SENG 474: Assignment 3

Name: Subah Mehrotra

[redacted]

Lloyd’s Algorithm (K-means)

This algorithm finds evenly spaced sets of points in subsets of Euclidean spaces and partitions of these subsets into well-shaped and uniformly sized convex cells. It repeatedly finds the centroid of each set in the partition and then re-partitions the input according to which of these centroids is closest. This algorithm was implemented using the Euclidean distance and two different kinds of initializations: uniform random initialization and kmeans ++ initialization.

Uniform Random Initialization

The data are partitioned randomly by k and centroids of these groups are appointed to be the initial centres. Thus, centres are calculated, not selected from the existent dataset cases. This method yields centres that lie close to each other and to the general centroid of the data. The number of clusters used was from 2 up to 7. The best number of clusters were seen by plotting a graph of error, which is the squared mean distance between two points, against the number of clusters.

Dataset 1

The kink was found to be when **K was 4**, therefore k=4 will be selected for dataset 1. It was seen that the error decreased rapidly as the value of K was increased from 2 to 4 and then the rate of decay of error slowed down between the K values of 4 to 8 (see fig 1).

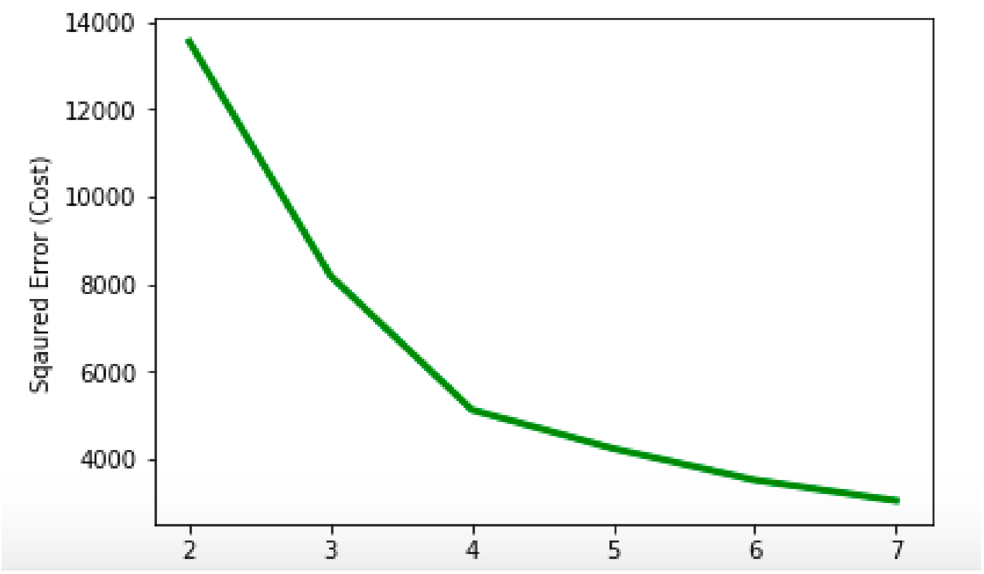

Figure 1: Graph between error against number of clusters for dataset 1

Figure 2 shows scatter plot with 4 clusters for dataset1:

```
Number of custers:
4
```

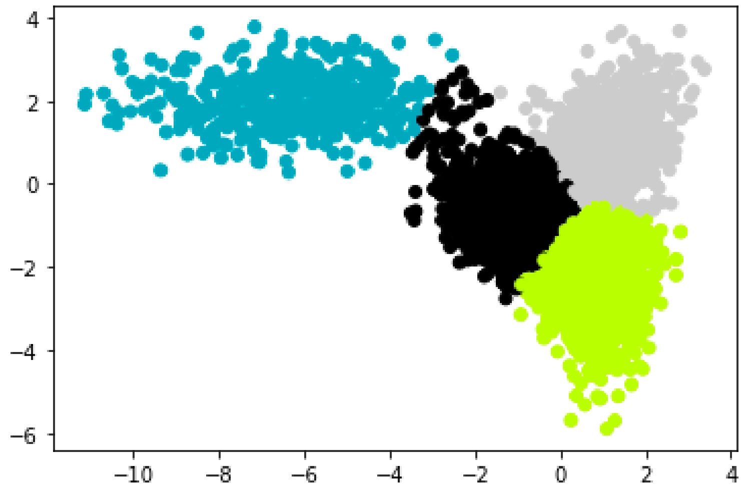

Figure 2: Scatter plot with 4 clusters for Dataset1

Dataset 2

The kink was hard to see for this dataset, it could have been either 3 or 7. I decided to go with 7 clusters, so the kink was found to be when **K was 7**, therefore k=7 will be selected for dataset 2. It was seen that the error decreased rapidly as the value of K was increased from 2 to 7 and then the rate of decay of error slowed down between the K values of 7 to 8 (see fig 3).

