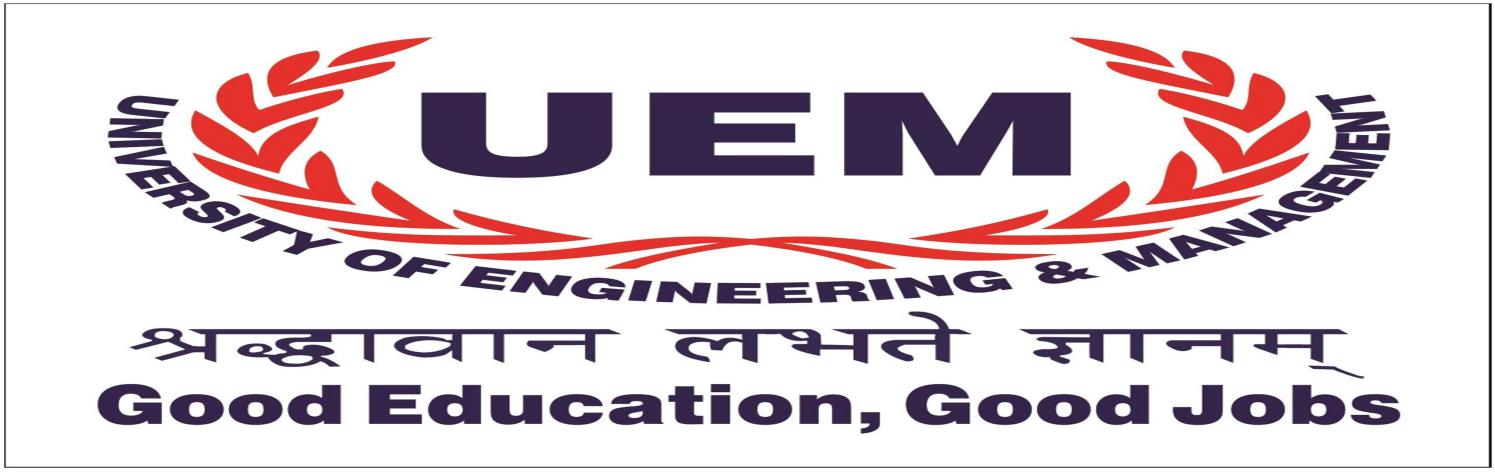


SUBJECT :- Design & Analysis Of Algorithm

SUBJECT CODE :- CS591

PROJECT REPORT ON :- DBSCAN Algorithm

Project By
Saptorshe Das (CSE3C – 72)
Aman Sharma (CSE 3C - 73)

INDEX:

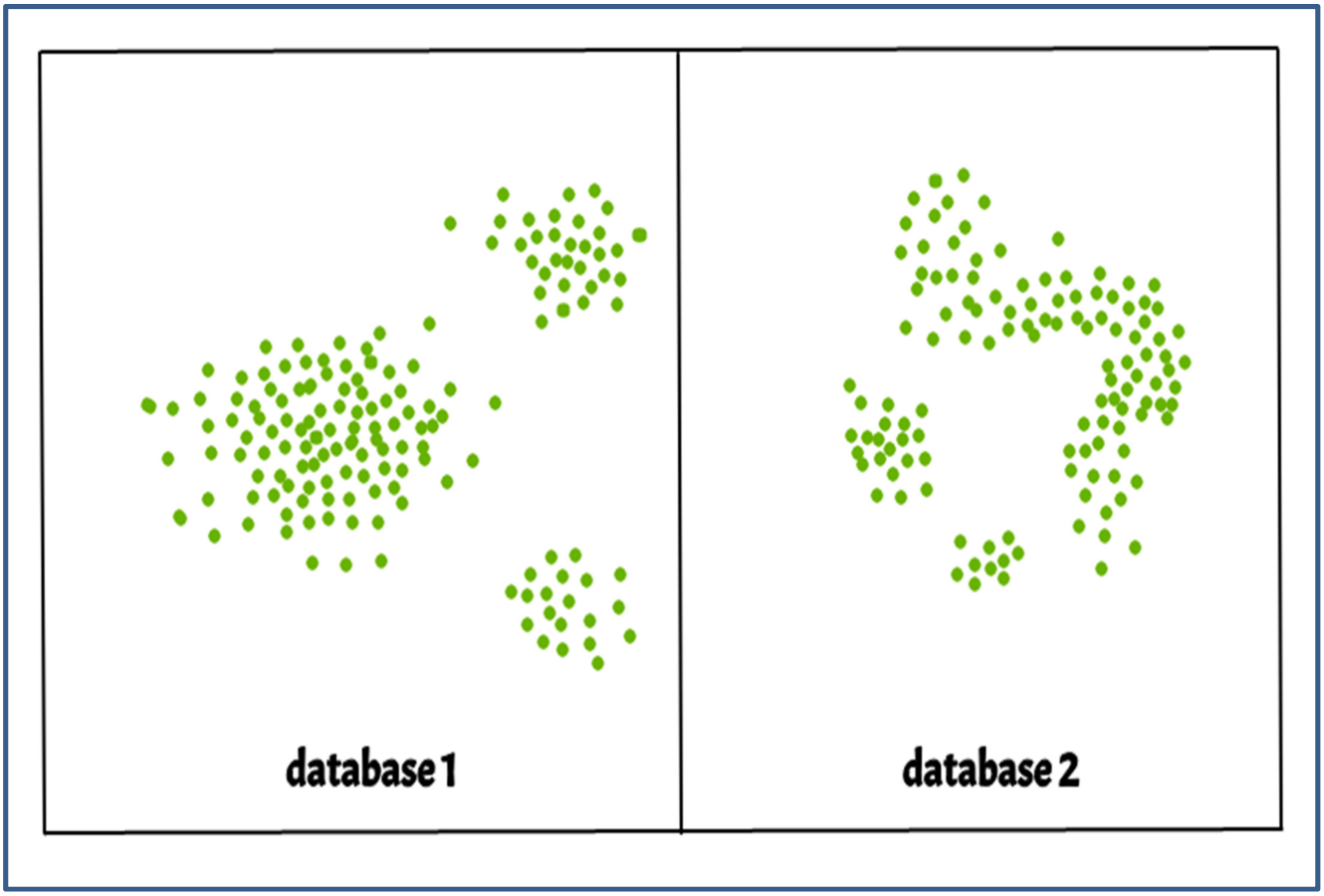
Acknowledgement

We, Saptorshe Das and Aman Sharma from CSE3C, would firstly like to thank our University for providing us with the infrastructure we used to learn about the subject. Secondly, we would thank Sukalyan Goswami sir,
HOD of Computer Science Engineering department, UEM Kolkata. Thirdly we would love to thank our concerned subject teachers Bipasha Mukherjee and other faculties for providing us with the project idea and necessary study material for the project.

Overview Of the Topic

Clustering analysis or simply Clustering is basically an Unsupervised learning method that divides the data points into a number of specific batches or groups, such that the data points in the same groups have similar properties and data points in different groups have different properties in some sense. It comprises of many different methods based on different evolution.
E.g. K-Means (distance between points), Affinity propagation (graph distance), Mean-shift (distance between points), DBSCAN (distance between nearest points), Gaussian mixtures (Mahalanobis distance to centers), Spectral clustering (graph distance) etc.

Fundamentally, all clustering methods use the same approach i.e. first we calculate similarities and then we use it to cluster the data points into groups or batches. Here we will focus on **Density-based spatial clustering of applications with noise** (DBSCAN) clustering method.
Clusters are dense regions in the data space, separated by regions of the lower density of points. The ***DBSCAN algorithm*** is based on this intuitive notion of “clusters” and “noise”. The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.



