Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

A.Y. 2022-2023

DATA MINING AND WAREHOUSE

AYUSH JAIN

COMPUTER ENGINEERING | TE – B2 | 60004200132

EXPERIMENT – 10

Aim: Study experiment on Spatial Mining

Theory:

1. Spatial Association Rules:

Spatial association means connectedness or relationship between and among variables over space. A single variable may be spatially autocorrelated; that is, values of the variable are somehow connected or related spatially. Many variables may be associated one with another at one or more sites. If there is spatial interaction there is also spatial association. Maps can depict spatial association. A mathematical shorthand technique can be used to represent, in general, measures of spatial association. Scientists test or theorize about variables to determine whether spatial association, either observed or expected, actually can be confirmed. Statistical procedures that have been developed for identifying and measuring the existence of spatial association are outlined.

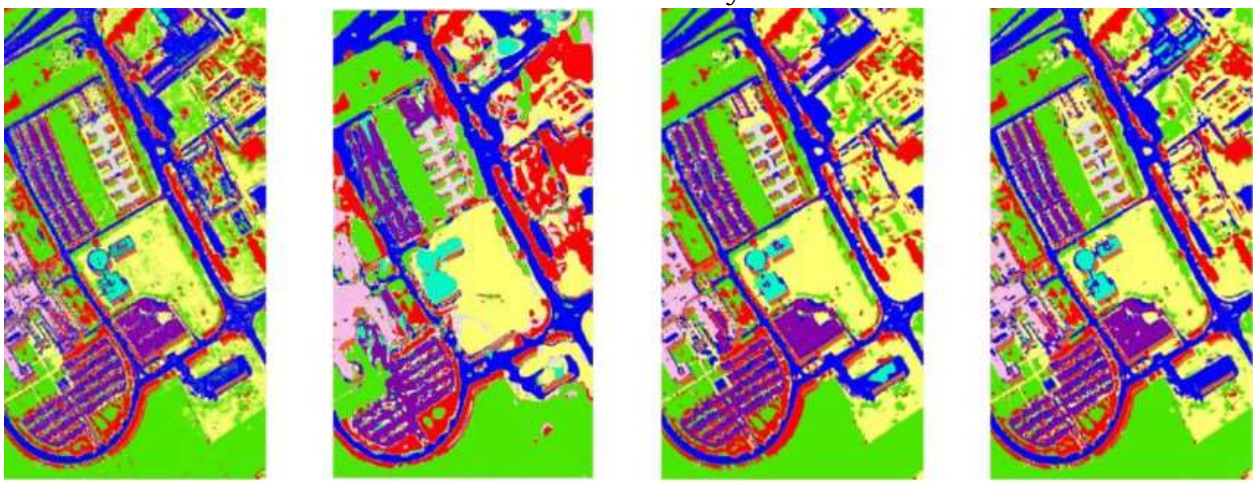
2. Spatial Classification:

Spatial classification is a type of machine learning technique that is used to assign labels to data points based on their spatial relationships. This is often used in applications such as image or geospatial data analysis, where the spatial location of the data points is an important factor in determining their classification.

Spatial classification algorithms can be divided into two main categories: supervised and unsupervised. In supervised spatial classification, the algorithm is trained on a labeled data set, where the correct labels for each data point are known. The algorithm uses this training data to learn the relationship between the spatial location of the data points and their labels, and can then be applied to new data points to predict their labels.

In unsupervised spatial classification, the algorithm is not given any labeled training data. Instead, it must discover the relationships between the data points and their labels by analyzing the data itself. This can be done using techniques such as clustering, where the algorithm groups data points into clusters based on their spatial relationships.

Some common algorithms used for spatial classification include decision trees, support vector machines, and k-nearest neighbors. These algorithms can be applied to a wide range of spatial data sets, including image data, geospatial data, and sensor data. The choice of which algorithm to use for a particular classification problem depends on the characteristics of the data and the desired accuracy of the model.