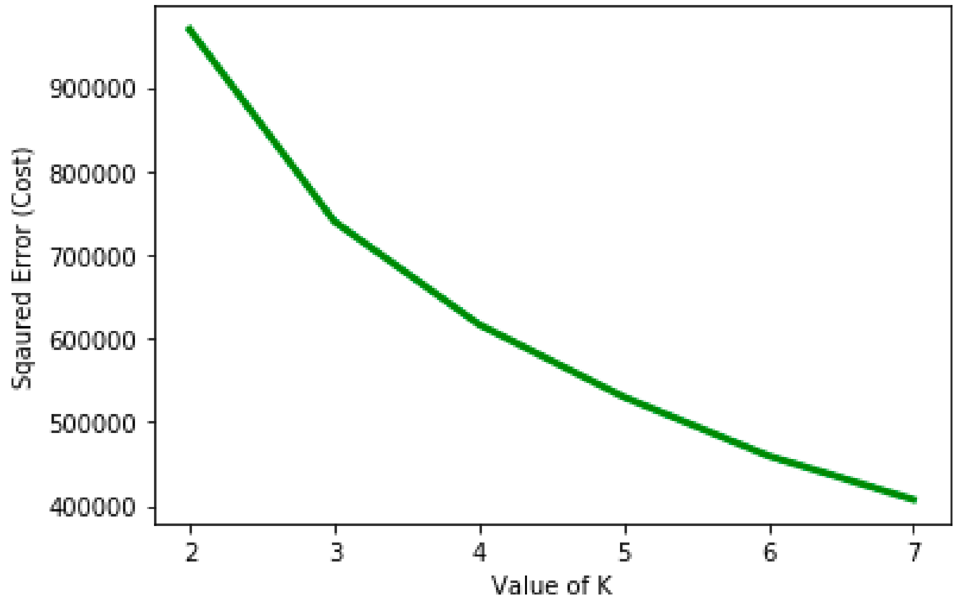

Figure 3: Graph between error against number of clusters for dataset 2

Figure 4 shows scatter plot with 7 clusters for dataset2:

```
Number of custers:
7

<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x1271b07f0>
```

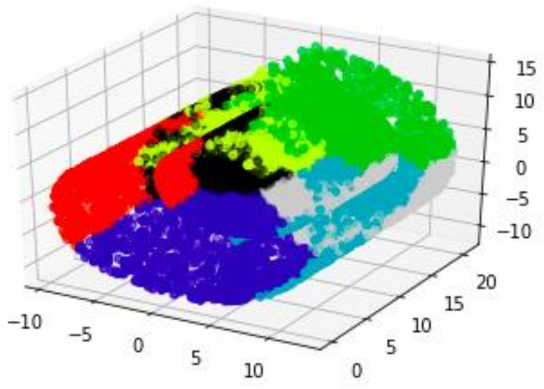

Figure 4: A 3d scatter plot with 7 clusters for Dataset 2

K-Means ++ Initialization

K-Means ++ is an algorithm for choosing the initial values (or "seeds") for the *k*-means clustering algorithm. It is a good way of avoiding the sometimes-poor clustering found by the standard *k*-means algorithm. The number of clusters used was from 2 up to 7. The best number of clusters were seen by plotting a graph of error, which is the squared mean distance between two points, against the number of clusters. The results were very similar to Uniform Random Initialization.

Dataset 1

The kink was found to be when **K was 4**, therefore k=4 will be selected for dataset 1. It was seen that the error decreased rapidly as the value of K was increased from 2 to 4 and then the rate of decay of error slowed down between the K values of 4 to 8 (see fig 5).

