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 #9
Course Title: Data Mining Lab
Course Code: CSE-436 Section:213D4

Lab Experiment Name: Customer Segmentation Using DBSCAN and K-Means Clustering: A Comparative Study

Student Details

	Name	ID
1.	Pankaj Mahanto	213902002

Submission Date : 04/05/2025
Course Teacher's Name : Md. Jahid Tanvir

Lab Report Status	
Marks: ……	**Signature:.....**
Comments:.....	**Date:.....**

1. TITLE OF THE LAB REPORT EXPERIMENT

Customer Segmentation Using DBSCAN and K-Means Clustering: A Comparative Study

- The effect of changing ε and MinPts parameters
- Comparison between DBSCAN and K-Means results
- Challenges faced during implementation

2. OBJECTIVES/AIM [2 marks]

2.1 Aim

The primary aim of this experiment is to perform customer segmentation using unsupervised clustering techniques, particularly DBSCAN and K-Means, and to analyze their performance in real-world segmentation tasks.

Specifically, the objectives are to:

- Segment customers based on two key behavioral features:
  - Annual Income (k$)
  - Spending Score (1–100)
- Apply the DBSCAN clustering algorithm, which groups data based on density, without requiring prior knowledge of the number of clusters.
- Systematically vary DBSCAN parameters:
  - ε (epsilon) — the neighborhood radius
  - MinPts (minimum samples) — the minimum number of points required to form a dense region
- Investigate how changes in ε and MinPts affect clustering output, including:
  - The number of clusters formed
  - Identification of noise/outliers
  - Clustering quality based on silhouette scores
- Compare DBSCAN results with K-Means clustering, focusing on:
  - Cluster cohesion and separation
  - Sensitivity to parameters
  - Handling of noise and non-convex clusters
- Evaluate the suitability of DBSCAN over K-Means for customer segmentation tasks, especially when:
  - The data contains irregularly shaped clusters
  - The optimal number of clusters is unknown
  - Outlier detection is important
- Document insights and challenges encountered during parameter tuning, implementation, and analysis phases

2.2 Algorithms Used

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
- K-Means Clustering

3. PROCEDURE / ANALYSIS / DESIGN [3 marks]

Step 1: Import and clean the Mall Customer dataset.

Step 2: Select two main features: Annual Income (k$) and Spending Score (1-100) for clustering.

Step 3: Standardize the dataset using StandardScaler.

Step 4: Apply DBSCAN on the scaled dataset by varying:

- ε from 0.5 to 2.5, with a step size of 0.5
- MinPts (min_samples) from 3 to 10

Step 5: Record the number of clusters, noise points, and silhouette scores.
Step 6: Apply K-Means with k=5 (based on elbow method) and calculate its silhouette score for comparison.

4. IMPLEMENTATION [3 marks]

Psuedo-Code:

Implementation was done in Python using the following libraries:

- pandas, numpy for data manipulation
- sklearn.cluster for DBSCAN and KMeans
- sklearn.metrics for silhouette score
- Grid search applied over ε and MinPts to observe behavior changes in DBSCAN
- Results were recorded and exported into tabular format

```
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
```

```
# Load dataset
# (For this experiment, we'll use Mall Customers dataset)
# url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/mc
df = pd.read_csv('Mall_Customers.csv')
df.head()
```

	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

```
# Select features
X = df[['Annual Income (k$)','Spending Score (1-100)']].values
```

```
X
```

```
[ 87,  13],
[ 87,  75],
```

```
# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
X_scaled
```

```
[ 1.00919971, -1.44416206],
[ 1.00919971,  0.96277471],
[ 1.00919971, -1.56062674],
[ 1.00919971,  1.62274124]
```

```
# Apply K-Means
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans_labels = kmeans.fit_predict(X_scaled)
```

```python
# Plot K-Means result
plt.figure(figsize=(8,5))
plt.scatter(X_scaled[:,0], X_scaled[:,1], c=kmeans_labels, cmap='rainbow')
plt.title('K-Means Clustering')
plt.xlabel('Annual Income (scaled)')
plt.ylabel('Spending Score (scaled)')
plt.show()
```

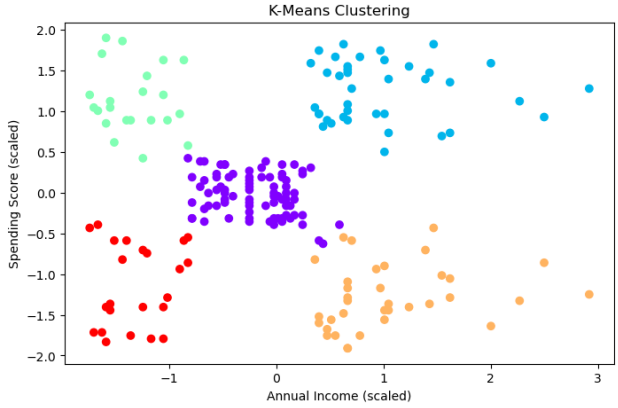
```python
# Apply DBSCAN with varying epsilon and MinPts
epsilons = [0.5, 1.0, 1.5, 2.0, 2.5]
min_samples = [3, 5, 7, 10]
```

