□ Day 13 – DBSCAN Clustering (Unsupervised Learning)

📌 What is DBSCAN?

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups data points based on density. Unlike K-Means, it:

- Does **not require the number of clusters** as input.
- Detects **noise and outliers** effectively.

⚙️ **Key Parameters:**

- eps: Max distance between two points in a cluster.
- min_samples: Min points to form a dense region.
- **Point Types**:
    - Core Point
    - Border Point
    - Noise Point (label -1)

🗀 **Dataset Used**

Mall_Customers.csv
Features:

- Age
- Annual Income (k$)
- Spending Score (1–100)
- Gender (converted: Male = 0, Female = 1)

🛠️ **Implementation Steps**

✅ **1. Import Libraries**

```
import pandas as pd, numpy as np, seaborn as sns, matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from sklearn.cluster import DBSCAN

from sklearn.neighbors import NearestNeighbors

from sklearn.metrics import silhouette_score
```

✅ **2. Data Preprocessing**

```
df = pd.read_csv('/content/Mall_Customers (1).csv')

df.drop(columns=['CustomerID'], inplace=True)

df['Gender'] = df['Gender'].map({'Male': 0, 'Female': 1})
```

📊 **EDA**

- Scatter plot for Age vs Income
- Box plot for Age vs Gender

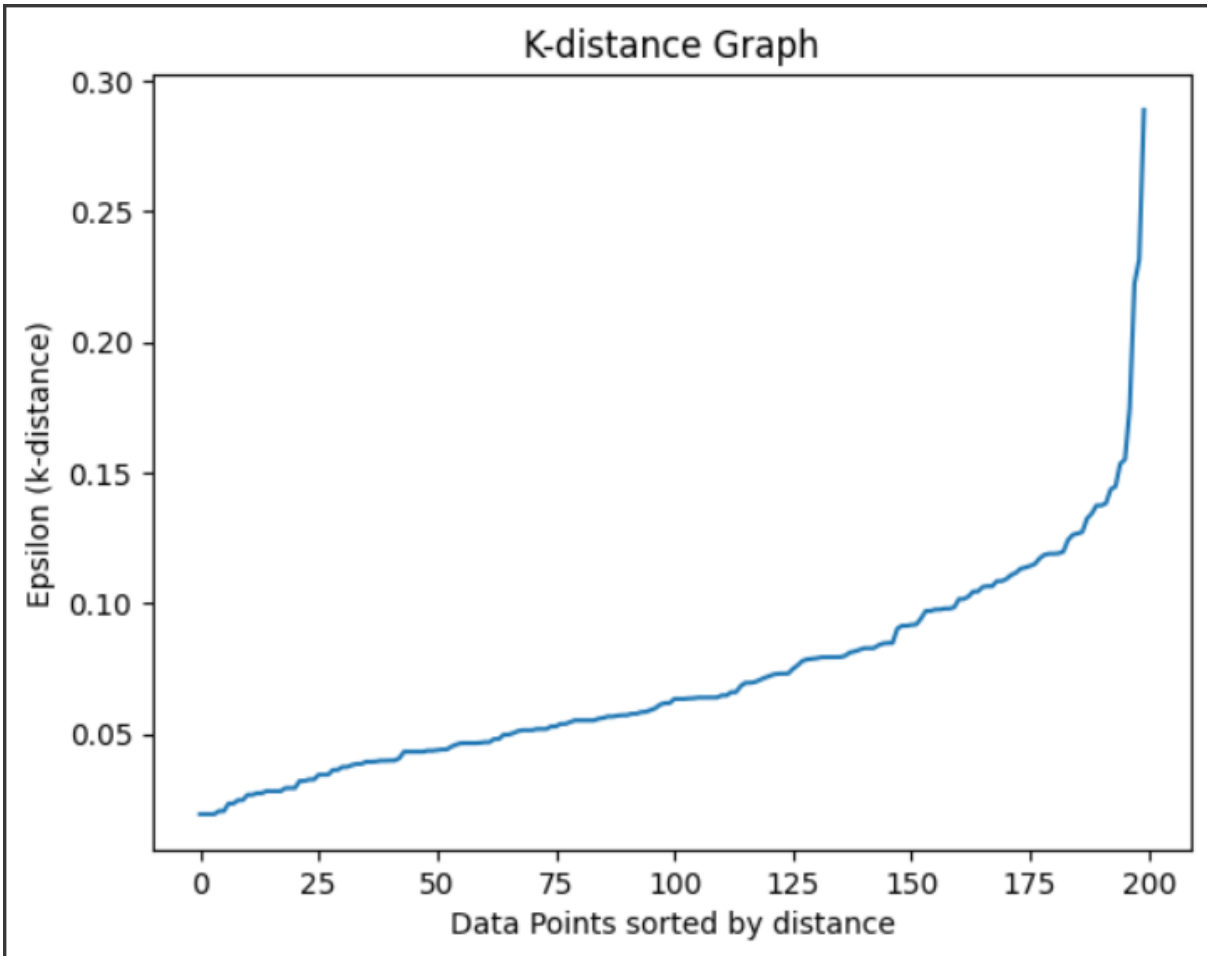
□ Scaling

```
scaler = MinMaxScaler()
df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']] = scaler.fit_transform(
   df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
)
```

