Experiment No:9

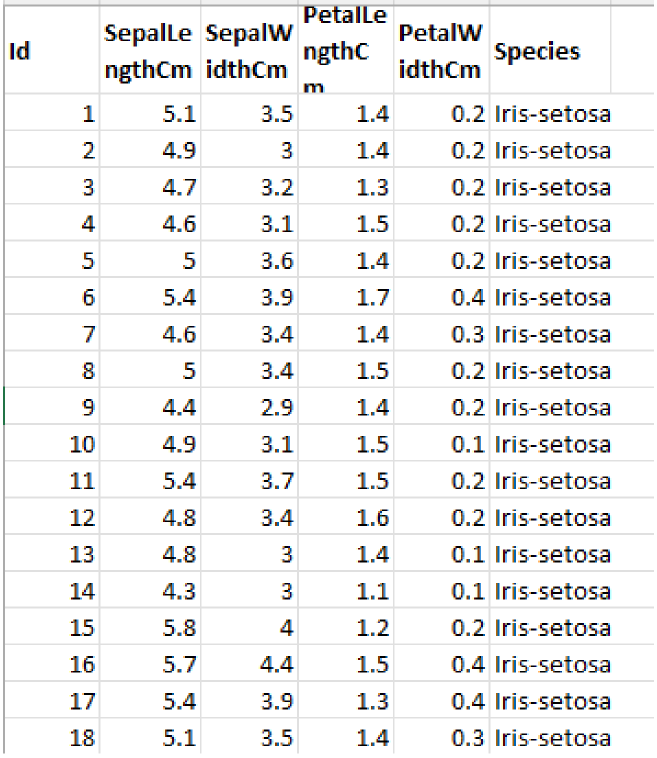
Course Outcome: CO5 **Blooms Level: L3**

Aim: To implement DBSCAN (Density-Based Spatial Clustering of Applications with Noise) clustering algorithm

Abstract:DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is an unsupervised machine learning algorithm used for identifying clusters of arbitrary shape and detecting noise in datasets. Unlike traditional clustering techniques such as K-Means, DBSCAN does not require prior knowledge of the number of clusters. It works by grouping together points that are closely packed based on two key parameters: the neighborhood radius (ε) and the minimum number of points required to form a dense region (MinPts). Points that do not belong to any cluster are classified as noise. DBSCAN is particularly effective for datasets containing clusters of varying densities and is robust to outliers. Its ability to automatically determine the number of clusters and handle complex structures makes it a powerful tool for exploratory data analysis and real-world applications such as anomaly detection, image segmentation, and spatial data mining.

Sample Input and Output:

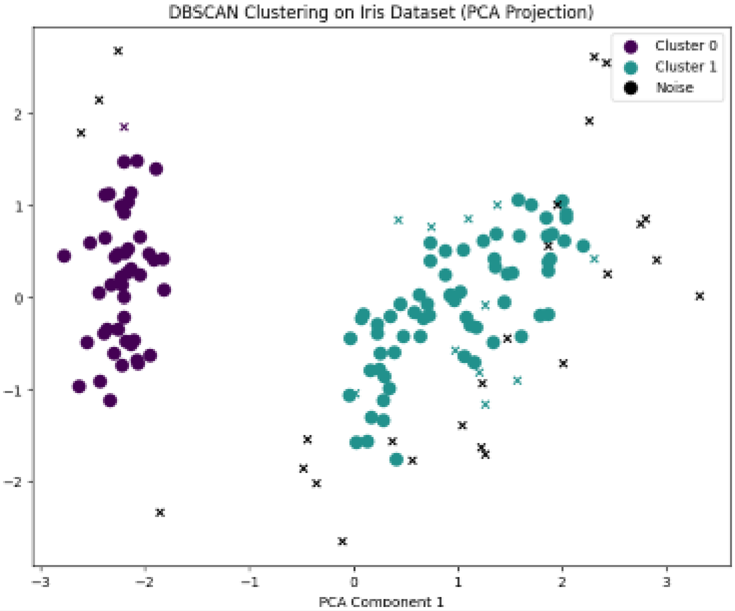
Case 1:

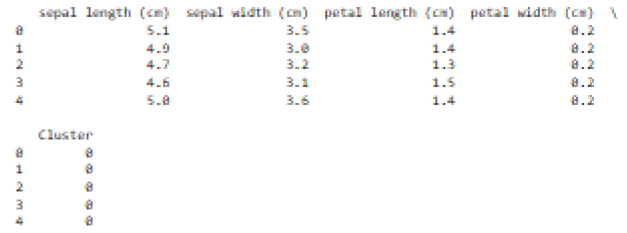


Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa

Sample Output

```
Cluster label distribution:
 Cluster
 1    78
 0    46
-1    26
Name: count, dtype: int64
```





```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   Cluster
0        0
1        0
2        0
3        0
4        0
```

Theory:

2.1 Introduction

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is an unsupervised machine learning algorithm that groups together points that are closely packed and marks points that lie alone in low-density regions as *outliers*.

Unlike K-Means, DBSCAN does not require the number of clusters to be specified beforehand. It can discover clusters of arbitrary shapes and sizes.

2.2 Key Concepts

- **Core Points:** Points that have at least a minimum number of other points (MinPts) within a distance (ε).
- **Border Points:** Points that are reachable from a core point but have fewer than MinPts within ε.
- **Noise Points:** Points that are not reachable from any core point.

2.3 Important Parameters

Parameter	Description
`eps (ε)`	Maximum distance between two samples to be considered as neighbors.
`min_samples`	Minimum number of samples required to form a dense region (core point).

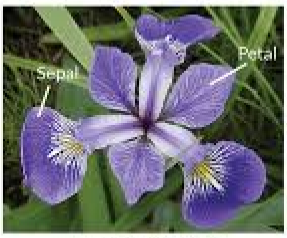
2.4 Advantages

- Can find clusters of arbitrary shape.
- Automatically detects outliers/noise.
- Does not require number of clusters beforehand.

2.5 Limitations

- Performance depends on parameter tuning (ε and MinPts).
- Not suitable for datasets with varying densities.

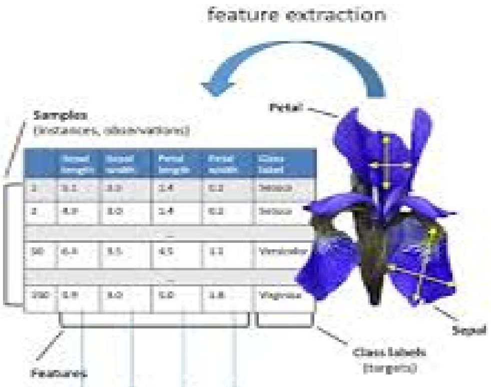
Program:
Dataset Information
The Iris dataset consists of 150 samples of iris flowers, 3 classes (Setosa, Versicolor, and Virginica) of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2. The latter are not linearly seperable from each other.
Attribute information:
1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm.
5. class:-Iris Setosa--Iris Versicolour—Iris Virginica