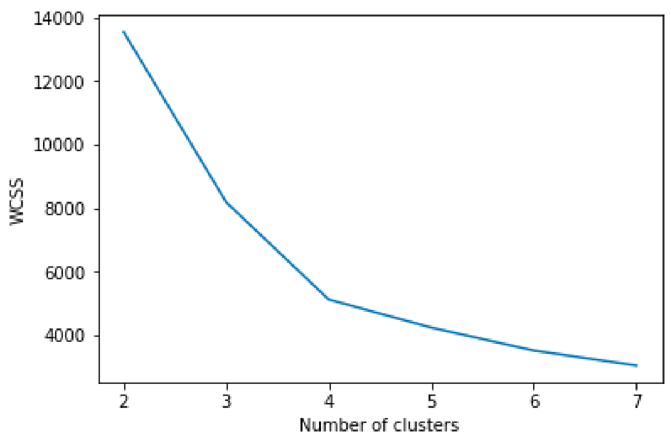

Figure 5: Graph between error against number of clusters for dataset 2

Figure 6 shows scatter plot with 4 clusters for dataset1:

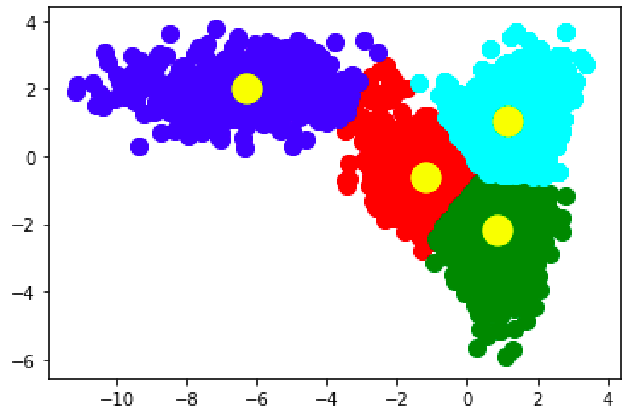

Figure 6: Scatter plot with 4 clusters for Dataset1

Dataset 2

The kink was hard to see for this dataset, it could have been either 3 or 7. I decided to go with 7 clusters, so the kink was found to be when **K was 7**, therefore k=7 will be selected for dataset 2. It was seen that the error decreased rapidly as the value of K was increased from 2 to 7 and then the rate of decay of error slowed down between the K values of 7 to 8 (see fig 7).

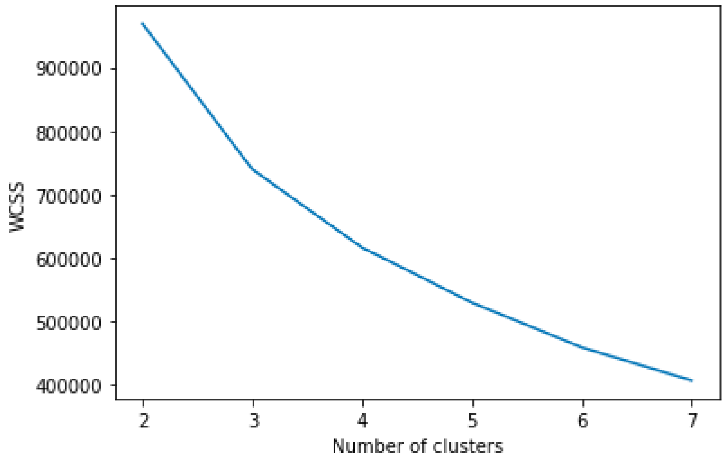

Figure 7: Graph between error against number of clusters for dataset 2

Figure 8 shows scatter plot with 7 clusters for dataset2:

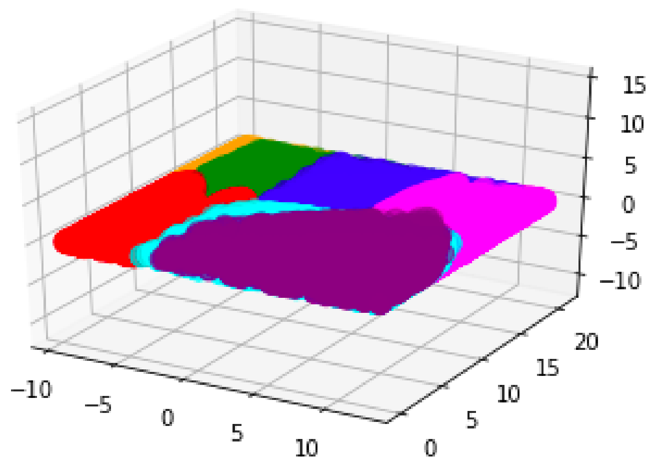

Figure 8: A 3d scatter plot with 7 clusters for Dataset 2

Hierarchical Agglomerative Clustering

The algorithm treats each object as a singleton cluster. Next, pairs of cluster are successively merged until all clusters have been merged into one big cluster containing al objects. The result is a tree-based representation of the objects, named dendrogram. This was done by using both single and average linkage. Where the height of a node indicates the dissimilarity between its two child nodes

Single Linkage

Dataset 1

Figure 9 shows the Dendrogram with single linkage for dataset 1. The cut in this dendrogram was made at 0.5 because it clusters together all the points from 2016 to 3456. After 0.5 we kind of see a rapid increase in the distance (the blue part).

