DBSCAN

A Density-Based Clustering Algorithm

What is DBSCAN?

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise

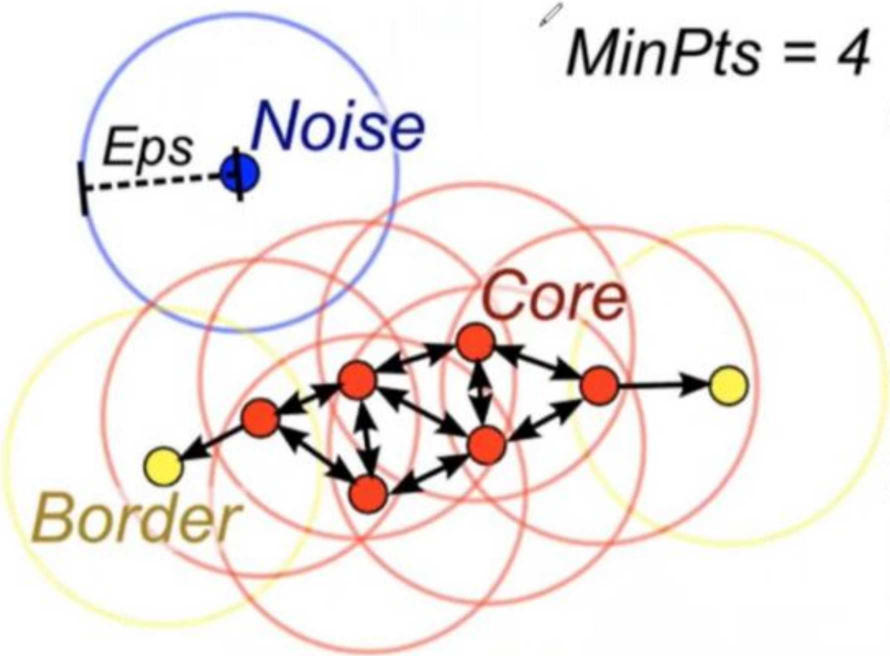
It is an unsupervised clustering algorithm that groups points based on density.

Key Idea:

- Points in high-density regions are grouped together.
- Low-density points are classified as noise or outliers.

Key Terms:

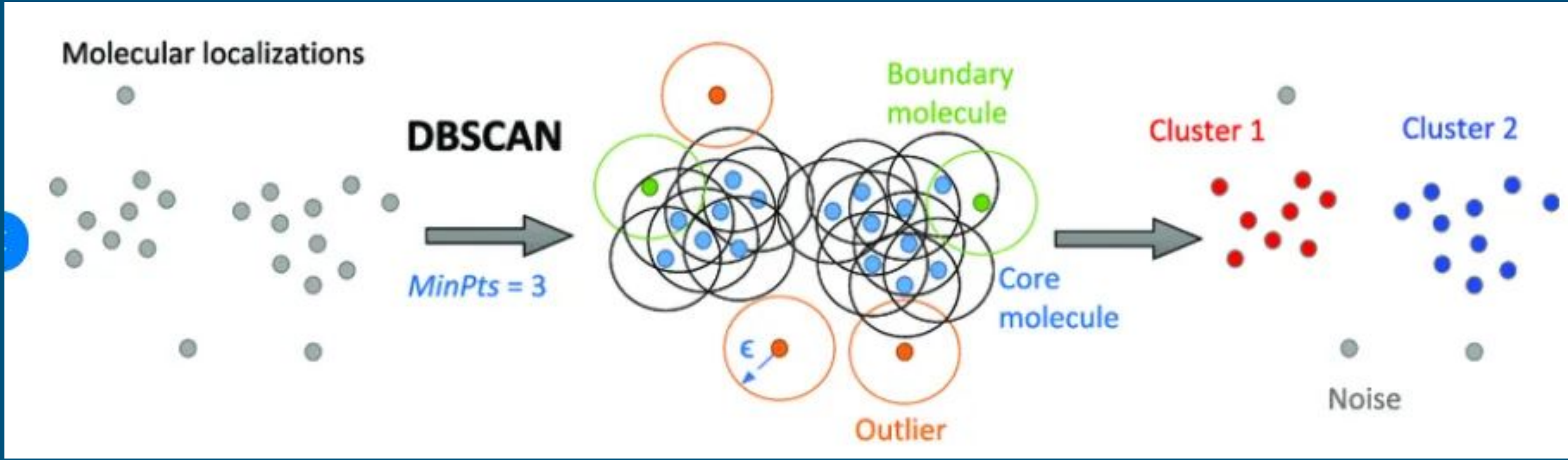
- Core points: Have at least MinPts neighbors within distance ε (epsilon).
- Border points: Have fewer than MinPts but are within ε of a core point.
- Noise points: Are neither core nor border points.



