**NAME OF EXPERIMENT**: -Implementation DBSCAN algorithm for clustering analysis using Python.

**INTRODUCTION**: DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups together data points that are closely packed together in high-density regions, while also identifying and removing points that are not part of any cluster (noise). The DBSCAN algorithm has two important parameters: epsilon and minPts. Epsilon determines the radius around each data point, and minPts determines the minimum number of points required to form a dense region or cluster. These parameters can be difficult to set, and tuning them correctly is important for getting good clustering results.

**Epsilon (ε):** The distance that specifies the neighborhoods. Two points are considered to be neighbors if the distance between them is less than or equal to ɛ.

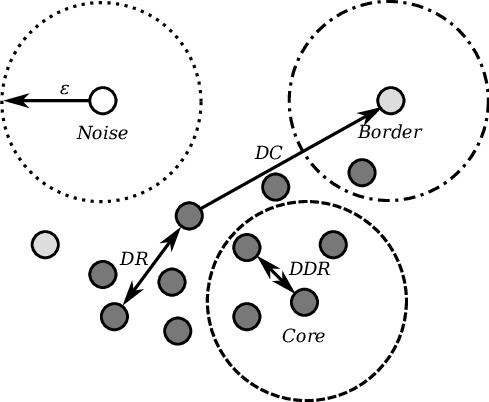
**MinPts:** The minimum number of points (a threshold) clustered together for a region to be considered dense.

Based on these two parameters, points are classified as core point, border point, or outlier:

**Core Point:** Data point that has at least minPts number of points within epsilon (ε) distance.

**Border Point:** Data point that has at least one core point within epsilon (ε) distance and lower than minPts number of points within epsilon (ε) distance from it.

**Noise or Outlier Point:** Data point that has no core points within epsilon (ε) distance.



DBSCAN Algorithm: -   
Input:

D: a data set containing n objects

ε: The radius parameter, and

Minpts: the neighborhood density threshold

Output: A set of density-based clusters  
Method:

1. Mark all objects as unvisited.
2. Do until no object is unvisited.
3. {
4. Randomly select an unvisited object p.
5. Mark p as visited.
6. If the ε-neighborhood of p has at least Minpts objects
7. Create a new cluster C, and add p to C.
8. Let N be the set of objects in the ε-neighborhood of p.
9. For each point p’ in N

{

* If p’ is unvisited

-mark p' as visited.

-If the ε-neighborhood of p' has at least Minpts points, Add those points to N.

* If p' is not yet a member of any cluster, add p' to C.

}

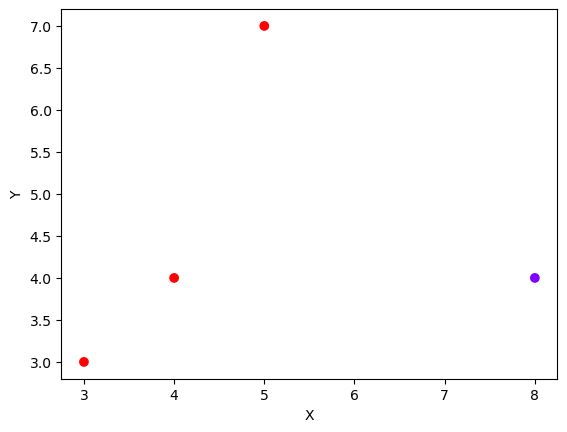
1. Output C.
2. Else mark p as noise
3. }

MANUAL CALCULATION: -

If Epsilon (ε)=3.5 and Minpts=2, what are the clusters that DBSCAN would discover with the following 4 data points shown in table:

|  |  |  |
| --- | --- | --- |
| Data Point | X | Y |
| S1 | 5 | 7 |
| S2 | 8 | 4 |
| S3 | 3 | 3 |
| S4 | 4 | 4 |

* Solution,



**S1**

**S2**

**S3**

**S4**

Step1: Obtain Euclidean distance among all the points.

Identify the neighbors of each point.