🔍 Choosing eps (K-Distance plot)

```
scaled_features = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']


knn = NearestNeighbors(n_neighbors=6)
nbrs = knn.fit(df[scaled_features])
distances, _ = nbrs.kneighbors(df[scaled_features])
plt.plot(sorted(distances[:, 1]))
plt.xlabel("Data Points sorted by distance")
plt.ylabel("Epsilon (k-distance)")
plt.title("K-distance Graph")
plt.show()
```



DBSCAN Application

dbscan = DBSCAN(eps=0.13, min_samples=5, metric='euclidean')

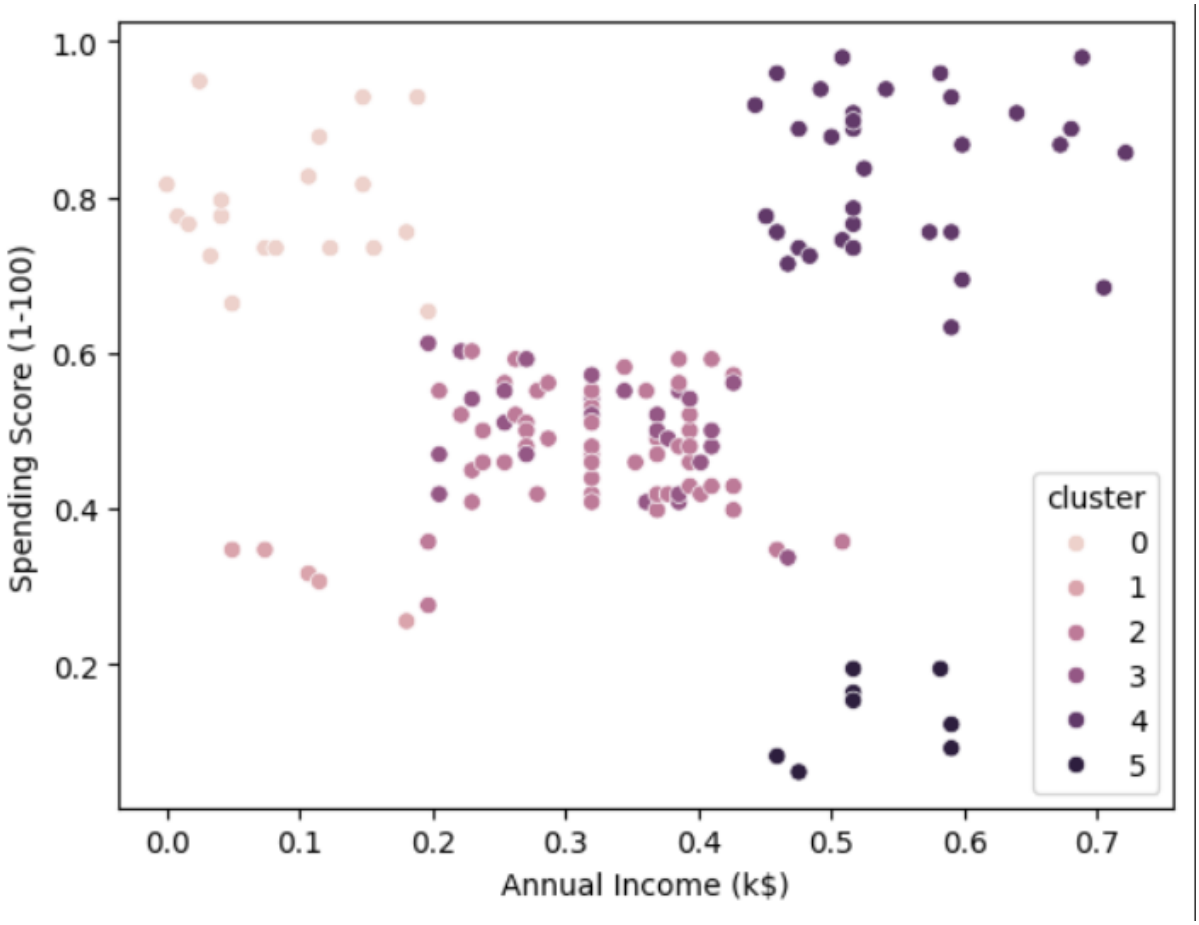
df['cluster'] = dbscan.fit_predict(df[scaled_features])

- Cluster labels assigned
- Label -1 means noise

Cluster Visualization

df_filtered = df[df['cluster'] != -1]

sns.scatterplot(data=df_filtered, x='Annual Income (k$)', y='Spending Score (1-100)', hue='cluster')



☑ Silhouette Score

score = silhouette_score(df_filtered[scaled_features], df_filtered['cluster'])

print("Silhouette Score:", score)

output : Silhouette Score: 0.45887319649308095

📝 Conclusion

- DBSCAN clustered dense regions and ignored noise.
- No need to predefine cluster count.
- Useful for detecting **irregular shapes** and **outliers**.
- A strong alternative to K-Means and Hierarchical Clustering for real-world data.