Red: Core Points

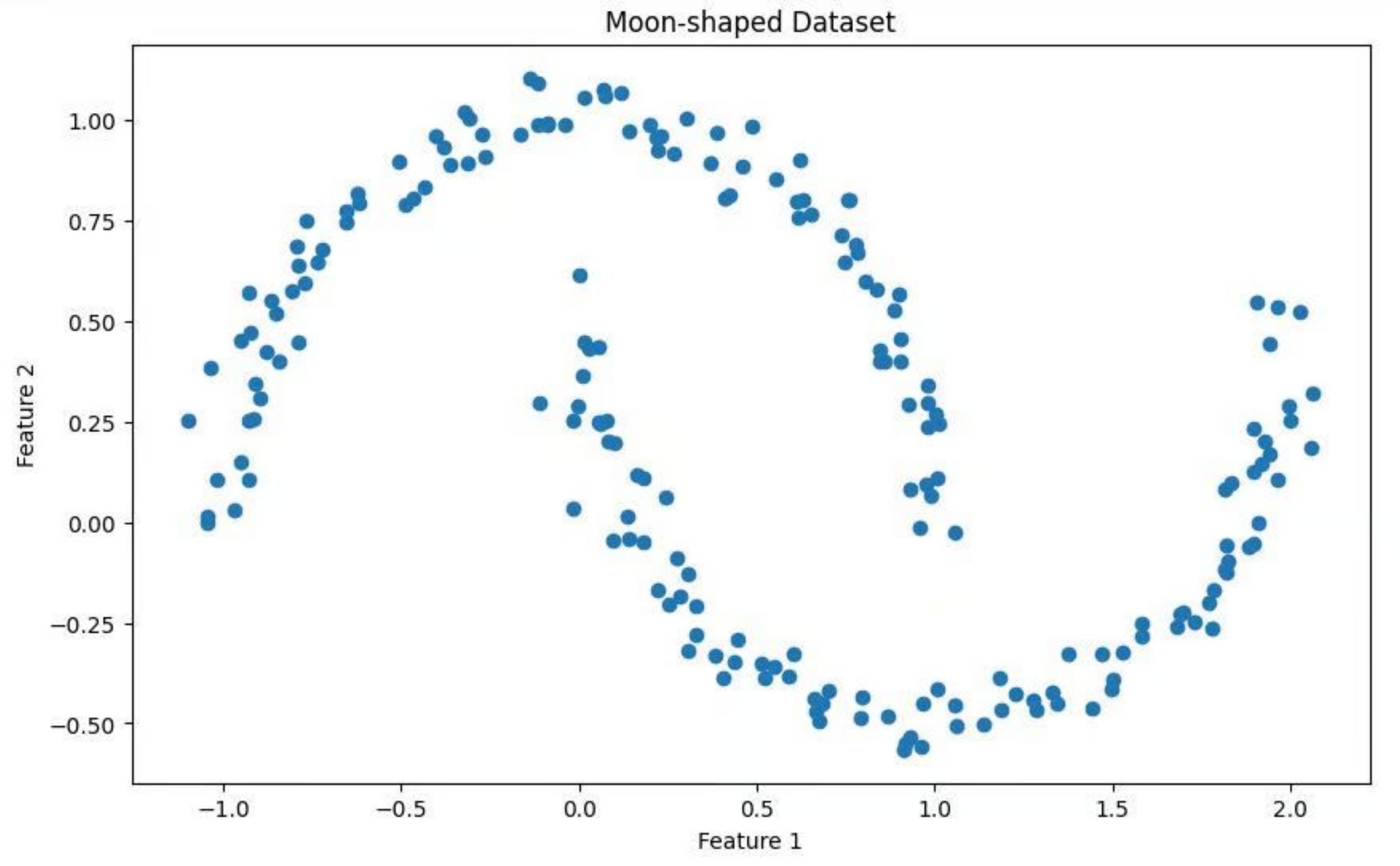
Yellow: Border points. Still part of the cluster because it's within epsilon of a core point, but not does not meet the min_points criteria