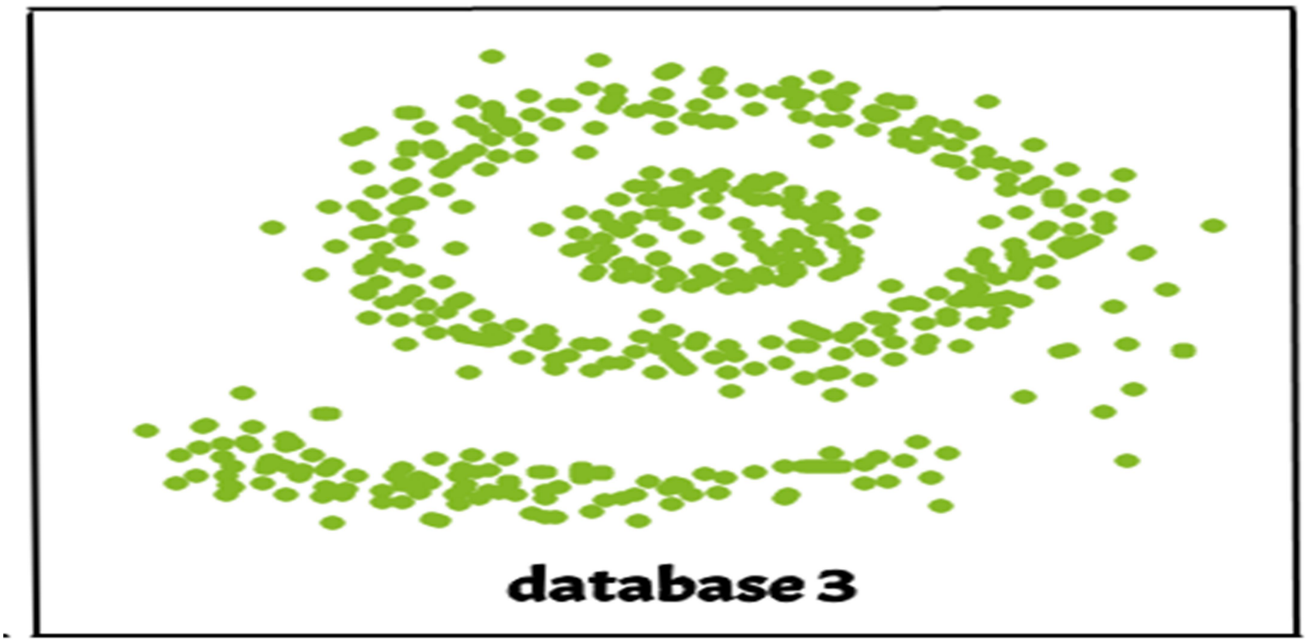
Why DBSCAN ?

Partitioning methods (K-means, PAM clustering) and hierarchical clustering work for finding spherical-shaped clusters or convex clusters. In other words, they are suitable only for compact and well-separated clusters. Moreover, they are also severely affected by the presence of noise and outliers in the data.

Real life data may contain irregularities, like –

i) Clusters can be of arbitrary shape such as those shown in the figure below.

ii) Data may contain noise.



The figure below shows a data set containing nonconvex clusters and outliers/noises. Given such data, k-means algorithm has difficulties for identifying these clusters with arbitrary shapes.

DBSCAN algorithm requires two parameters –

1. **eps** : It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to 'eps' then they are considered as neighbors. If the eps value is chosen too small then large part of the data will be considered as outliers. If it is chosen very large then the clusters will merge and majority of the data points will be in the same clusters. One way to find the eps value is based on the ***k-distance graph***.
2. **MinPts**: Minimum number of neighbors (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen. As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, `MinPts >= D+1`. The minimum value of MinPts must be chosen at least 3.

