Experiment - 07

Aim: To implement outlier detection using Density-Based method.

Theory:
Density-Based Outlier Detection: Density-based methods identify outliers based on the density of data points in their neighborhood. A point is considered an outlier if it resides in a low-density region, significantly differing from the dense clusters.

These methods are particularly useful for:

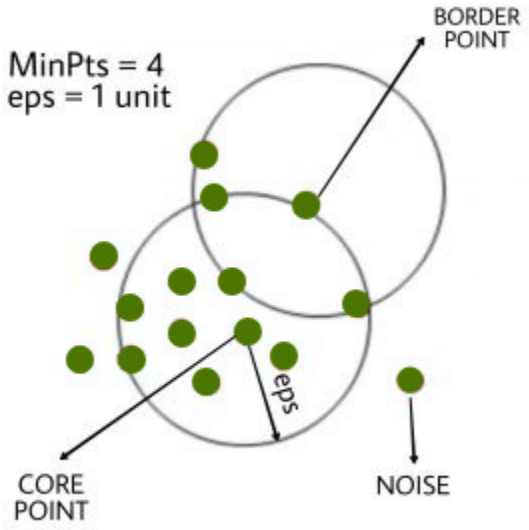
- Handling non-linear distributions of data.
- Detecting global as well as local outliers.
- Working well with high-dimensional data.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise): DBSCAN is a density-based clustering algorithm that also identifies outliers as points that do not belong to any cluster. It relies on two parameters:

- **ε (epsilon):** Defines the radius of a neighborhood around a point.
- **minPts:** The minimum number of points required to form a dense region.

A point is classified as:

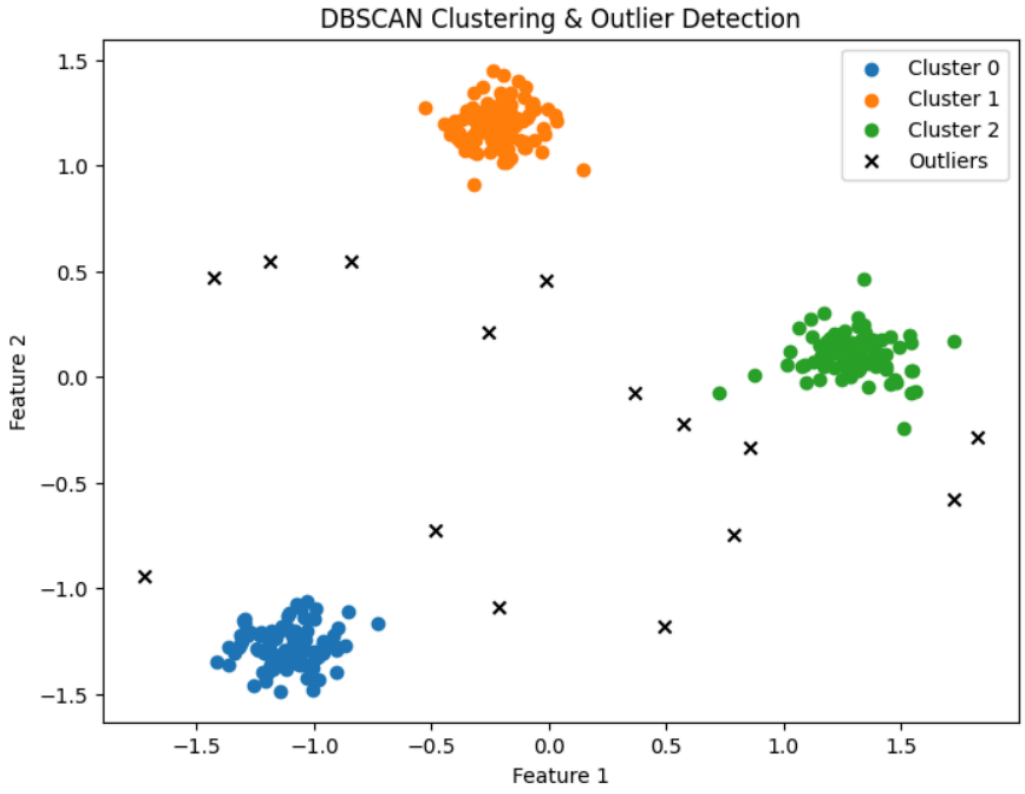
- **Core point:** If it has at least `minPts` within radius ε.
- **Border point:** If it has fewer than `minPts` neighbors but is in a core point's neighborhood.
- **Outlier (Noise point):** If it does not satisfy the above conditions.



Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

# Generate synthetic data with clusters and some noise
X, _ = make_blobs(n_samples=300, centers=3, cluster_std=0.6,
random_state=42)
X = np.vstack([X, np.random.uniform(low=-10, high=10, size=(20, 2))])
# Add noise points

# Standardize features (DBSCAN is sensitive to scale)
X = StandardScaler().fit_transform(X)

# Apply DBSCAN
dbscan = DBSCAN(eps=0.3, min_samples=5)
labels = dbscan.fit_predict(X)

# Identify outliers (DBSCAN labels them as -1)
outliers = X[labels == -1]

# Plot results
plt.figure(figsize=(8,6))
unique_labels = set(labels)
colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k']
for label in unique_labels:
    if label == -1:
        # Outliers in black
        plt.scatter(outliers[:, 0], outliers[:, 1], c='black',
marker='x', label='Outliers')
    else:
        plt.scatter(X[labels == label, 0], X[labels == label, 1],
label=f'Cluster {label}')

plt.legend()
plt.title("DBSCAN Clustering & Outlier Detection")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

Output:



Conclusion: Hence, we successfully implemented outlier detection using Density-Based method