Euclidean distance =

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | S1 | S2 | S3 | S4 |
| S1 | 0 | 4.24 | 4.47 | 3.16 |
| S2 | 4.24 | 0 | 5.09 | 4 |
| S3 | 4.47 | 5.09 | 0 | 1.41 |
| S4 | 3.16 | 4 | 1.41 | 0 |

Point out the neighbors within the boundary of radius ε = 3.5

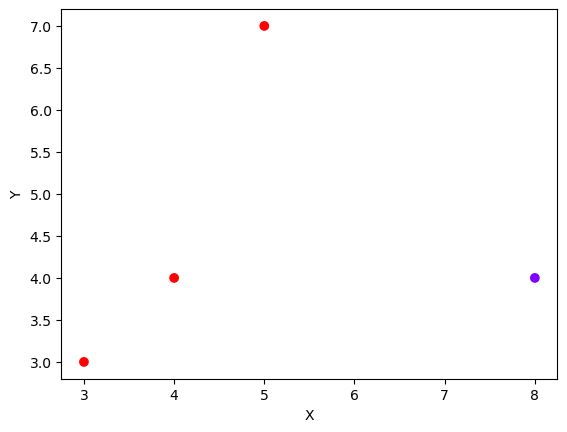
S1: S4,

S2: null,

S3: S4,

S4: S1, S3 {i.e., S4 <ε and S1, S3 <ε}

Step2: Assertion whether the points is the core point for the mean point 2 or not.



**S1**

**S2**

**S3**

**S4**

**Noisy data**

ADVANTAGES OF DBSCAN ALGORITHM: -

1. Does not require the number of clusters to be specified in advance, which makes it useful for datasets with unknown or variable cluster structures.
2. Can handle clusters of different shapes and sizes, and is robust to outliers and noise in the data.
3. Can detect clusters of arbitrary shapes and sizes, as long as they are separated by areas of low density.
4. Can be faster and more efficient than other clustering algorithms for large datasets.

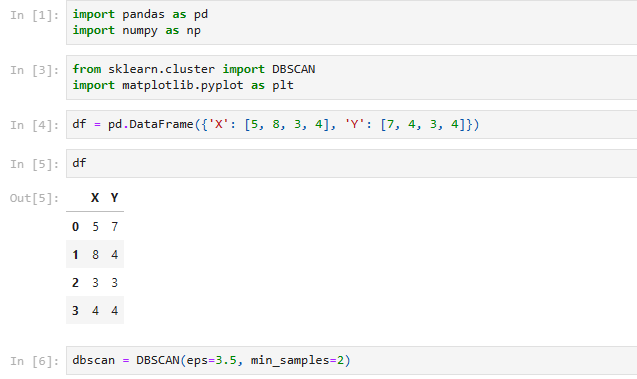
DISADVANTAGES OF DBSCAN CLUSTERING ALGORITHM: -

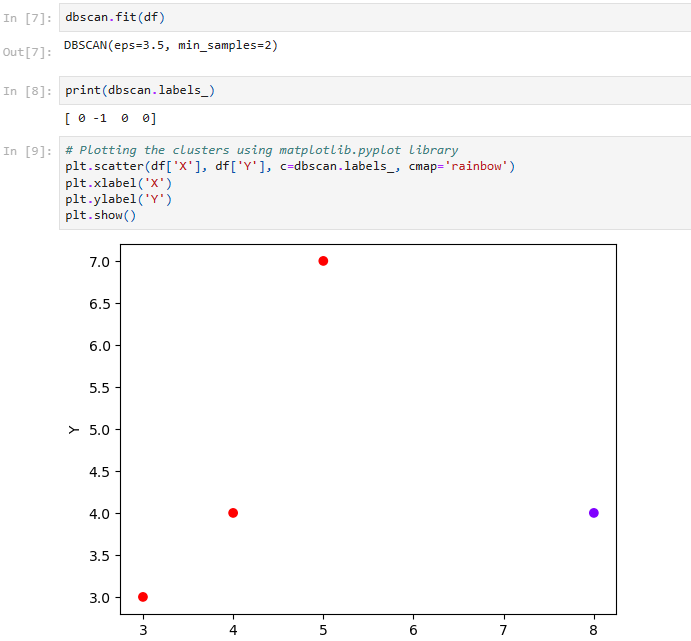
* Choosing appropriate values for hyperparameters (epsilon and minPts) can be difficult and time-consuming.
* Can produce different results depending on the choice of hyperparameters and initialization conditions.
* Does not work well with datasets that have clusters with vastly different densities.
* Can struggle with high-dimensional data due to the curse of dimensionality.

PROGRAM IMPLEMENTATION IN PYTHON: -

Requirement = Anaconda Navigator

Source Code: -





**CONCLUSION:** -

Hence, the DBSCAN algorithm using Python was successful implemented for cluster analysis .