Blue: Noise point. Not assigned to a cluster

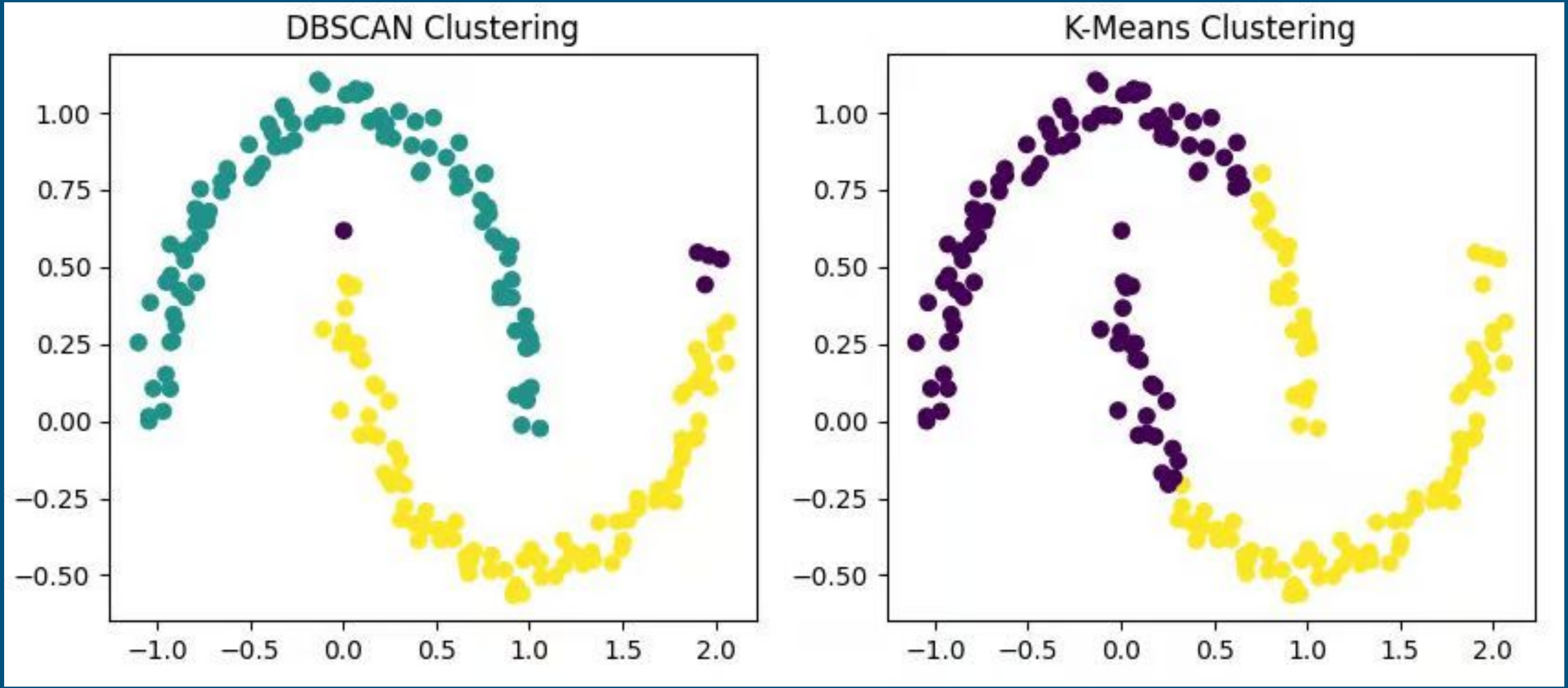

Molecular localizations
DBSCAN
MinPts = 3
ϵ
Boundary molecule
Core molecule
Outlier
Cluster 1
Cluster 2
Noise

DBSCAN vs. K-Means: Key Differences

Feature	DBScan	K-Means
Approach	Density-based	Centroid-based
Shape of Clusters	Arbitrary (can be non-spherical)	Circular
Handles Noise?	Yes, can classify outliers	No, assigns every point to a cluster
Requires k (Number Of Clusters?	No	Yes



Moon-shaped Dataset
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
Feature 2
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Feature 1


DBSCAN Clustering
K-Means Clustering
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

PROBLEM STATEMENT

Given a dataset, Mall_Customers.csv, which contains information about customers of a mall.