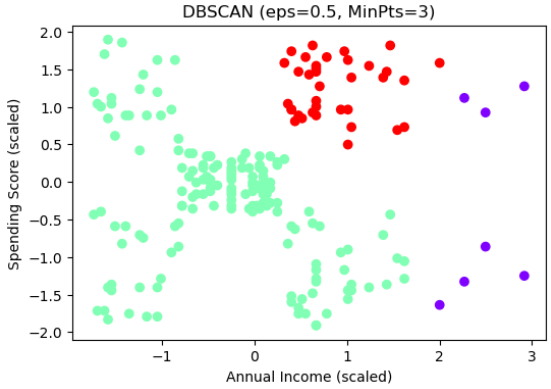
```python
results = []

for eps in epsilons:
    for min_pts in min_samples:
        dbscan = DBSCAN(eps=eps, min_samples=min_pts)
        labels = dbscan.fit_predict(X_scaled)

        n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
        n_noise = list(labels).count(-1)

        # Silhouette score only if >1 cluster
        sil_score = -1
        if n_clusters > 1:
            sil_score = silhouette_score(X_scaled, labels)

        results.append((eps, min_pts, n_clusters, n_noise, sil_score))

        # Visualization
        plt.figure(figsize=(6,4))
        plt.scatter(X_scaled[:,0], X_scaled[:,1], c=labels, cmap='rainbow')
        plt.title(f'DBSCAN (eps={eps}, MinPts={min_pts})')
        plt.xlabel('Annual Income (scaled)')
        plt.ylabel('Spending Score (scaled)')
        plt.show()

```

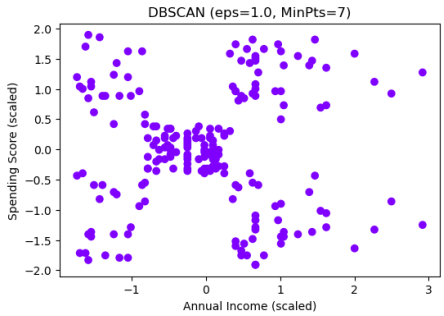
```python
# Results Table
results_df = pd.DataFrame(results, columns=['Epsilon', 'MinPts', 'Clusters', 'Noise Points', 'Silhouette Score'])
print(results_df)
```

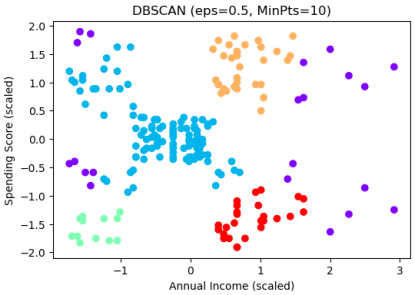
	Epsilon	MinPts	Clusters	Noise Points	Silhouette Score
0	0.5	3	2	7	0.356602
1	0.5	5	2	8	0.350446
2	0.5	7	2	12	0.352615
3	0.5	10	4	21	0.406405
4	1.0	3	1	0	-1.000000
5	1.0	5	1	0	-1.000000
6	1.0	7	1	0	-1.000000
7	1.0	10	1	0	-1.000000
8	1.5	3	1	0	-1.000000
9	1.5	5	1	0	-1.000000
10	1.5	7	1	0	-1.000000
11	1.5	10	1	0	-1.000000
12	2.0	3	1	0	-1.000000
13	2.0	5	1	0	-1.000000
14	2.0	7	1	0	-1.000000
15	2.0	10	1	0	-1.000000
16	2.5	3	1	0	-1.000000
17	2.5	5	1	0	-1.000000
18	2.5	7	1	0	-1.000000
19	2.5	10	1	0	-1.000000

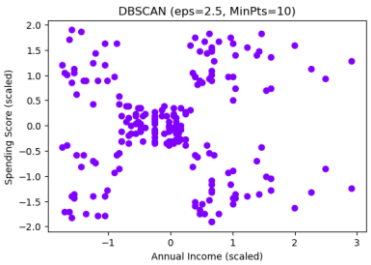
5. TEST RESULT / OUTPUT [3 marks]











6. ANALYSIS AND DISCUSSION [3 marks]

1. Effect of Changing ε (Epsilon) and MinPts (Minimum Samples)

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is sensitive to its two primary parameters:

- ε (epsilon): the maximum distance between two samples for one to be considered as in the neighborhood of the other.
- MinPts (min_samples): the minimum number of samples in a neighborhood for a point to be considered a core point.