(a) SVM (94.33%) (b) **M²S³VM** (99.01%) (c) LORSAL-MLL (97.75%) (d) MPM-LBP (97.71%)

- Spatial classification is a type of machine learning technique that is used to assign labels to data points based on their spatial relationships.
- Spatial classification algorithms can be divided into two main categories: supervised and unsupervised.
- In supervised spatial classification, the algorithm is trained on a labeled data set, where the correct labels for each data point are known.
- In unsupervised spatial classification, the algorithm must discover the relationships between the data points and their labels by analyzing the data itself.
- Some common algorithms used for spatial classification include decision trees, support vector machines, and k-nearest neighbors.
- Spatial classification can be applied to a wide range of spatial data sets, including image data, geospatial data, and sensor data.
- The choice of which algorithm to use for a particular classification problem depends on the characteristics of the data and the desired accuracy of the model.

*Algorithms*1. Decision Trees
2. Support Vector Machines (SVMs)

3. K-Nearest Neighbors (KNN)
4. Random Forests
5. Neural Networks

These are just a few examples of algorithms that can be used for spatial classification. The choice of which algorithm to use for a particular classification problem depends on the characteristics of the data and the desired accuracy of the model. Other factors to consider when selecting a spatial classification algorithm include the computational complexity of the algorithm, the amount of training data available, and the availability of specialized libraries or software for implementing the algorithm.

. Spatial Clustering DBScan:

Spatial clustering can be defined as the process of grouping objects with certain dimensions into groups such that objects within a group exhibit similar characteristics when compared to those which are in the other groups. It is an important part of spatial data mining since it provides certain insights into the distribution of data and characteristics of spatial clusters. Spatial clustering methods are mainly categorized into four: Hierarchical, Partitional, Density based and Grid based. All those categorizations are based on the specific criteria used in grouping similar objects. In this paper, we will introduce each of these categories and present some representative algorithms for them. In addition, we will compare these algorithms in terms of four factors such as time complexity, inputs, handling of higher dimensions and handling of irregularly shaped clusters.

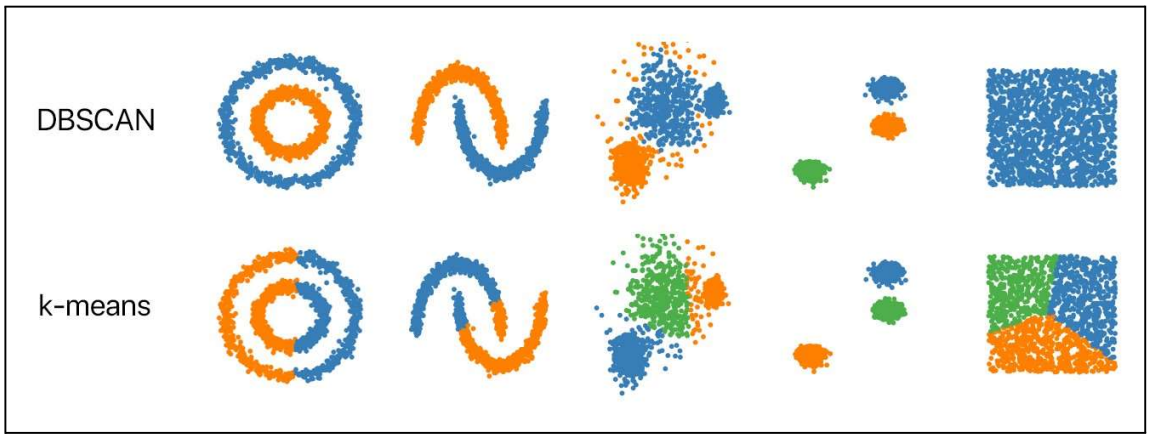
Density-based approaches apply a local cluster criterion. Clusters are regarded as regions in the data space in which the objects are dense, and which are separated by regions of low object density (noise). These regions may have an arbitrary shape and the points inside a region may be arbitrarily distributed. The important algorithms in this category includes DBSCAN,DENCLUE,OPTICS etc

Why do we need a Density-Based clustering algorithm like DBSCAN when we already have K-means clustering?