We have to perform clustering using the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm.

1. Importing Necessary Libraries

```
[1]: import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn.cluster import DBSCAN
```

numpy: Used for numerical operations.

pandas: Used for handling tabular data.

seaborn & matplotlib.pyplot: Used for visualization.

DBSCAN from sklearn.cluster: Used for clustering.

2. Loading the Dataset

```
[2]: df = pd.read_csv('Mall_Customers.csv')
     X_train = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
```

Reads the ‘Mall_Customers’ CSV file containing customer data.

Extracts three features for clustering:

- Age
- Annual Income (k$)
- Spending Score (1-100)

3. Applying DBSCAN Clustering & Storing the Clusters

```
[3]: clustering = DBSCAN(eps=12.5, min_samples=4).fit(X_train)
     DBSCAN_dataset = X_train.copy()
     DBSCAN_dataset.loc[:,'Cluster'] = clustering.labels_
```

eps=12.5: Defines the maximum distance for two points to be considered neighbors.

min_samples=4: A cluster must have at least 4 points.

fit(X_train): Performs DBSCAN clustering on the data.

- Creates a copy of X_train for visualization.
- clustering.labels_ assigns a cluster label to each data point.
  - -1 indicates an outlier (noise point).
  - Other numbers represent different clusters.

4. Analyzing Clusters

```
[4]: DBSCAN_dataset.Cluster.value_counts().to_frame()
```

[4]:

Cluster	count
0	112
2	34
3	24
-1	18
1	8
4	4

- Displays the count of points in each cluster.

