Text Clustering

Name: Vineela Velicheti
Student Id: 012445186
Rank: 27
F-1 Score: 0.179

Dbscan Pseudocode: Three main functions are involved in the dbscan approach followed. They are:

1. ***distance_calc***
2. ***points_classifier***
3. ***dbscan***

1. **distance_calc:** In this function the pairwise distances are calculated for all points in the given dataset. Initially a custom function has been used to calculate the distance but because of the computational time, later pairwise distances (from sklearn) has been used. The function returns a list of lists such that in each list the distance for one point with the remaining points in dataset is maintained.

```
/* takes input the dataset and number of rows in dataset*/
distance_calc(mat,nrows):
        distance_int = sklearn.metrics.pairwise.pairwise_distances(mat,mat)
        distance_ext = distance_int.tolist()
        return distance_ext /*list of lists with distances for one point in dataset with
remaining*/
```

2. **points_classifier:** The points in the dataset are classified as core, border and noise in this function.

```
points_classifier(distance_ext,nrows,eps,min_pts):
 /* 2 empty lists are created*/
         points_classification
         neighbor_count   /* list that maintains the count of neighbors inside eps for each
        point*/
  for all objects in dataset:
        assign ‘N’ initially in the points_classification list
  for all objects in distance_ext:
        new variable count is initialized to zero
        for each distance measure in each object in distance_ext:
                if(distance < eps):
                        count is incremented
   neighbor_count.append(count)
```

```
for all points in dataset:
   if(neighbor_count[i] > min_pts):
                  point is classified as Core Point
for all points in dataset:
   if corresponding value in points_classification is ‘N’:
         if point has one core point in its neighborhood:
            point is classified as Border Point
            break
return points_classification  /* remaining points are left as ‘N’ indicating noise*/
```

3. **dbscan:** The main function dbscan takes input data, eps and min_pts. It calls both the above functions and gets the points_classification list. This function also uses another function cluster_formation that forms clusters for all core points.

```
nrows = no. of rows in dataset

/* code to connect core points in clusters*/
for i in range of nrows:
      if(i not in connected_points):
         if(points_classification[i]=='C'):
            cluster=[]/* new empty cluster is formed for each point*/
            connected_points.append(i) /* i is added to connected points list*/
            clusterformation(i,points_classification,distance_ext,cluster,clusters_list,
                               connected_points,nrows,eps)
             clusters_list.append(cluster) /* list of lists such that each inner list consists of all
                                            points belonging to one cluster */

/* code to connect all border points to the cluster of their nearest core point*/
for i in range of nrows:
      if(i not in connected_points):
         if point is classified as border point:
            connected_points.append(i)
            nearest neighbor index for i is calculated using distance_ext[i]
                  if(points_classification[nn_index] is Core Point):
                     break
            for each cluster in clusters list:
                    for each index in the cluster:
                              if(index == nn_index):
                                        cluster.append(i) /* point is appended to the cluster list*/


/*code to connect all noise points to a noise cluster*/
```

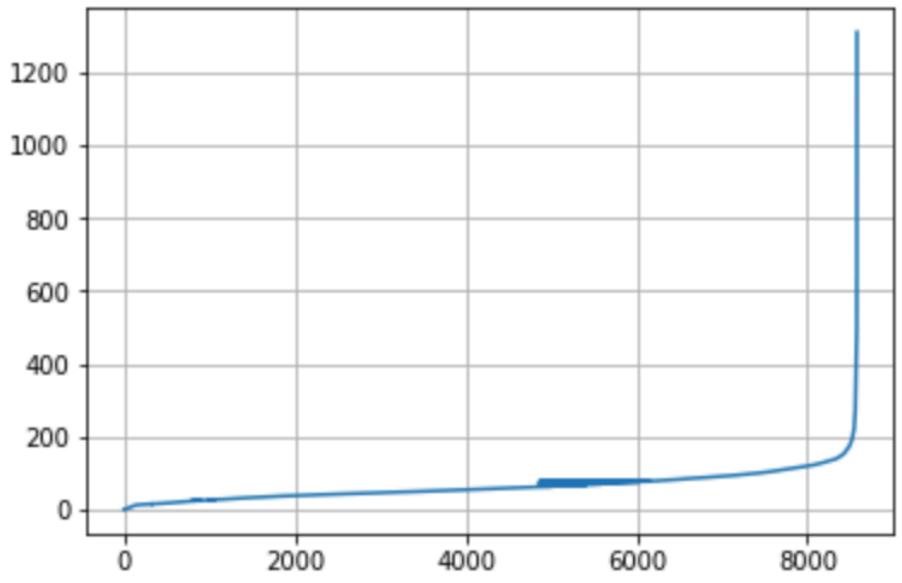
```
        for i in range(nrows):
                if(i not in connected_points):
                    if(points_classification[i]=='N'):
                        connected_points.append(i)
                        noise_cluster.append(i)
            clusters_list.append(noise_cluster)


        /*
        After all points are clustered we have cluster_list in which each list is a separate cluster
        A new list is created now telling us to which cluster each point belongs (expected format)
        */
        for i in range(nrows):
                for j in range(length of clusters_list):
                     for k in clusters_list[j]:
                                if(k==i):
                                        cluster_point_index.append(j)
                                         break
         return cluster_point_index

def clusterformation (i,points_classification,distance_ext,cluster,clusters_list,connected_points,
                        nrows,eps):
        cluster.append(i) /* point is allotted to a cluster*/
        for j in range(nrows):
                if (j inside eps of i and j not in connected points and j is a Core Point):
                        cluster.append(j)
                        connected_points.append(j)
                        for k in range(nrows):
                            if (k inside eps of j and k not in connected points and k is a Core Point):
                                connected_points.append(k)
                                cluster.append(k)

   return
```

Determining Eps: The eps value is determined for Minpts varying from 3 to 21 in step of 2 for the given dataset. For each k in the Minpts list, the distance to the kth nearest neighbor is calculated and sorted. This is used as y axis and a graph is plotted   to get the approximate eps value. The average of all the determined eps values was around 150. So eps value is initialized as 150 and then decreased in steps of 10 and best accuracy is achieved when eps is 97.



Silhouette score: Silhouette score was used as the internal measure to check the cluster validity. The silhouette score was calculated for all values in Minpts list with eps values as 80,97,150 . The graph for the number of clusters with silhouette score is constructed for better understanding.

Feature Selection: A feature selection algorithm Variance Threshold is implemented. A variance of 20% was given as initial threshold value but the number of dimensions was reducing to 19 which resulted in low accuracy. So, no threshold was given for better accuracy.