TEAM 51

Foundation of Data Mining

Project 1

-By

Sheetal Tukaram Budbadkar

# Task:

Implement K-means and DBSCAN Algorithms

# K-MEANS ALGORITHM

Defined as: a function in clustering_algorithms.py

The pseudo code used to define the function is as follows:

```
Step 1: Initially randomly choose k data points as centroids
Step 2: Create a list of size k to store the value of centroids that will be calculated
Step 3: Until the list of centre point doesn't match the initial clusters repeat steps 4 to 8
Step 4: check
    a. if list of clusters is empty (This happens in 1st iteration). In this case the centre points to consider in calculation will be initial cluster centroids defined in step 1
    b. else the initial cluster value will be updated as current list of centres (which would have been calculated in last iteration) and that will also be considered as centre points
Step 5: for each point in data set repeat steps 6 and 7
Step 6: calculate distance between the point and all centre points
Step 7: Check with cluster centre is the closest and assign that number to the point
Step 8: for each cluster calculate new cluster centres
```

# DBSCAN ALGORITHM

Defined as: a function in clustering_algorithms.py

Clusters points are evaluated and defined as:

1. Noise: value =-1
2. Un-clustered: value = 0
3. All other points will be given value of cluster to which they belong

Uses functions:

a. generateIDs() -> generates values that can be used to define clusters
b. getEpsilonNeighbours () -> finds and returns the list of indexes of all points present within epsilon value

c. ExpandCluster() -> tries to assign each point to cluster

The pseudo code used to define the function is as follows:

Step 1: Create a list of size k to store the value of centroids that will be calculated.

Step 2: Create an variable clusterID using generateIDs() to create a list of ids that you can use to define clusters. Also store the first value as current cluster id

Step 3: for each points in data set repeat steps 4 and 5

Step 4: If the point is not clustered then call ExpandCluster () to define cluster.

Step 5: if the ExpandClluster() function defines any cluster just update cluster id.

Functions:

1. ExpandCluster()
   Step 1: get all indexes of points that are neighbours of current point using getEpsilonNeigbours() and store it in variable seeds
   Step 2: If the no of neighbours is less than value for minpnts then its it not a core object. You can assign that point as noise and return false along with the updated list where you marked the current point as noice.
   Step 3: for each point which are neighbours assign them cluster id
   Step 4: remove the current point from the list seeds
   Step 5: for all remaining points in seeds repeat steps 6 to 9
   Step 6: get neighbour points of current seed point
   Step 7: if seed point also has more than minpnts neighbours then it is also a core point. You can assign the cluster id to this point as well.
   Step 8: Add the point to list of seeds if its unclassified
   Step 9: remove the current seed point also from the list of seeds since that is now processed.

2. getEpsilonNeighbours()
   step 1: for all points in data set, repeat steps 2
   step 2: calculate the distance between current point and data point
   step 3: if the distance is less than epsilon. Then it means it's a neighbour point.
   Step 4: add this point to list of index of all neighbour point.