K-Means clustering may cluster loosely related observations together. Every observation becomes a part of some cluster eventually, even if the observations are scattered far away in the vector space. Since clusters depend on the mean value of cluster elements, each data point plays a role in forming the clusters. A slight change in data points might affect the clustering outcome. This problem is greatly reduced in DBSCAN due to the way clusters are formed. This is usually not a big problem unless we come across some odd shape data.

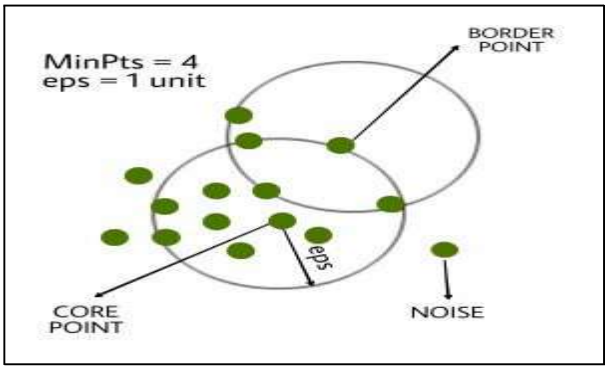
Another challenge with k-means is that you need to specify the number of clusters ("k") in order to use it. Much of the time, we won't know what a reasonable k value is a priori.

What's nice about DBSCAN is that you don't have to specify the number of clusters to use it.



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 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 the 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.



DBSCAN Algorithm:

Step 1 : Find all the neighbor points within eps and identify the core points or visited with more than MinPts neighbors.

Step 2 : For each core point if it is not already assigned to a cluster, create a new cluster.

Step 3 : Find recursively all its density connected points and assign them to the same cluster as the core point.

Step 4 : A point a and b are said to be density connected if there existst 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.

Step 5 : Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

Program:

CLARANS Algorithm

```
import numpy as np


# Calculate the distance between two points
def distance(a, b):
    return np.sqrt(np.sum((a - b) ** 2))


# Initialize the CLARANS algorithm
def clarans(data, num_clusters, num_local_minima, max_neighbors):
    # Initialize the clusters randomly
    clusters = [[data[i]] for i in np.random.choice(len(data),
num_clusters, replace=False)]


    # Initialize the best configuration found so far
    best_cost = float("inf")
    best_clusters = None


    # Repeat the search a number of times
    for i in range(num_local_minima):
        # Pick a random non-empty cluster
        cluster_idx = np.random.choice([i for i, c in enumerate(clusters)
if len(c) > 0])
        cluster = clusters[cluster_idx]


        # Pick a random point in the cluster
        point_idx = np.random.choice(len(cluster))
        point = cluster[point_idx]
```

```
        # Calculate the cost of moving the point to each of the other
clusters
        costs = []
        for j, c in enumerate(clusters):
            if j    cluster_idx:
```

