Programming Assignment 3

Rohan Kamat
013759252

Approach 1: The first approach was coding the dbscan algorithm without using K-means as a basic clustering algorithm.This was the basic building block of the final algorithm which was developed with the K-means cluster from sklearn as the base mechanism.The NMI score was 0.27 for this particular approach.

Approach 2:

The development was algorithm required a lot of whiteboarding, the first algorithm developed alongside Kmeans gave a F1 score of 0.49.The approach taken was to make the K Means cluster more dense as compared to the original cluster, this meant breaking the clusters into smaller clusters and then relabelling entire data.

The first thing was loading the csr matrix from the file, and the normalizing it, the odd iterations were appended to index and the even iteration to the values.The length of each row divided by 2 produced the ptr.This was passed to sklearn csr_matrix method.The code from activity 3 was used to convert each values to L2norm, and then truncated svd for dimensionality reduction.The silhouette coefficient was 0.088 for eps=0.25 and minpts=5.

The Algorithm:

```
Def dbscan(mat,labels,minpts,eps):
        #initialize the arrays
        core,label ,visited ,noise empty arrays of length of the rows are initiated.
        #initializing the cluster
        Set Cluster to 1
        #The number of clusters from Kmeans was 150
        For i in range(150):
                for iters, val in enumerate(mat):
                if labels[iters] is not equal to i and not math.isnan(visited[iters]):starting from 1st
        label in the Kmeans which is iterating through clusters and checking if it is already
        visited.
                elif labels[iters] is equal to i and math.isnan(visited[iters]):check if not yet visited
                Create n as an empty list, Initialize the neighbours
                                        For iters1,v in enumerate(mat):
                                                if labels[iters1] is equal to i and iters1 is equal to
                                iters:avoiding the match to iteself
                                                        Continue
                                                elif labels[iters1] equal to i:the first point of
                                cluster is found
                                                        sim is equal to cosine_similarity(v,
                                mat[iters])
                                                        if sim[0][0] is less than eps:finding the
                                cosine similarity
```

n.append(iters1)

If len(n) is greater than eps, Check if the length of neighbours n is greater than minpts

If yes then mark the 1 in the index of core point,set it as visited and assign label to it.

Set core[iter1] to 1

Set visited[iter1] to 1

For neigh in n:

If math.isnan(visited[neigh]):

Label[neigh] is equal to1

Visited[neigh] is equal to1

Assign labels to all the neighbours of the corepoint and mark them as visited.

Else consider it as border point and leave it.

Else:

Noise[iters] is 1

If it is not a core point and a border point, update the noise array as 1.

Increment the cluster

cluster+=1

Return label,noise,cluster

Call the function

l,n,c=dbscan(mat_normalized,labels,10,0.25)

The labels still contains some nan values which are noise,

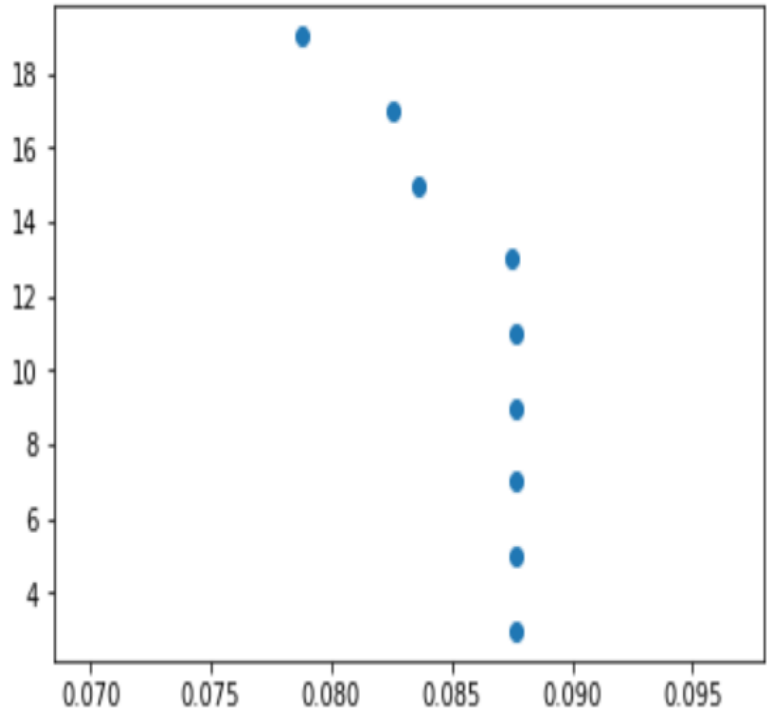
For values in noise which are 1 update the labels with the last possible cluster

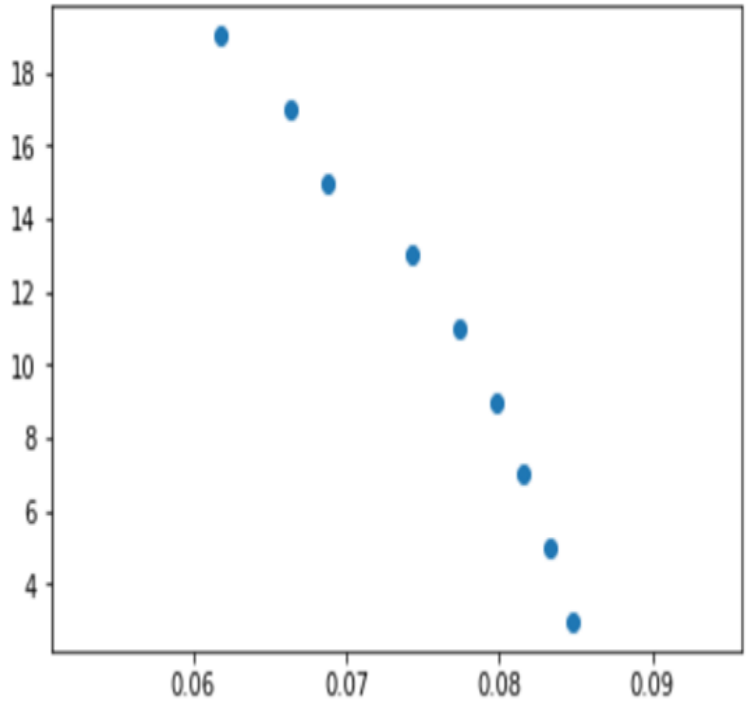
for iters,values in enumerate(l):

if math.isnan(values):

l[iters]=c+1

Write values of label to a file.





The silhouette metric from sklearn was used to calculate the error, the 2 scatters are plotted with eps=0.2,0.5 with minpts varying from 3,21 in the steps of 2.

Rank	NMI
8	0.50