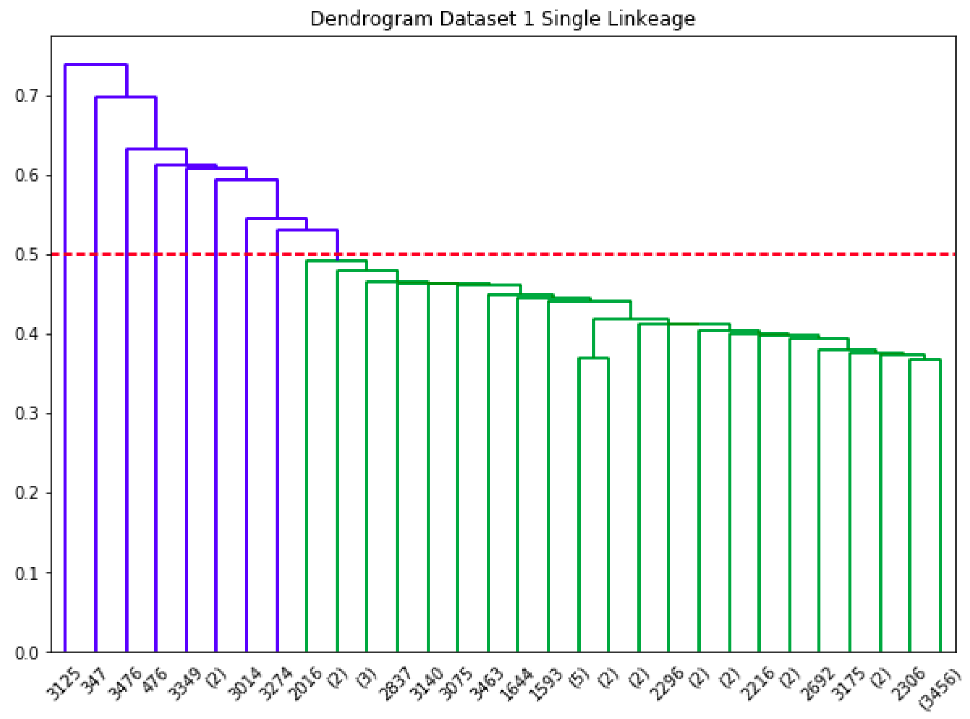

Figure 9: Dendrogram with single linkage for dataset 1

Dataset 2

Figure 10 shows the Dendrogram with single linkage for dataset 2. The cut in this dendrogram was made at 1.0 because it clusters together all the nearby points from 2299 to 6321. After 1.0 we kind of see a rapid increase in the height (the blue part).

