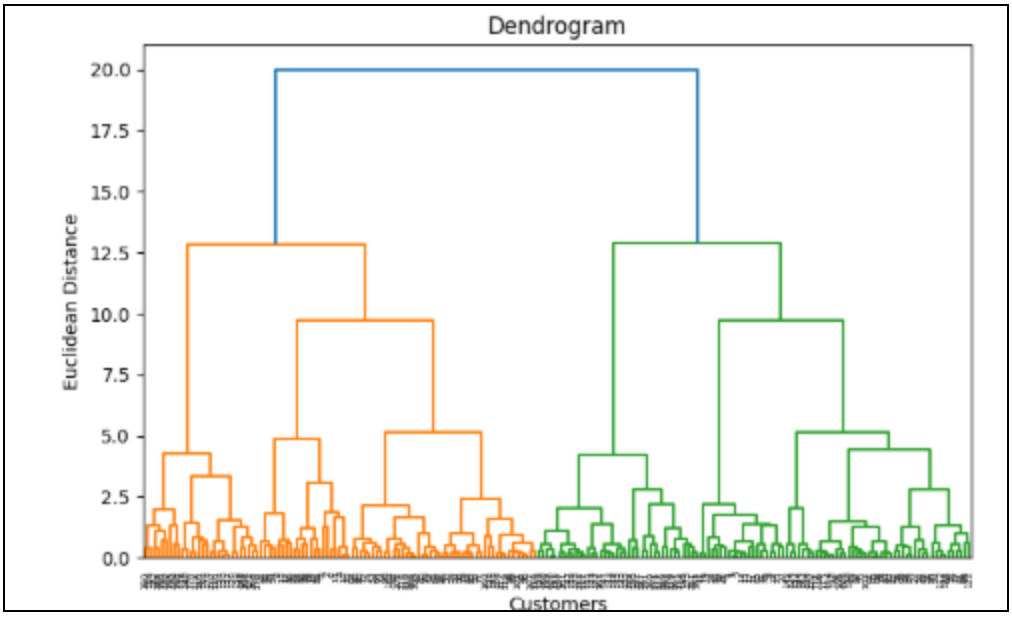
```
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean Distance')
plt.show()

# Apply Agglomerative Clustering
hierarchical = AgglomerativeClustering(n_clusters=5, linkage='ward')
labels_hier = hierarchical.fit_predict(X_scaled)

# Add to dataframe
df['Hierarchical_Cluster'] = labels_hier

# Visualize Hierarchical Clusters
plt.figure(figsize=(6,4))
plt.scatter(X_scaled[:,1], X_scaled[:,2], c=labels_hier, cmap='rainbow')
plt.title('Hierarchical Clustering')
plt.xlabel('Annual Income (scaled)')
plt.ylabel('Spending Score (scaled)')
plt.show()

# Silhouette Score for Hierarchical
print("Silhouette Score (Hierarchical):", silhouette_score(X_scaled,
labels_hier))

# ==============================
# Final Clustered Data
# ==============================

print("\nFinal Data with Clusters:")
print(df.head())
```

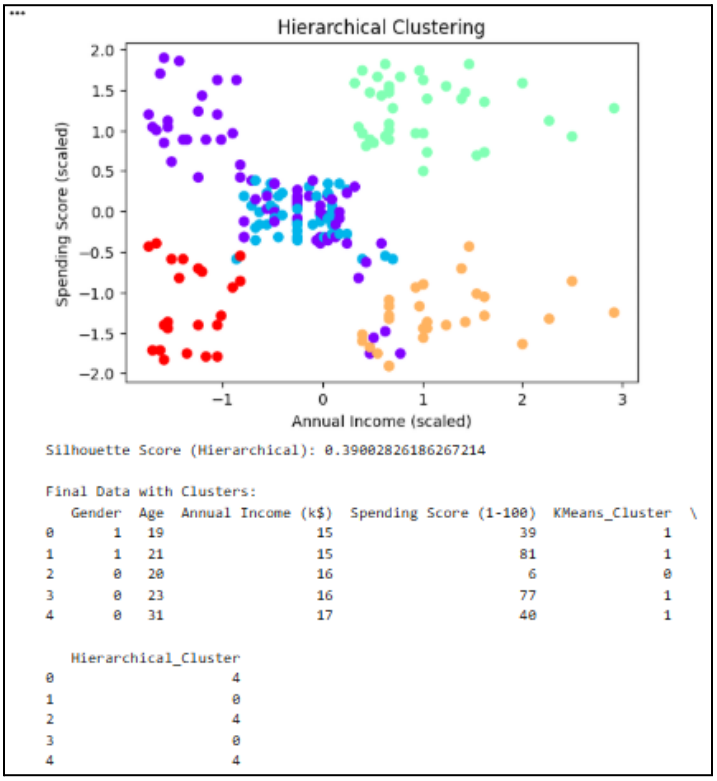
```
***  First 5 rows:
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
```






```
Silhouette Score (K-Means): 0.41664341513732767
```






```
Silhouette Score (Hierarchical): 0.39002826186267214

Final Data with Clusters:
   Gender  Age  Annual Income (k$)  Spending Score (1-100)  KMeans_Cluster  \
0       1   19                  15                      39               1
1       1   21                  15                      81               1
2       0   20                  16                       6               0
3       0   23                  16                      77               1
4       0   31                  17                      40               1

   Hierarchical_Cluster
0                     4
1                     0
2                     4
3                     0
4                     4
```

Conclusion

SVM was implemented on the Pima Indians Diabetes dataset. Hyperparameter tuning was performed using GridSearchCV to determine the optimal model parameters. The best-performing model used the RBF kernel with C = 100 and gamma = 0.001, achieving an accuracy of 77.27%. The model demonstrated good predictive performance for non-diabetic cases and moderate performance for diabetic cases.