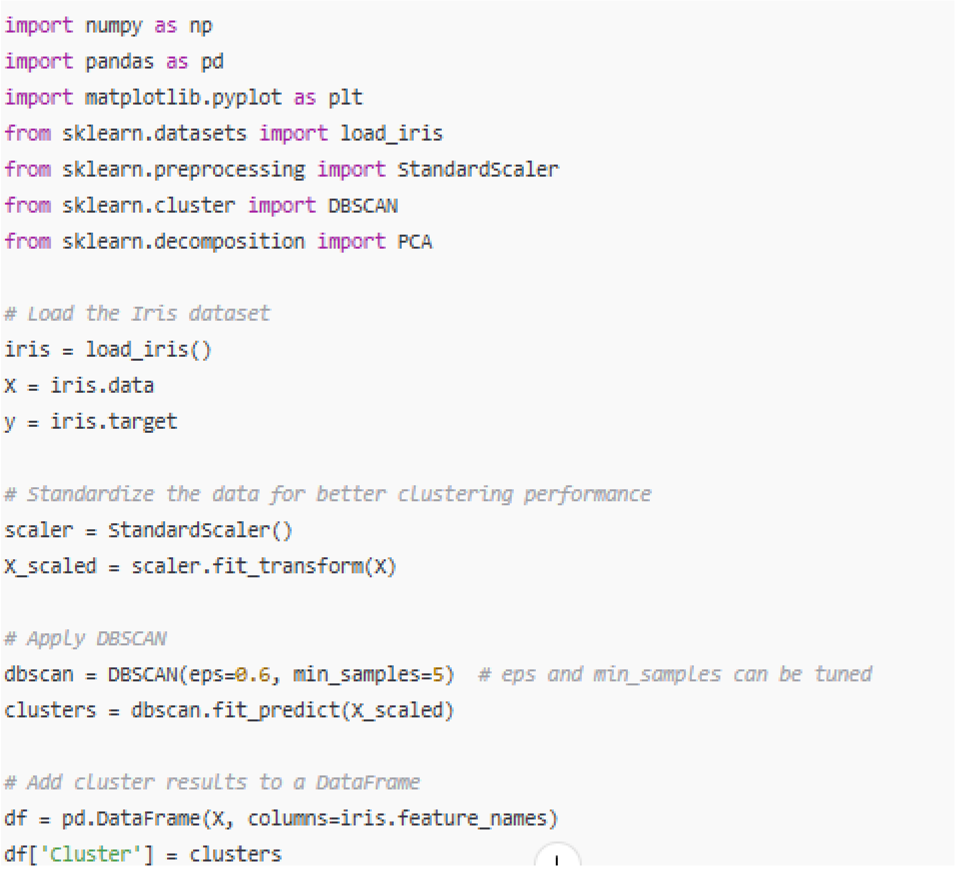
Iris Versicolor

Iris Setosa

Iris Virginica



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica



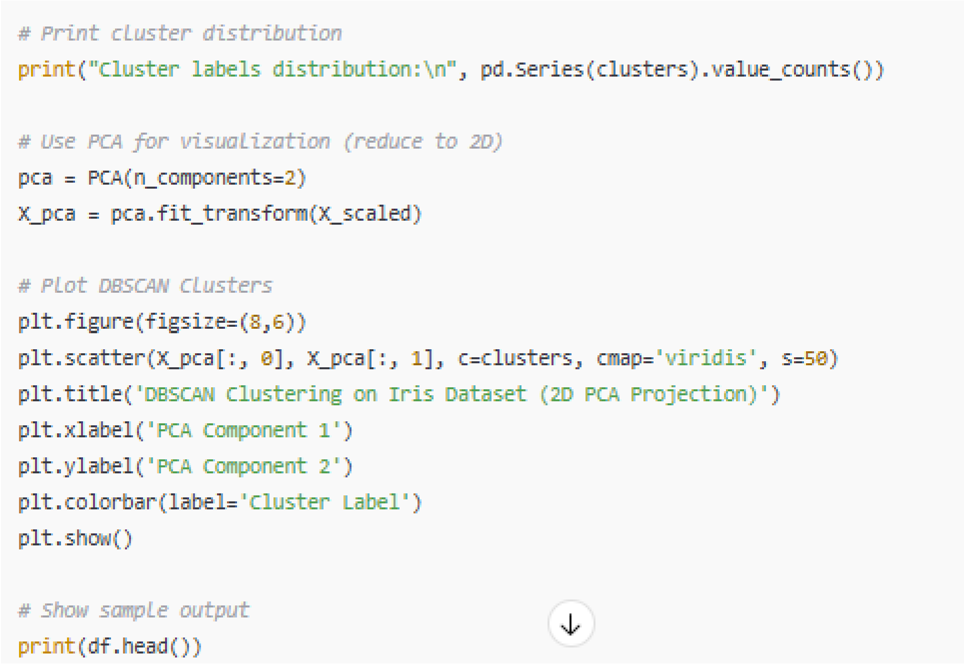
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN
from sklearn.decomposition import PCA

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Standardize the data for better clustering performance
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply DBSCAN
dbscan = DBSCAN(eps=0.6, min_samples=5)  # eps and min_samples can be tuned
clusters = dbscan.fit_predict(X_scaled)