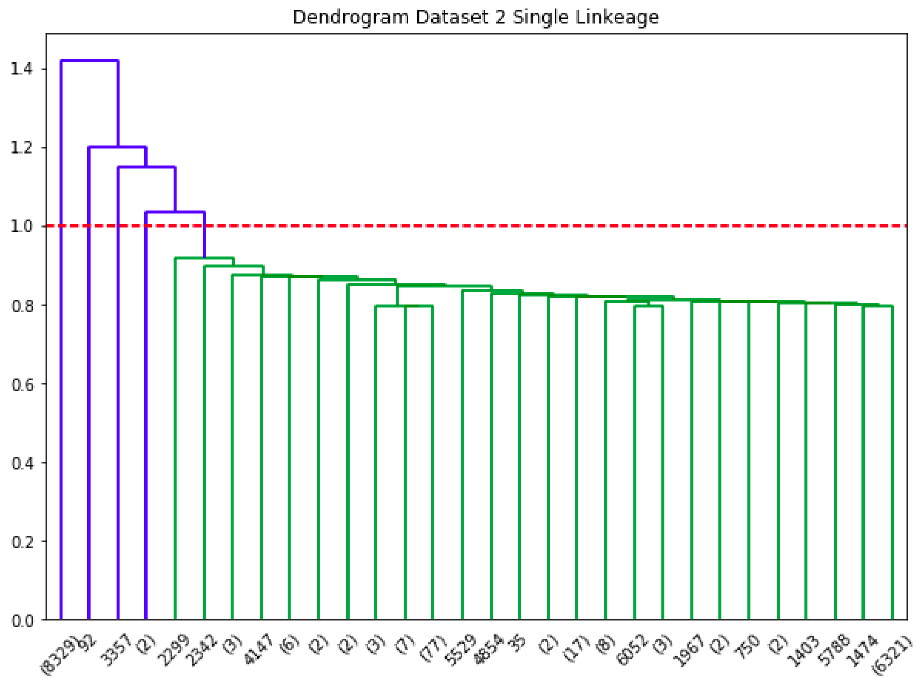

Figure 10: Dendrogram with single linkage for dataset 2

Average Linkage

Dataset 1

Figure 11 shows the Dendrogram with average linkage for dataset 1. The cut in this dendrogram was made at 4 because it clusters together all the nearby points. After 4 we see a rapid increase in the distance.

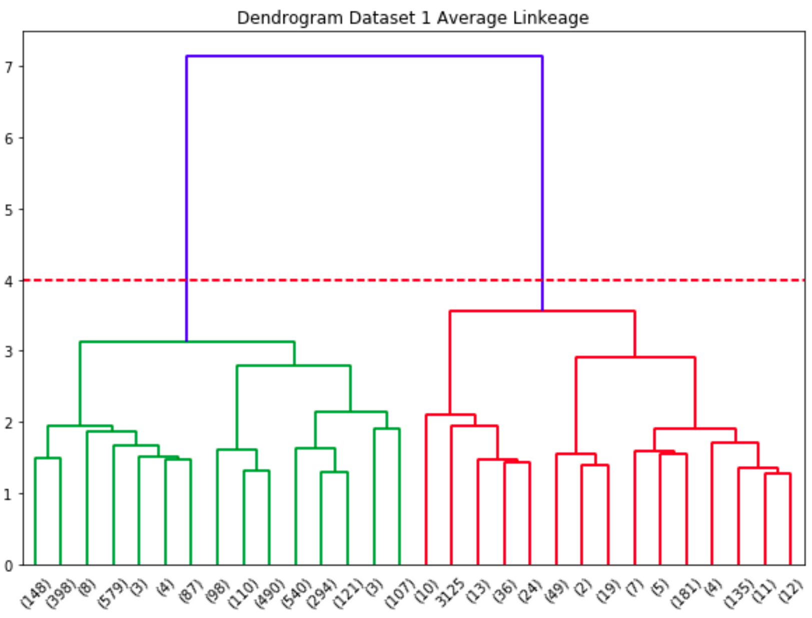

Figure 11: Dendrogram with average linkage for dataset 1

Dataset 2

Figure 12 shows the Dendrogram with average linkage for dataset 2. The cut in this dendrogram was made at 12 because it clusters together all the nearby points. After 12 we see a rapid increase in the height.

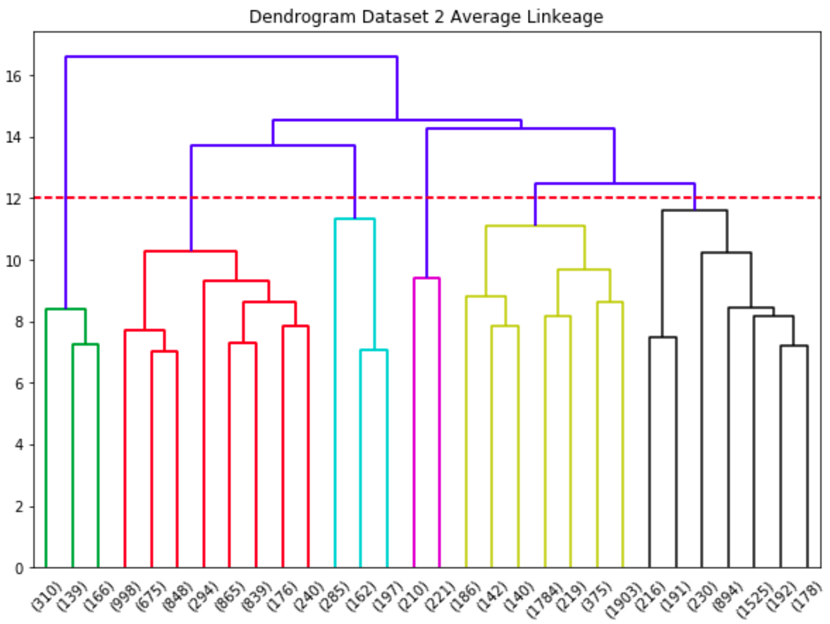

Figure 12: Dendrogram with average linkage for dataset 2