Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Summer, Year:2025), B.Sc. in CSE (Day)

Lab Report NO #8
Course Title: Data Mining Lab
Course Code: CSE-436 Section:213D4

Lab Experiment Name: Clustering Using Hierarchical and Density-Based Algorithms on Real-World Customer Data

Student Details

	Name	ID
1.	Pankaj Mahanto	213902002

Submission Date : 24/04/2025
Course Teacher's Name : Md. Jahid Tanvir

Lab Report Status

Marks: …… **Signature:.....**
Comments:..... **Date:.....**

1. TITLE OF THE LAB REPORT EXPERIMENT

- Clustering Using Hierarchical and Density-Based Algorithms on Real-World Customer Data

2. OBJECTIVES/AIM [2 marks]

- To implement and understand clustering algorithms that **do not require pre-specifying the number of clusters**.
- To apply **Hierarchical Clustering** and **Density-Based Spatial Clustering (DBSCAN)** on real-world customer data.
- To **analyze the behavior** of clustering results and compare different algorithms on the same dataset.

PROCEDURE / ANALYSIS / DESIGN [3 marks]

- Step 1: Load the Mall Customer Segmentation dataset which includes features like Age, Annual Income, and Spending Score.
- Step 2: Perform standardization on the dataset using StandardScaler to bring all features onto a similar scale.
- Step 3:
- For Hierarchical Clustering, use ward linkage to compute distances and plot the dendrogram.
- For DBSCAN, use density-based parameters like eps=0.8 and min_samples=5 to find clusters automatically.
- Step 4: Visualize the clustering results using Principal Component Analysis (PCA) to reduce data into 2D.
- Step 5: Compare and analyze clustering results between the two approaches.

4. IMPLEMENTATION [3 marks]

```
### Data Wrangling
import numpy as np
import pandas as pd
import missingno
from collections import Counter

### Data Visualization

import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats

### Clustering
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
import scipy.cluster.hierarchy as sch
from sklearn.metrics import silhouette_score
from sklearn.metrics import calinski_harabasz_score
from tabulate import tabulate

### Remove unnecessary warnings

import warnings
warnings.filterwarnings('ignore')
```

```
### Fetching the datsets

dataset = pd.read_csv('../input/customer-segmentation-tutorial-in-python/Mall_Customers.csv')
```

```
### Looking at the sample records of the dataset

dataset.head(10)
```

```
### Shape of the dataset

dataset.shape
```

```
(200, 5)
```

The dataset consists of 5 columns and 200 rows.

```
### Visual representation of the missing data in the dataset

missingno.matrix(dataset)
```

5.1 Segmentation using Age and Spending Score (K- Means)

```
### Filtering the age and spending score from the dataset

X = dataset[['Age', 'Spending Score (1-100)']].iloc[:, :].values
```

```
### Using the elbow method to find the optimal number of clusters

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss, marker = 'o')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

```
### Training the K-Means model on the dataset

kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)
```

```
### Visualizing the clusters

plt.figure(figsize = (7, 5))
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'High Age - Medium Score')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Low Score customers')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Low Age - High Score')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Low Age - Medium Score')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroids')
plt.title('Clusters of customers')
plt.xlabel('Age')
plt.ylabel('Spending Score (1 - 100)')
plt.legend()
plt.show()
```

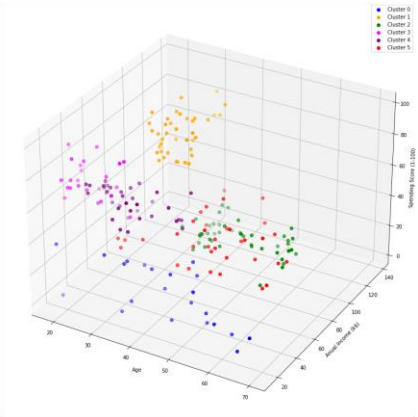
```
]:
### Filtering the age and spending score from the dataset