In this algorithm, we have 3 types of data points.
Core Point*: A point is a core point if it has more than MinPts points within eps.*
Border Point*: A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.*
Noise or outlier*: A point which is not a core point or border point.*

DBSCAN algorithm can be abstracted in the following steps –

1. Find all the neighbor points within eps and identify the core points or visited with more than MinPts neighbors.
2. For each core point if it is not already assigned to a cluster, create a new cluster.
3. Find recursively all its density connected points and assign them to the same cluster as the core point.
   A point *a* and *b* are said to be density connected if there exist a point *c* which has a sufficient number of points in its neighbors and both the points *a* and *b* are within the *eps distance*. This is a chaining process. So, if *b* is neighbor of *c*, *c* is neighbor of *d*, *d* is neighbor of *e*, which in turn is neighbor of *a* implies that *b* is neighbor of *a*.
4. Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

Below is the DBSCAN clustering algorithm in pseudocode:

```
DBSCAN(dataset, eps, MinPts){

# cluster index

C = 1

for each unvisited point p in dataset {

         mark p as visited

         # find neighbors

         Neighbors N = find the neighboring points of p


         if |N|>=MinPts:

             N = N U N'

             if p' is not a member of any cluster:

                 add p' to cluster C

}
```

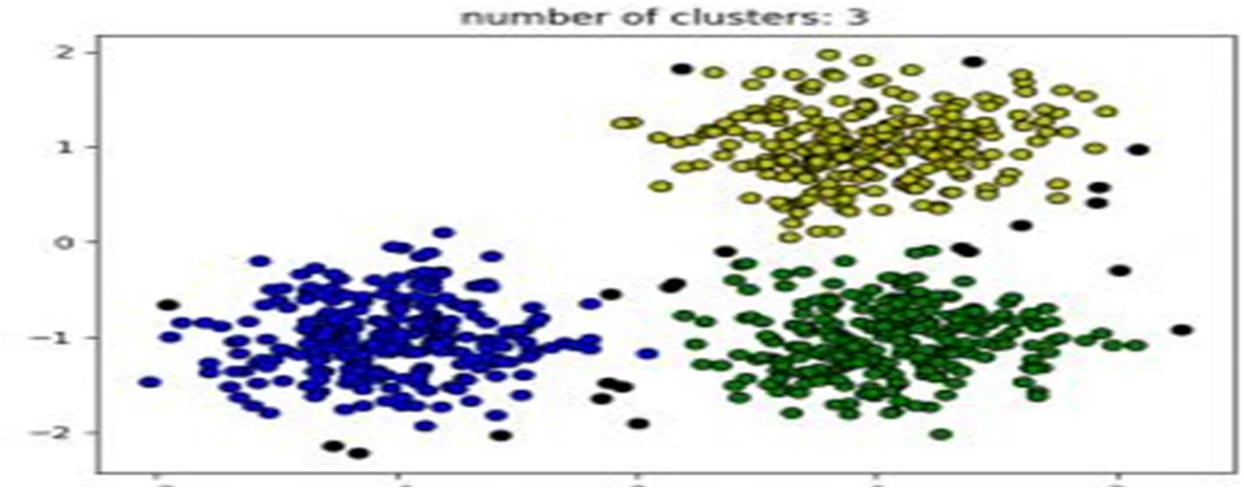
Python implemention Code :

Github Link : https://github.com/iamsaptorshe07/Algorith-Repo/blob/master/DBSCAN%20IMPLEMENTION.ipynb

Algorithm Code :

```
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets.samples_generator import make_blobs
from sklearn.preprocessing import StandardScaler
from sklearn import datasets

# Load data in X
db = DBSCAN(eps=0.3, min_samples=10).fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

print(labels)

# Plot result
import matplotlib.pyplot as plt

# Black removed and is used for noise instead.
unique_labels = set(labels)
colors = ['y', 'b', 'g', 'r']
print(colors)
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = 'k'

    class_member_mask = (labels == k)

    xy = X[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
                                      markeredgecolor='k',
                                      markersize=6)

    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
                                      markeredgecolor='k',
                                      markersize=6)

plt.title('number of clusters: %d' %n_clusters_)
plt.show()
```

Output :



Reference :

1 : https://en.wikipedia.org/wiki/DBSCAN

2 : https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/

3 : Dataset : https://www.kaggle.com/arjunbhasin2013/ccdata

4 : Sk-learn ML Package : http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

5 : Visualization Library : https://matplotlib.org/3.1.1/contents.html

REMARKS:

Date **Teacher's Signature**