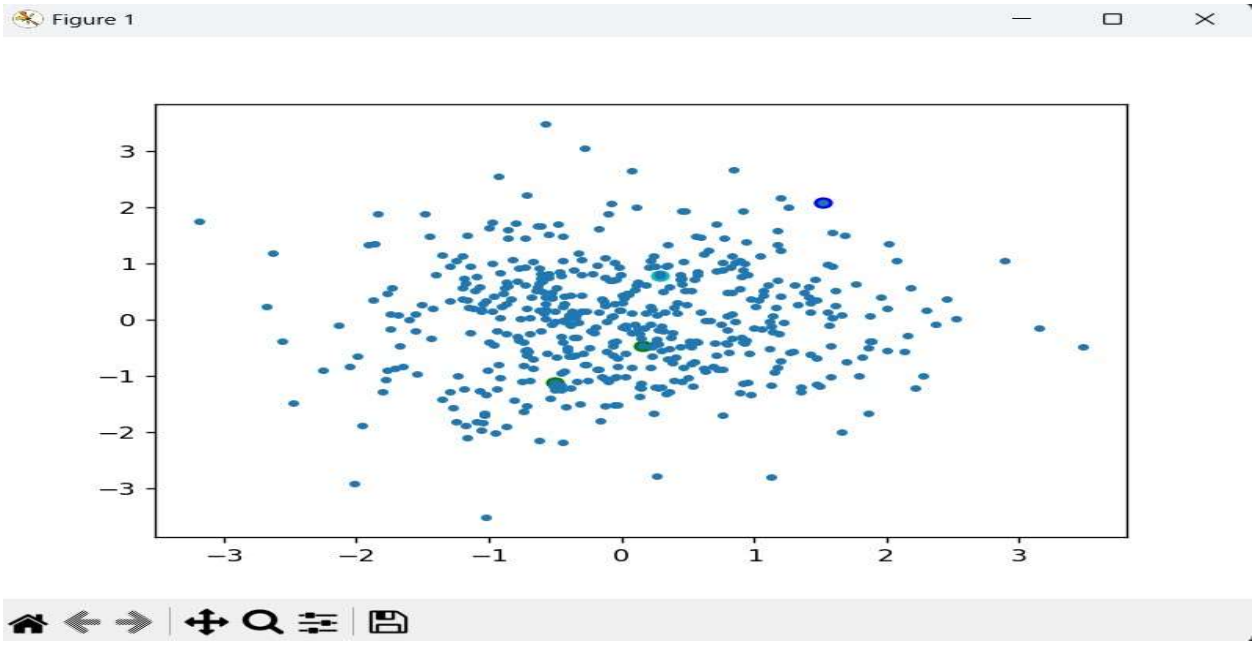
```
                continue
            cost = 0
            for p in c:
                cost += distance(p, point)
            costs.append(cost)

        # Move the point to the cluster with the lowest cost
        new_cluster_idx = np.argmin(costs)
        clusters[cluster_idx].pop(point_idx)
        clusters[new_cluster_idx].append(point)

        # Calculate the cost of the new configuration
        cost = 0
        for c in clusters:
            for p1 in c:
                for p2 in c:
                    cost += distance(p1, p2)
        cost    2

        # If the new configuration is better than the current best, update
the best configuration
        if cost < best_cost:
            best_cost = cost
            best_clusters = [c[:] for c in clusters]
```

```
        # If we have reached the maximum number of neighbors, start a new
search
        if i + 1    max_neighbors:
            clusters = [c[:] for c in best_clusters]

    # Return the best clusters found
    return best_clusters

# Generate some random data
```

```
data = np.random.randn(500, 2)

# Plot the data points
xs = [p[0] for p in data]
ys = [p[1] for p in data]
plt.plot(xs, ys, ".")

# Run the CLARANS algorithm with the generated data
clusters = clarans(data, num_clusters=4, num_local_minima=100,
max_neighbors=50)

# Print the clusters
for i, c in enumerate(clusters):
    print("Cluster %d: %d points" % (i, len(c)))

colors = ["b", "g", "r", "c"]
for i, c in enumerate(clusters):
    xs = [p[0] for p in c]
    ys = [p[1] for p in c]
    plt.scatter(xs, ys, c=colors[i])
```

```
# Show the plot
plt.show()
```

Output:




```
λ> python.exe clarans.py
Cluster 0: 1 points
Cluster 1: 2 points
Cluster 2: 0 points
Cluster 3: 1 points
```

Conclusion: Hence, we studied Spatial Association, Spatial Classification and Spatial Clustering and their clustering each. We also implemented CLARANS algorithm