Experiment 10

NAME:PARTH PAREKH **SAP ID:60004200006**

BATCH:A1 **BRANCH: COMPUTER**

Aim:

To write and explain one algorithm each on

1. Spatial Association Rules.
2. Spatial Classification.
3. Spatial Clustering - DBScan.

Theory:

1. Spatial Association Rules

Spatial association rules are association rules about spatial data objects. Either the antecedent or the consequent of the rule must contain some spatial predicates (such as near, close):

- Nonspatial antecedent and spatial consequent: All elementary schools are located close to single-family housing developments
- Spatial antecedent and nonspatial consequent If a house is located in Highland Park, it is expensive.
- Spatial antecedent and spatial consequent: Any house that is near downtown is south of Plano.

Support and confidence for spatial association rules is defined identically to that for regular association rules. Unlike traditional association rules, however, the underlying database being examined usually is not viewed as a set of transactions. Instead, it is a set of spatial objects.
The approach is similar to that discussed earlier for classification in that a two-step approach is used. As with traditional association rule algorithms, all association rules that satisfy the minimum confidence and support are generated by this algorithm. Because of the large number of possibilities for topological relationships, it is assumed that the data mining request indicates what spatial predicate(s) is to be used.
It is assumed that a data mining query is input. The query contains selection information that is used to retrieve the objects from the database that are of interest. The topological predicates defining the spatial relationships of interest are also input. Using these predicates, P, an initial table is built, CP, that identifies which pairs of objects satisfy P at a coarse level. The input minimum supports are actually a set of support values to be used at different levels in the processing is the support level to be used at the coarse filtering level. After this filtering, the pairs of objects that satisfy the coarse predicates are counted to see if their support is above the minimum. In effect, this frequent coarse predicate (FCP) database is the set of large one-itemsets. The predicates in FCP are then examined to find the frequent predicates at a fine level (FFP). The last step expands these frequent predicates of size 1 to all arbitrary predicate sizes and then generates the rules as with traditional association rules. This is performed similarly to Apriori. By finding the FCRs first, the number of objects to be examined is reduced at the last step.

Algorithm:

Input:

```
D        //Data, including spatial and nonspatial attributes
C        //Concept hierarchies
S        //Minimum support for levels
α        //Confidence
q        //query to retrieve interested objects
P        //Topological predicate(s) of interest
```

Output:

```
         R        //Spatial association rules
```

Spatial association rule algorithm:

```
         D’=q(D);
         CP is built by applying the coarse predicate version of P to D’;
         // CP consists of the set of coarse predicates satisfied by pairs of objects in D’.
         Determine the set of frequent coarse predicates FCP by finding the coarse predicates that
satisfy s;
         Find the set of frequent fine predicates FFP from FCP;
         Find R by finding all frequent fine predicates and then generating rules;
```

This algorithm works in a similar manner to the Apriori algorithm in that large "predicate sets" are determined. Here a predicate set is a set of predicates of interest. A 1-predicate might be ((clore.to, park)), so all spatial objects that are close to a park will be counted as satisfying this predicate. A 2-predicate could be ((close to, park), (south of. Plano)). Counts of 1-predicate sets are counted, then those that are large are used to generate 2-predicate sets, and these are then counted. In actuality. the algorithm can be used to generate multilevel association rules if desired or rules at a coarse level rather than a fine level.

2. Spatial Classification.

To partition collections of spatial objects, spatial classification issues are utilised. Nonspatial qualities, spatial predicates (spatial attributes), or spatial and nonspatial attributes might all be used to classify spatial objects. Concept hierarchies and sampling can both be employed. Generalization and progressive refinement techniques, like those employed in other forms of spatial mining, can be utilised to enhance efficiency.

Algorithm:

Spatial Decision Tree:

A two-step method, similar to that used for association rules, is utilised to create decision trees in one spatial categorization technique. The technique is based on the idea that spatial things may be defined using items that are close by. The classes are then supposed to be described using an aggregate of the most relevant predicates for neighbouring items. To construct the decision tree, the most relevant predicates (spatial and nonspatial) are first determined. It is hoped that this process will create smaller and more accurate decision trees. These relevant predicates are the ones that will be used to build the t is assumed that a training sample is used to perform this step and that decision tree. It is weights are assigned to attributes and predicates. Initial weights are 0. Two corresponding objects are examined for each object. The nearest miss is the spatial object closest to the target object that is in a different class. The nearest hit is the closest target in the same class. For each predicate value in the target object, if the nearest hit object the same value, then the weight of that predicate is increased. If it has a different value, then the weight is decreased. Likewise, the