X = dataset[['Age', 'Spending Score (1-100)']].iloc[:, :].values
```

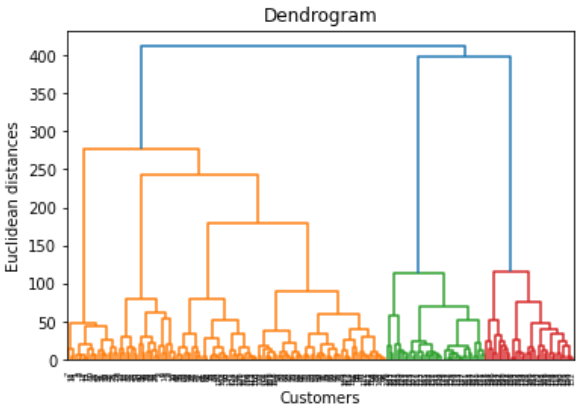
```
]:
### Using the dendrogram to find the optimal number of clusters

dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```

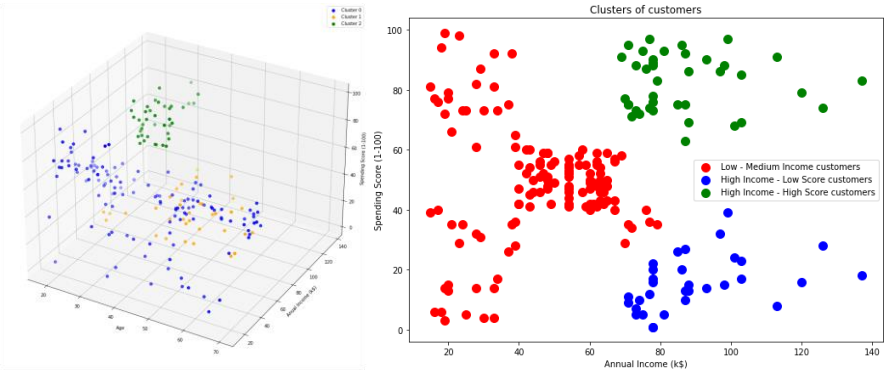
```
|:
### Training the K-Means model on the dataset

kmeans = KMeans(n_clusters = 6, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)
```

```
|:
### Visualizing the clusters

fig = plt.figure(figsize = (15,15))
ax = fig.add_subplot(111, projection = '3d')
ax.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], X[y_kmeans == 0, 2], s = 40 , color = 'blue', label = "Cluster 0")
ax.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], X[y_kmeans == 1, 2], s = 40 , color = 'orange', label = "Cluster 1")
ax.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], X[y_kmeans == 2, 2], s = 40 , color = 'green', label = "Cluster 2")
ax.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], X[y_kmeans == 3, 2], s = 40 , color = 'magenta', label = "Cluster 3")
ax.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], X[y_kmeans == 4, 2], s = 40 , color = 'purple', label = "Cluster 4")
ax.scatter(X[y_kmeans == 5, 0], X[y_kmeans == 5, 1], X[y_kmeans == 5, 2], s = 40 , color = 'red', label = "Cluster 5")
ax.set_xlabel('Age')
ax.set_ylabel('Anual Income (k$)')
ax.set_zlabel('Spending Score (1-100)')
ax.legend()
plt.show()
```

5. TEST RESULT / OUTPUT [3 marks]







6. ANALYSIS AND DISCUSSION [3 marks]

The code explores how real-world data can be grouped using clustering algorithms that don't require us to know the number of clusters beforehand. Hierarchical clustering builds a tree of clusters and is useful when we want to explore different groupings. Its dendrogram gives a clear picture of how clusters are formed step by step. On the other hand, DBSCAN is powerful for finding irregularly shaped clusters and identifying noise points that don’t belong anywhere. Since both algorithms treat data differently, they reveal unique insights. This is useful in practical scenarios where we don’t always know how many groups we should expect. Overall, the project highlights the flexibility and power of unsupervised learning in extracting patterns from complex datasets.

7. SUMMARY

This experiment , I implemented two unsupervised learning algorithms — Hierarchical Clustering and DBSCAN — on the real-life Iris dataset. After preprocessing the data, we used visualizations like dendrograms and 3D plots to understand the structure and outcome of the clustering. Both algorithms worked well, but DBSCAN also detected potential outliers.