# Add cluster results to a DataFrame
df = pd.DataFrame(X, columns=iris.feature_names)
df['Cluster'] = clusters
```



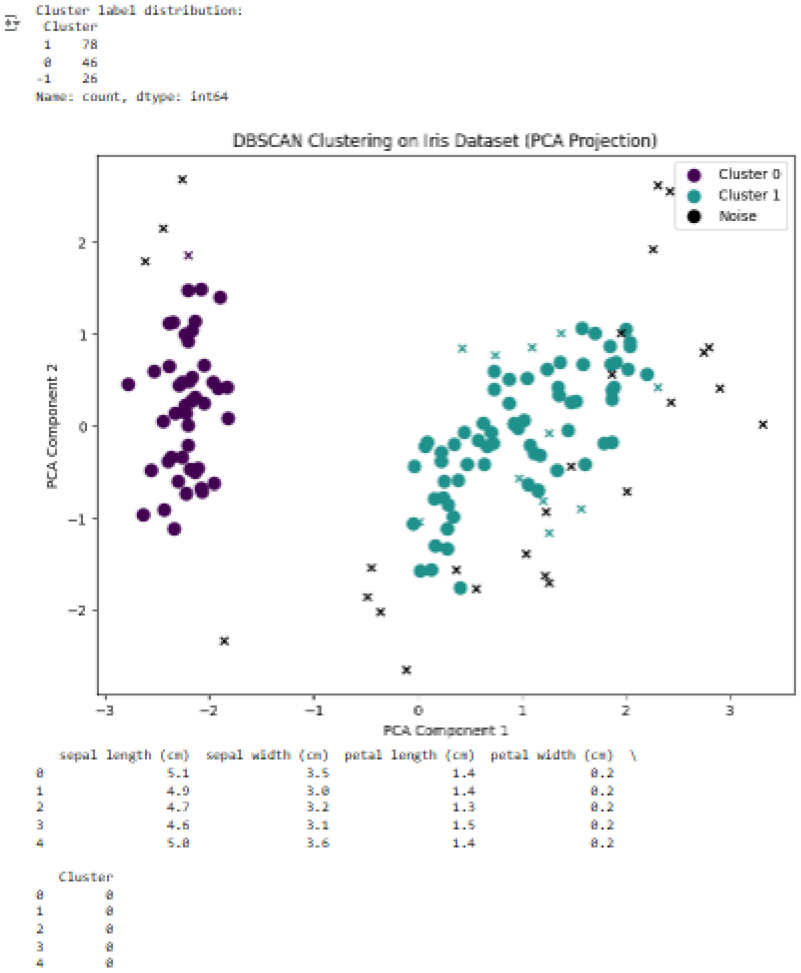
```
# Print cluster distribution
print("Cluster labels distribution:\n", pd.Series(clusters).value_counts())

# Use PCA for visualization (reduce to 2D)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Plot DBSCAN Clusters
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters, cmap='viridis', s=50)
plt.title('DBSCAN Clustering on Iris Dataset (2D PCA Projection)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster Label')
plt.show()

# Show sample output
print(df.head())
```

Output:



```
Cluster label distribution:
 Cluster
 1    78
 0    46
-1    26
Name: count, dtype: int64
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   Cluster
0        0
1        0
2        0
3        0
4        0
```

ConclusionThe experiment demonstrates how the DBSCAN algorithm can detect clusters of varying shapes and densities, while also identifying outliers in a dataset. Unlike K-Means, it is robust to noise and does not require specifying the number of clusters.

Exercise 1:

Implement the **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)** algorithm to identify and visualize clusters in a non-linear dataset.How can the **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)** algorithm be implemented to identify clusters and detect noise in a **non-linear dataset** such as the *Make Moons* dataset, and how effectively does it handle irregularly shaped clusters compared to traditional clustering methods like K-Means?

Students shall draw flowchart of exercise question in the writeup and submit.

Exercise 2:

How can the DBSCAN algorithm be applied to a large, real-world dataset such as the *Mall Customers dataset* to discover meaningful clusters of customers with similar income and spending patterns, while detecting outlier behaviors that may represent exceptional or unusual spending profiles?

Students shall draw flowchart of exercise question i n the writeup and submit.