weight is decreased (increased) if the nearest miss has the same (different) value. Only predicates with positive weights above a predefined threshold are then used to construct the tree. It is proposed that, because of the complexity of finding the relevant predicates, relevant predicates be found first at a coarse level and then at a finer level. MBRs, instead of actual objects, and a generalized coarse close to relationship are first used to find the relevant predicates. Then these relevant predicates and true objects are used during the second pass.
For each object in the sample, the area around it, called its buffer, is examined. A description of this buffer is created by aggregating the values of the most relevant predicates of the items in the buffer. Obviously, the size and shape of the buffer impact the resulting classification algorithm. It is possible, although unrealistic, to perform an exhaustive search around all possible buffer sizes and shapes. The objective would be to choose the one that results in the best discrimination between classes in the training set. This would be calculated using the information gain. Other approaches based on picking a particular shape were examined, and the authors finally used circles (equidistance buffers).
To construct the tree, it is assumed that each sample object has associated with it a set of generalized predicates that it satisfies. Counts of the number of objects that satisfy (do not satisfy) each predicate can then be determined. This is then used to calculate information gain as is done in ID3. Instead of creating a multiway branching tree, a binary decision tree is created. The resulting algorithm to construct the decision tree is shown below:

```
Input:
        D       //Data, including spatial and nonspatial attributes
        C       //Concept hierarchies
Output:
        T       //Binary decision tree
```

Spatial decision tree algorithm:

- find a sample S of data from D with known classification;
- identify the best predicates p to use for classification;
- determine the best buffer size and shape;
- using p and C, generalize the predicates for each buffer;
- build binary T using the generalized predicates and ID3;

3. Spatial clustering - DBScan

Clustering analysis is an unsupervised learning method that separates the data points into several specific bunches 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.
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 the Density-based spatial clustering of applications with noise (DBSCAN) clustering method.

Density-Based Clustering Algorithms (DBSCAN)

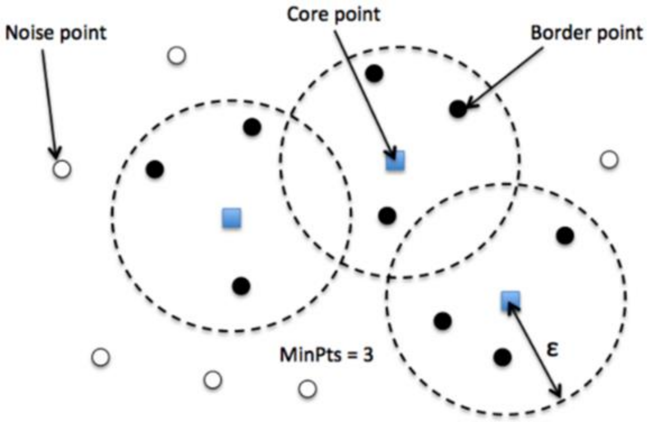
Density-Based Clustering is a term used to describe unsupervised learning approaches for identifying unique groups/clusters in data. It is based on the premise that a cluster in data space is a continuous region of high point density that is separated from other clusters by contiguous regions of low point density.
Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a base algorithm for density-based clustering. It can discover clusters of different shapes and sizes from a large amount of data, which is containing noise and outliers.

The DBSCAN algorithm uses two parameters:

- **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.
- **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.

These parameters can be understood if we explore two concepts called Density Reachability and Density Connectivity.
Reachability in terms of density establishes a point to be reachable from another if it lies within a particular distance (eps) from it.
Connectivity, on the other hand, involves a transitivity-based chaining-approach to determine whether points are located in a particular cluster. For example, p and q points could be connected if p->r->s->t->q, where a->b means b is in the neighbourhood of a.

There are three types of points after the DBSCAN clustering is complete:



- **Core**: This is a point that has at least m points within distance n from itself.
- **Border**: This is a point that has at least one Core point at a distance n.
- **Noise**: This is a point that is neither a Core nor a Border. And it has less than m points within distance n from itself.

Algorithmic steps for DBSCAN clustering

- The algorithm proceeds by arbitrarily picking up a point in the dataset (until all points have been visited).
- If there are at least 'minPoint' points within a radius of 'ε' to the point then we consider all these points to be part of the same cluster.
- The clusters are then expanded by recursively repeating the neighbourhood calculation for each neighbouring point.

Input:

```
D      //Data
MP     //MinPts or Minimum number of neighbours within eps radius
Eps    //Maximum distance for density measure
```

Output:

```
K      //Set of clusters
```

DBScan follows the algorithm:

1. Find all the neighbour points within eps and identify the core points or visited with more than MinPts neighbours.
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 neighbours and both the points a and b are within the eps distance. This is a chaining process. So, if b is neighbour of c, c is neighbour of d, d is neighbour of e, which in turn is neighbour of a implies that b is neighbour of a.

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

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
}
```

Conclusion:

Hence, we understood the different algorithms used in spatial rule mining, spatial classification and spatial clustering. We understood that Spatial association rule mining is the generation of rules indicating association relationships in a set of spatial data. It works in a similar way to association rule mining, but the underlying database is usually represented as a collection of spatial objects. Spatial classification challenges are used to categorise groups of items based on their spatial relationships. This can be accomplished through concept hierarchies as well as sampling. Spatial clustering finds groups or clusters in a set of spatial objects based on their distances. Spatial clustering techniques must be able to handle huge multidimensional databases efficiently. They should also be able to recognise groupings of various shapes.