5. Analyzing Outliers

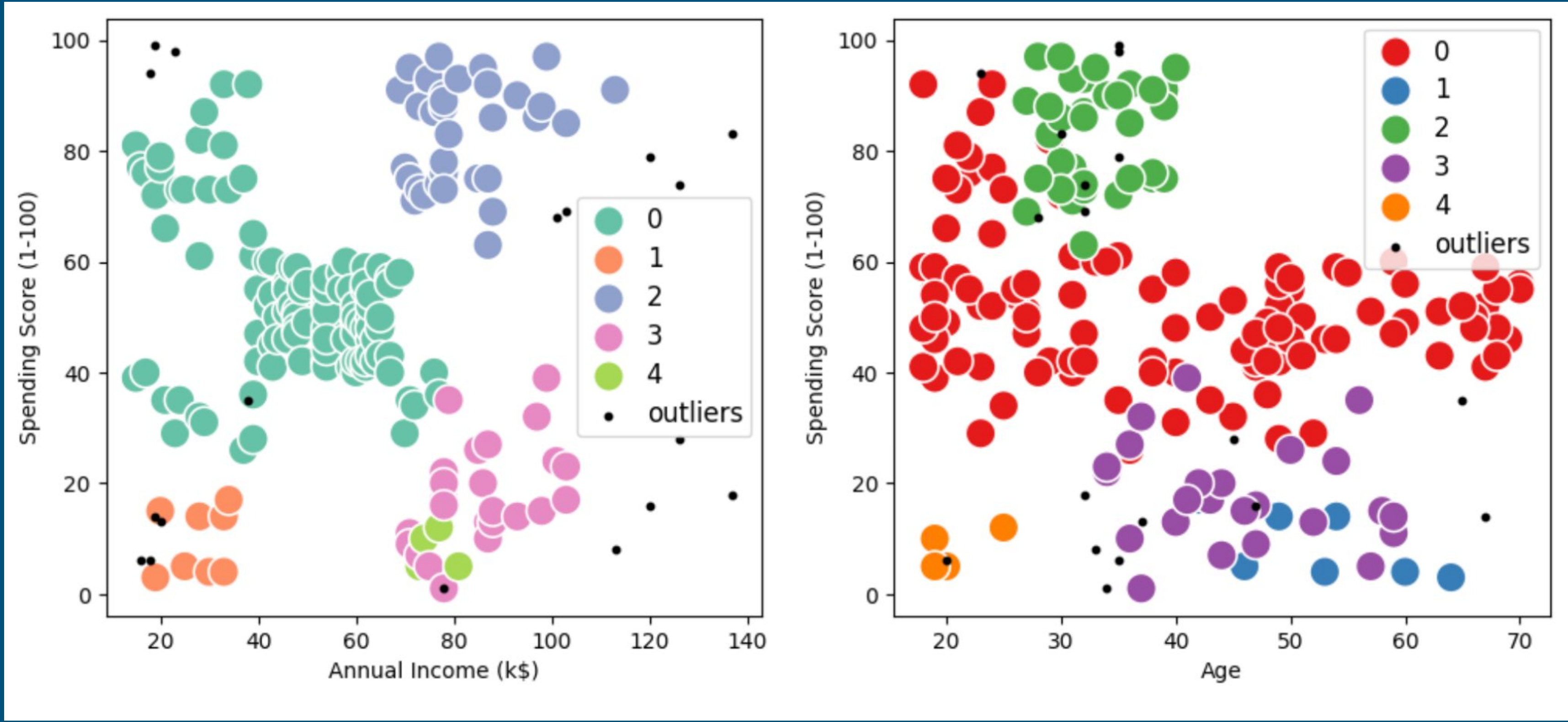
```
[5]: outliers = DBSCAN_dataset[DBSCAN_dataset['Cluster'] == -1]

fig2, axes = plt.subplots(1, 2, figsize=(12, 5))

sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',
                data=DBSCAN_dataset[DBSCAN_dataset['Cluster'] != -1],
                hue='Cluster', ax=axes[0], palette='Set2', legend='full', s=200)

sns.scatterplot(x='Age', y='Spending Score (1-100)',
                data=DBSCAN_dataset[DBSCAN_dataset['Cluster'] != -1],
                hue='Cluster', palette='Set1', ax=axes[1], legend='full', s=200)

axes[0].scatter(outliers['Annual Income (k$)'], outliers['Spending Score (1-100)'],
                s=10, label='outliers', c="k")
axes[1].scatter(outliers['Age'], outliers['Spending Score (1-100)'],
                s=10, label='outliers', c="k")

axes[0].legend()
axes[1].legend()

plt.setp(axes[0].get_legend().get_texts(), fontsize='12')
plt.setp(axes[1].get_legend().get_texts(), fontsize='12')

plt.show()
```

- Extracts outliers (noise points), which are points labeled -1.

6. Plotting the Clusters

```
[5]: outliers = DBSCAN_dataset[DBSCAN_dataset['Cluster'] == -1]

fig2, axes = plt.subplots(1, 2, figsize=(12, 5))

sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',
                data=DBSCAN_dataset[DBSCAN_dataset['Cluster'] != -1],
                hue='Cluster', ax=axes[0], palette='Set2', legend='full', s=200)

sns.scatterplot(x='Age', y='Spending Score (1-100)',
                data=DBSCAN_dataset[DBSCAN_dataset['Cluster'] != -1],
                hue='Cluster', palette='Set1', ax=axes[1], legend='full', s=200)

axes[0].scatter(outliers['Annual Income (k$)'], outliers['Spending Score (1-100)'],
                s=10, label='outliers', c="k")
axes[1].scatter(outliers['Age'], outliers['Spending Score (1-100)'],
                s=10, label='outliers', c="k")

axes[0].legend()
axes[1].legend()

plt.setp(axes[0].get_legend().get_texts(), fontsize='12')
plt.setp(axes[1].get_legend().get_texts(), fontsize='12')

plt.show()
```

- Creates a 1-row, 2-column subplot for visualizing clusters.
- Plots clusters based on Annual Income vs. Spending Score.
- Plots clusters based on Age vs. Spending Score.
- Then we plot outliers as black (c="k") dots.
- Later we add legends to both plots.
- Finally, we displays the scatter plots using plt.show()


Spending Score (1-100)
Annual Income (k$)
0
1
2
3
4
outliers
Spending Score (1-100)
Age
0
1
2
3
4
outliers

THANK YOU

Farhan Javed

BTECH 3rd Year

Roll No.: 12