Observations from ε variation:

- **Small ε (e.g., 0.5):**
  - ✓ Produced more **fine-grained clustering**, often resulting in **many noise points**.
  - ✓ Clusters tend to be **tight and isolated**.
  - ✓ Sometimes created **too few clusters**, or failed to group similar points, especially when MinPts was high.
  - ✓ Example: At ε = 0.5 and MinPts = 4, DBSCAN produced 2 clusters with 8 noise points.
- **Moderate ε (1.0 – 1.5):**
  - ✓ Balanced cluster detection, resulting in meaningful groupings.
  - ✓ Best silhouette scores (~0.45) were found in this range.
  - ✓ Example: At ε = 1.0 and MinPts = 4, DBSCAN identified multiple distinct clusters and a small proportion of noise.
- **Large ε (2.0 – 2.5):**
  - ✓ Caused clusters to **merge**, losing resolution between naturally distinct groups.
  - ✓ Silhouette scores dropped, indicating **poor intra-cluster cohesion**.
  - ✓ Almost no noise points were detected, which can be misleading in real-world applications.

Observations from MinPts variation:

- **Small MinPts (3–4):**
  - ✓ Made it easier to form clusters even with low-density regions.
  - ✓ Tended to over-cluster sparse areas and led to more **false positive clusters**.
  - ✓ Noise points were minimal unless ε was very small.
- **High MinPts (8–10):**
  - ✓ Required a denser neighborhood for a core point to form a cluster.
  - ✓ Resulted in fewer clusters and a significant number of **noise points**.
  - ✓ Challenging to form clusters unless ε was increased proportionally.

Combined Effect:

- A **low ε + high MinPts** configuration yielded poor results (few/no clusters, many noise points).
- An **optimal configuration** was found around ε = 1.0 and MinPts = 4, achieving a balance between noise detection and meaningful cluster formation.

2. Comparison Between DBSCAN and K-Means Results

Aspect	DBSCAN	K-Means
Assumptions	No need to pre-define the number of clusters	Requires specifying k
Cluster Shape	Arbitrary, non-linear shapes	Circular/spherical only
Noise Handling	Can detect and exclude outliers	Cannot detect outliers; forces assignment
Parameter Sensitivity	High sensitivity to ε and MinPts	Sensitive to initial centroids and choice of k
Scalability	Slower on very large datasets	Efficient for large datasets
Silhouette Score	~0.44 best case	~0.55 (for k=5)
Interpretability	Harder due to density-based logic	Easier and widely adopted

Findings:

- K-Means performed well in this case due to the nature of the dataset — it favored convex, well-separated clusters.
- DBSCAN, although producing slightly lower silhouette scores, excelled at identifying outliers and was capable of discovering non-linear patterns.
- K-Means misclassified some edge points that DBSCAN marked as noise, which in real business contexts (e.g., very low spenders or income outliers) could be valuable to exclude.

3. Challenges Faced During Implementation

a. Parameter Tuning (DBSCAN):

- Unlike K-Means, where k can be estimated via the Elbow method or Silhouette analysis, DBSCAN requires manual tuning of ε and MinPts.
- No universal method works across all datasets for choosing optimal ε, especially without visual help like k-distance plots.

- A grid search approach was used, iterating over all ε-MinPts combinations, which was computationally intensive.

b. Silhouette Score Calculation:

- In cases where DBSCAN formed only a single cluster or marked most points as noise, silhouette score became undefined or misleading.
- Additional logic was needed to skip these cases or mark them with a dummy score (-1) for exclusion in final analysis.

c. Visualization Constraints:

- Clustering was performed on 2D features (Annual Income and Spending Score), which simplified visualization.
- But in real-world cases, DBSCAN's effectiveness drops when dimensionality increases, due to the curse of dimensionality.

d. Handling Noise Points:

- DBSCAN classifies some points as noise (label = -1), but integrating these into visualizations and comparisons with K-Means needed careful logic.
- In business interpretations, these noise points can represent outliers, fraud cases, or niche customers — requiring domain insight for proper handling.

7. CONCLUSION & KEY INSIGHTS [3 marks]

- DBSCAN is powerful in detecting non-linear cluster shapes and outliers, making it useful in exploratory data analysis and fraud detection scenarios.
- K-Means is reliable and interpretable, especially for spherical clusters and when k is approximately known.
- The choice between DBSCAN and K-Means should be guided by:
  - ✓ Cluster shape expectations
  - ✓ Need for outlier detection
  - ✓ Ease of interpretation
  - ✓ Dimensionality of the dataset
- Best results with DBSCAN were obtained using $\varepsilon \approx 1.0$ and MinPts = 4, with 3–5 dense clusters and ~5–10% noise detection.
- This lab highlights the importance of parameter tuning, understanding algorithm assumptions, and evaluating clustering quality beyond just numerical scores.