CS57300: Assignment 5
Name: Tunazzina Islam, PUID: 0031294421

1. Exploration:

python exploration.py digits-raw.csv digits-embedding.csv

1.1) Randomly picked one digit from each class in digits-raw.csv and visualize its image as a 28_28 grayscale matrix. I picked 5.

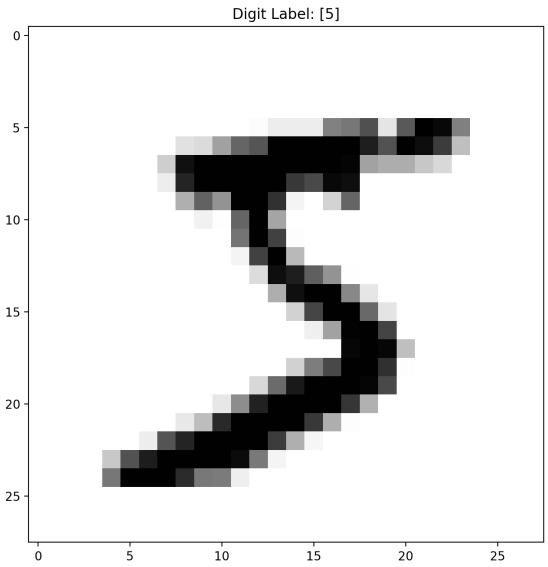

Figure1. Picked digit (5)

1.2) Visualize 1000 randomly selected examples in 2d from the file digits-embedding.csv, coloring the points to show their corresponding class labels.

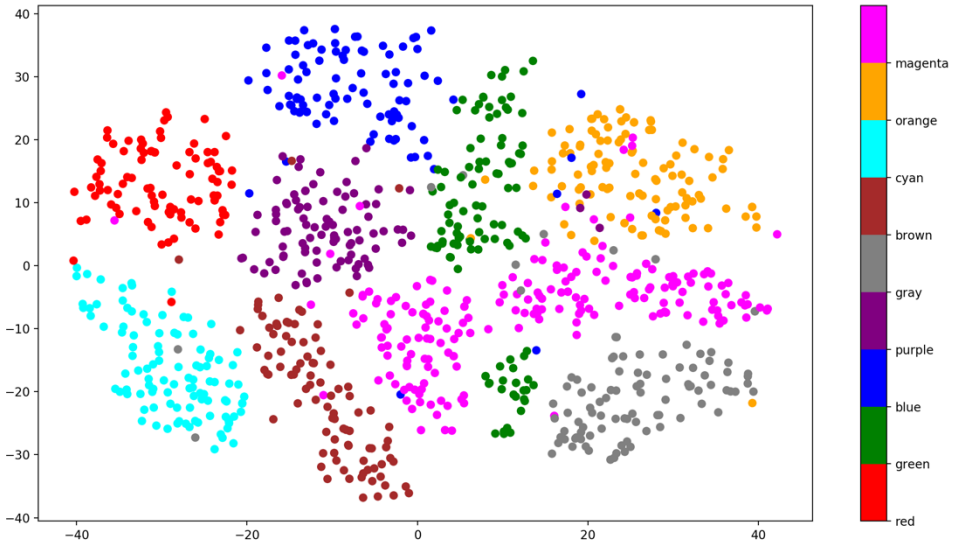

Figure2. Visualization of randomly selected 1000 examples

2. K-means Clustering:

2.1. **Implement K-means:**

python kmeans.py digits-embedding.csv 10

Output:
WC-SSD: 2867066.38
SC: 0.7
N_M_I : 0.5

2.2. **Analysis:**

2.2.1) Cluster the data with different values of K = [2, 4, 8, 16, 32]

Run the code:

python kmeans_2.2.1.py digits-embedding.csv

Plot WC-SSD and SC for dataset 1:

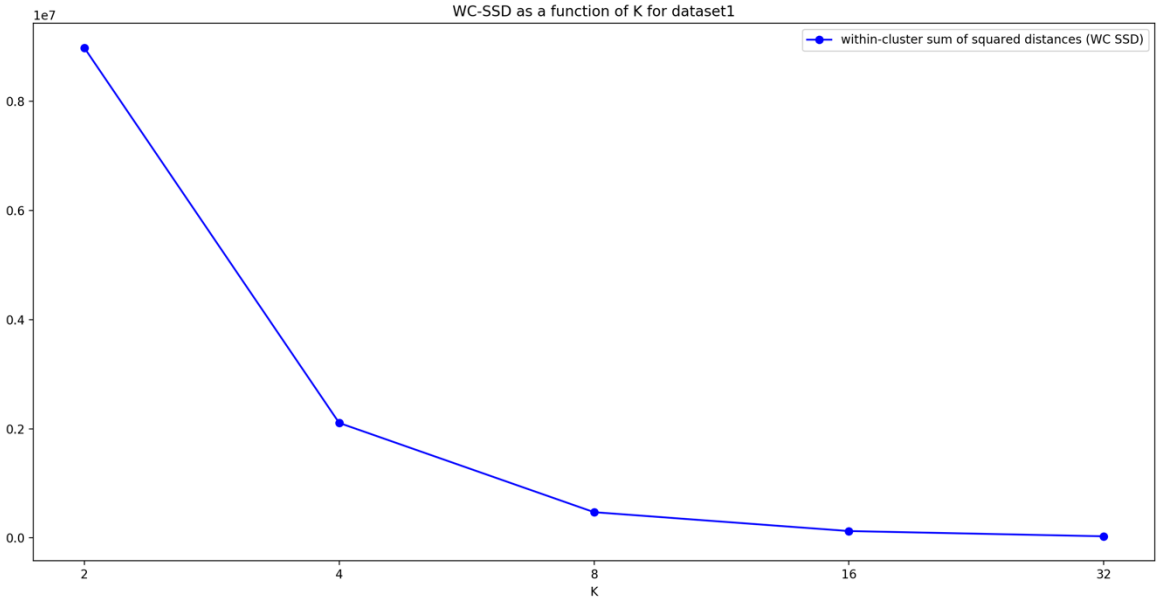

Figure3. Plot WC-SSD Vs K for dataset 1

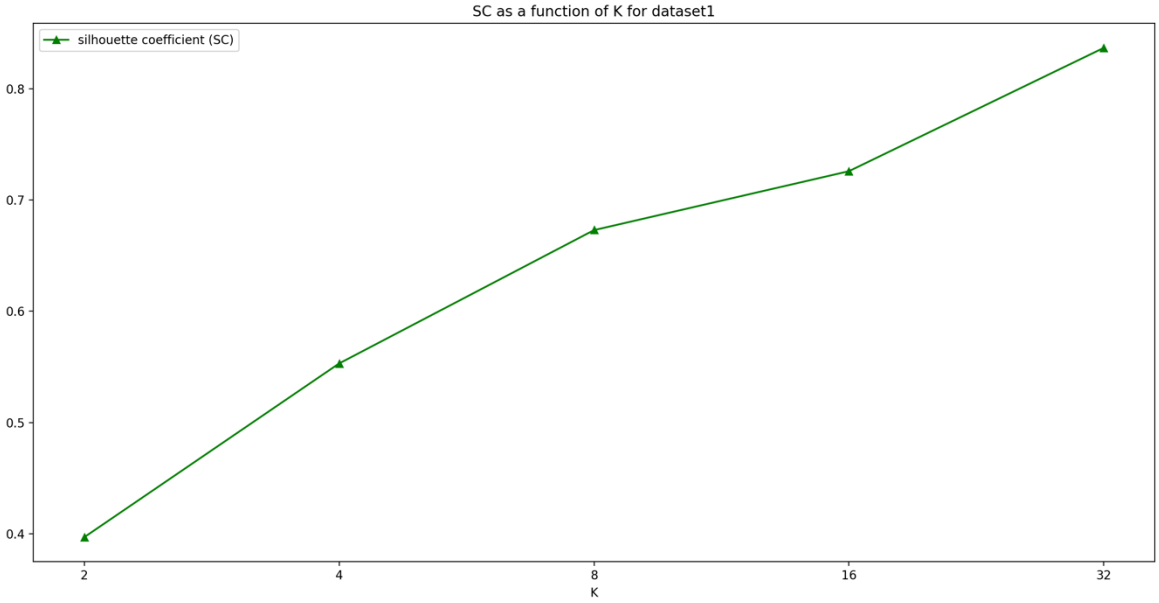

Figure4. Plot SC Vs K for dataset 1

Plot WC-SSD and SC for dataset 2:

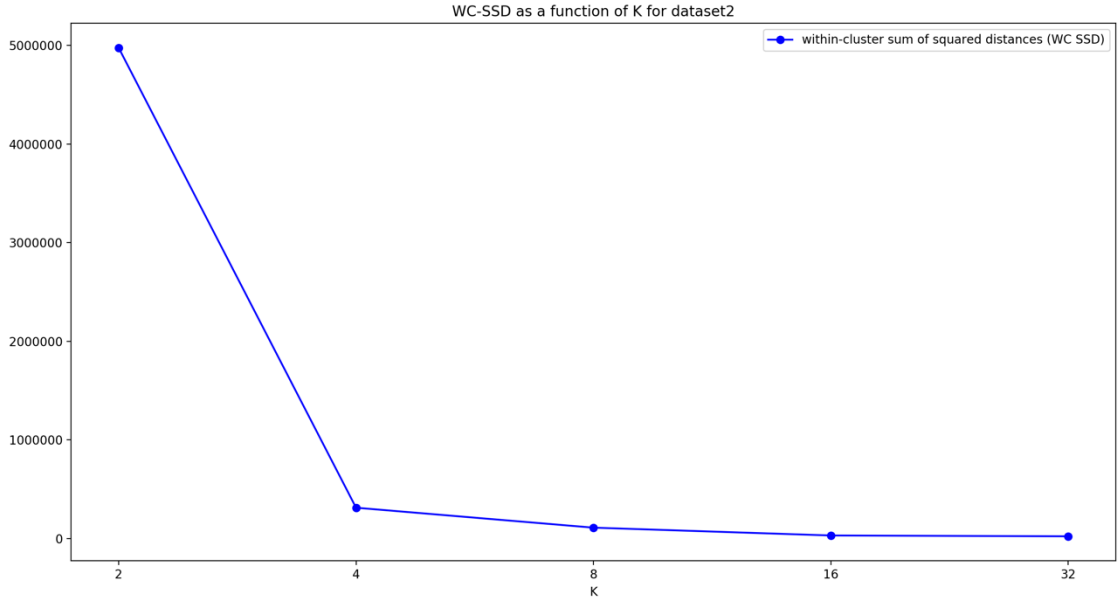

Figure5. Plot WC-SSD Vs K for dataset 2

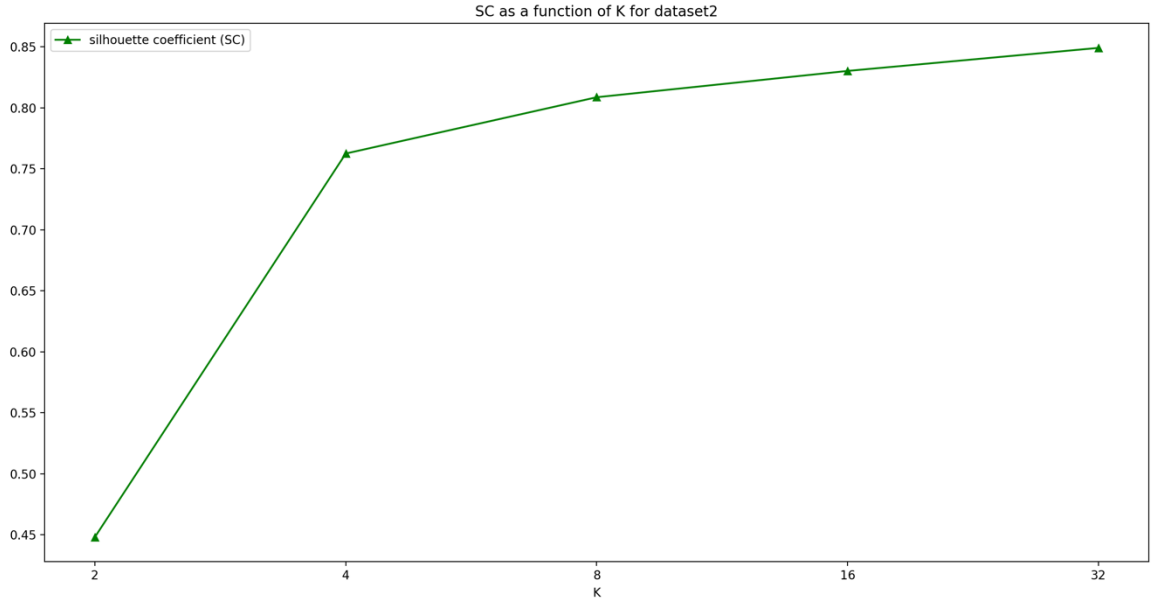

Figure6. Plot SC Vs K for dataset 2

Plot WC-SSD and SC for dataset 3:

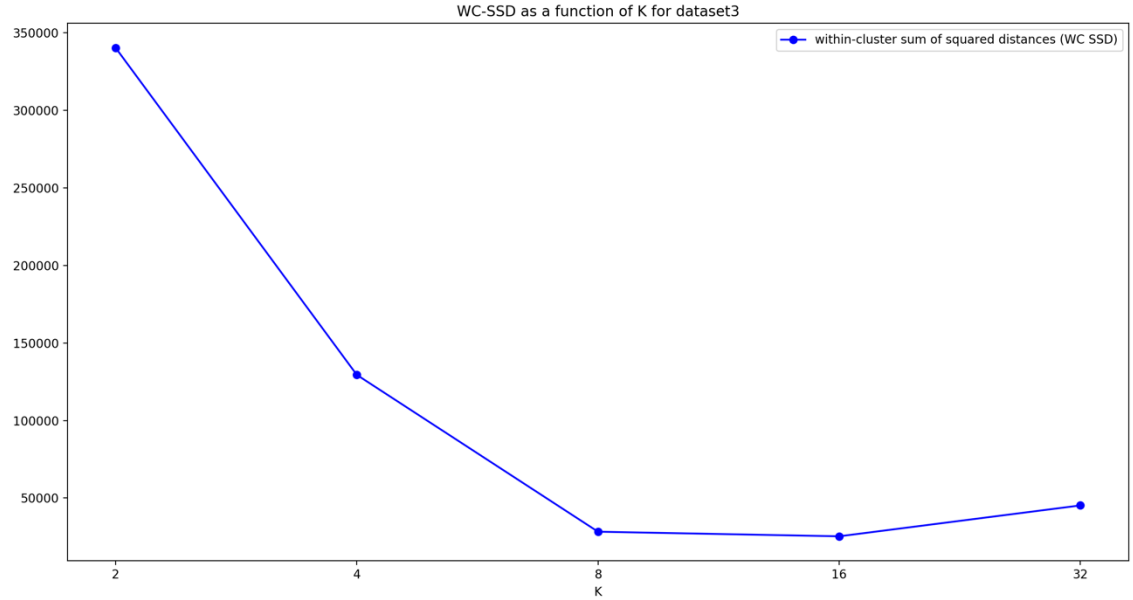

Figure7. Plot WC-SSD Vs K for dataset 3

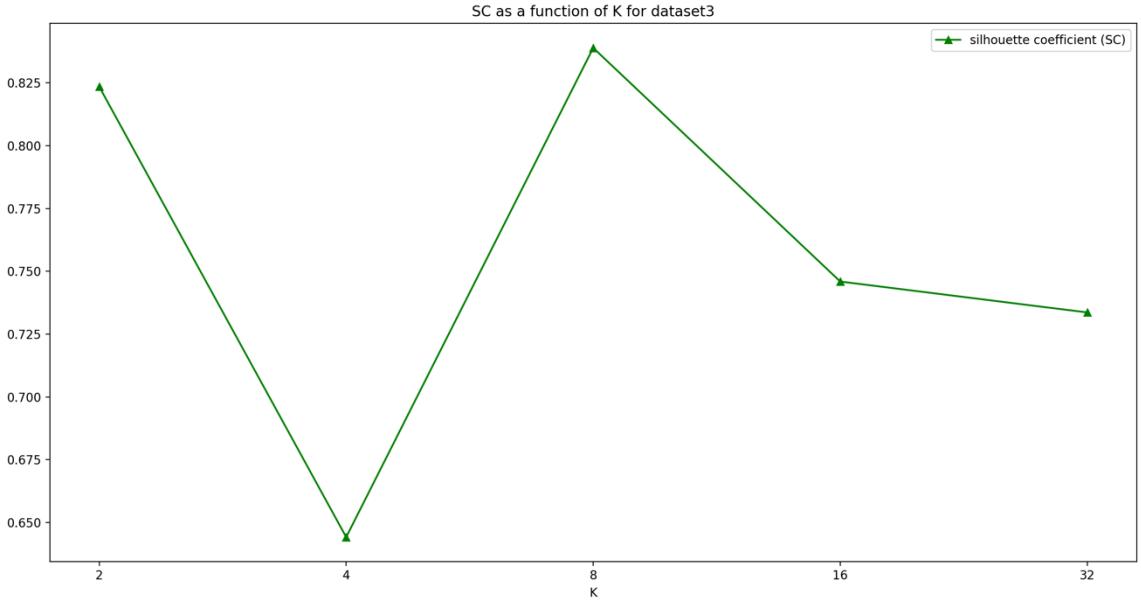

Figure8. Plot SC Vs K for dataset 3

2.2.2) Appropriate K for each dataset:

Explanation:

I clustered the data with different values of K = [2, 4, 8, 16, 32] and I noticed for K = 32 I got several empty clusters. Handling empty clusters is not part of the k-means algorithm but might result in better clusters quality. During the iteration, when I encountered an empty cluster, I places a random data point into that cluster and carried on. So, I think choosing, K = 32 is not a good idea.

I chose value of K for each dataset using Elbow method. According to Elbow method I can also choose K = 4. From the figures we can see, for different value of K (for dataset1 and dataset2) for WC-SSD the number of clusters is chosen at this point, hence the "elbow criterion" (fig.3 and fig.5). For dataset3, the elbow encountered in K = 8 for WC-SSD (fig.7). But I encountered empty cluster for K = 8 in dataset 3 and I preferred to choose K = 4 for dataset 3 too.

For WC-SSD, I observed similar trend for each dataset. For dataset1 and dataset2, I got increasing curve of silhouette coefficient. For dataset3, I received highest SC for K = 8 (fig.8).

2.2.3) Repeat Step 1 with 10 times using 10 different random seeds:
Run the code:
python kmeans_2.2.3.py digits-embedding.csv
Plot Mean and Standard deviation of WC-SSD and SC for dataset 1:

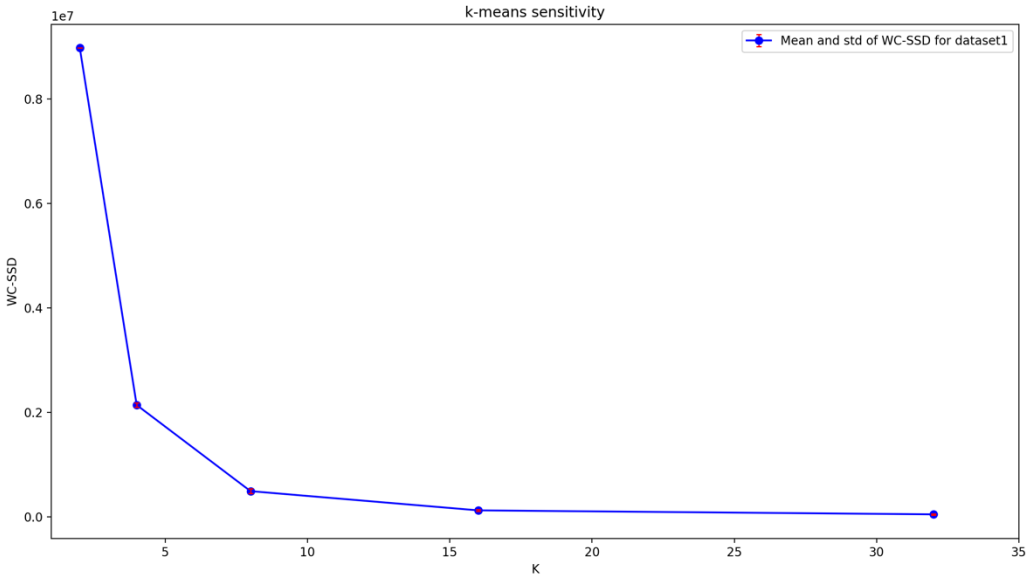

Figure9: Mean and Standard deviation of WC-SSD vs K for dataset 1

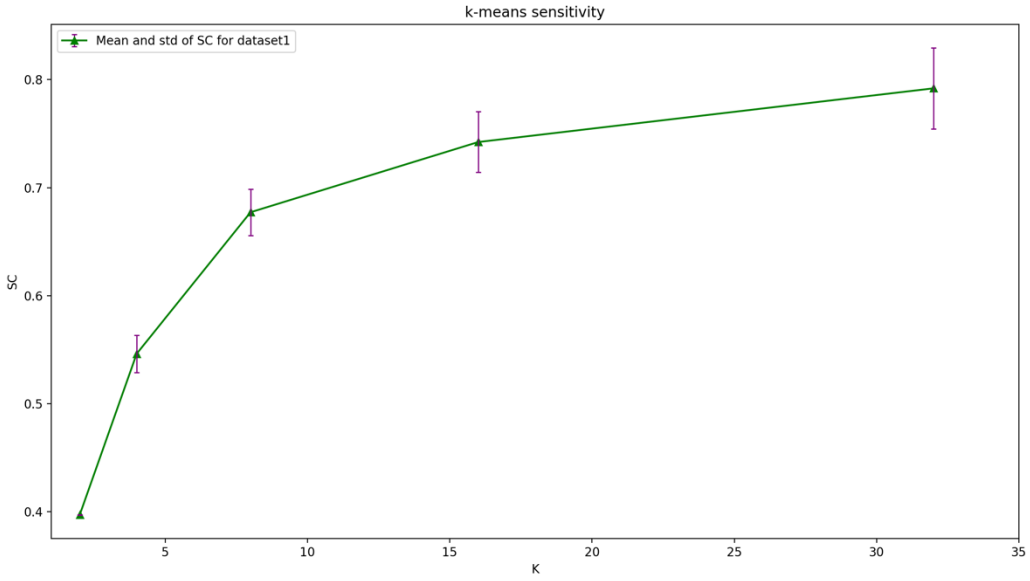

Figure10: Mean and Standard deviation of SC vs K for dataset 1

Plot Mean and Standard deviation of WC-SSD and SC for dataset 2:

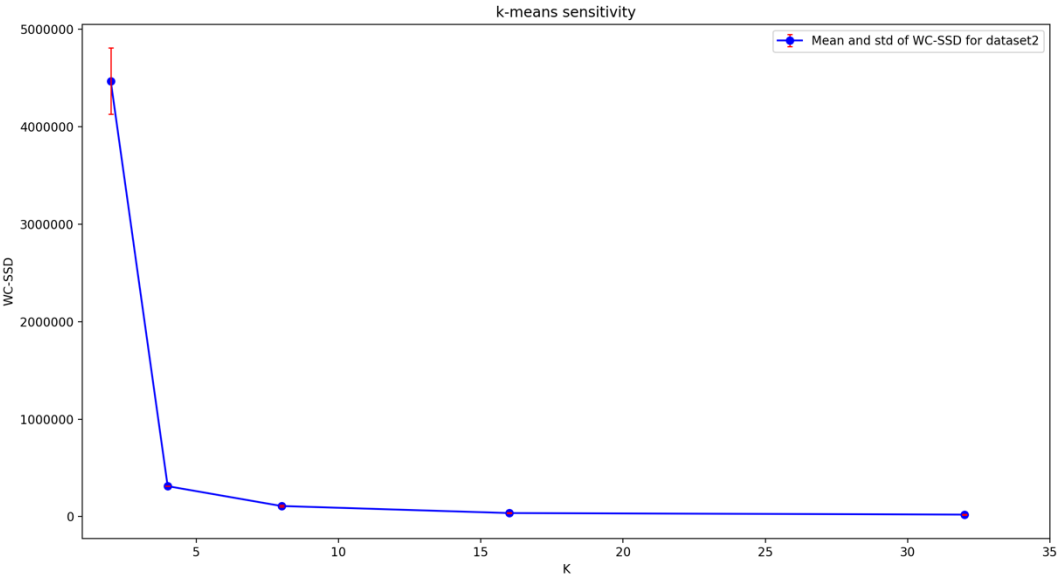

Figure11: Mean and Standard deviation of WC-SSD vs K for dataset 2

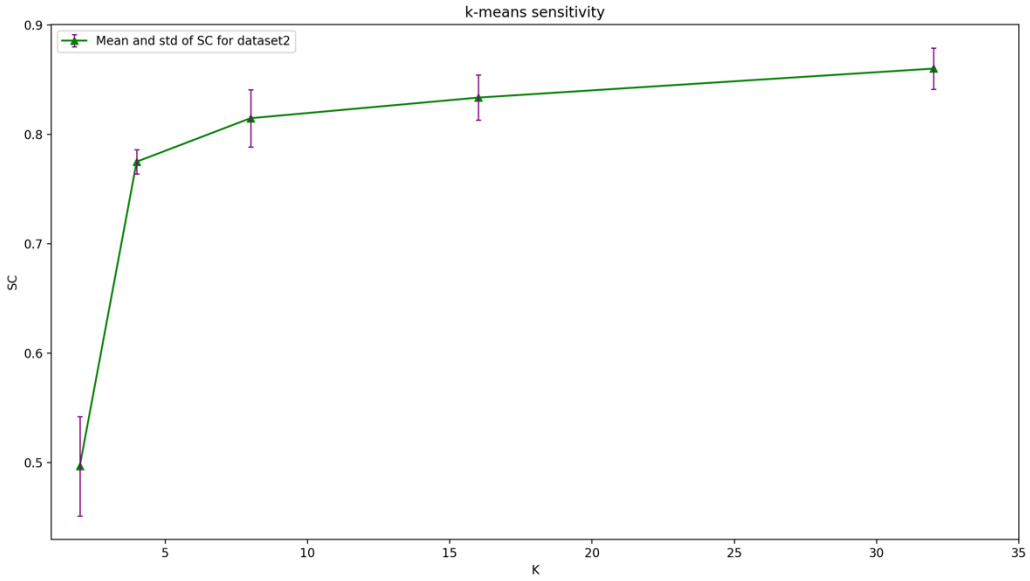

Figure12: Mean and Standard deviation of SC vs K for dataset 2

Plot Mean and Standard deviation of WC-SSD and SC for dataset 3:

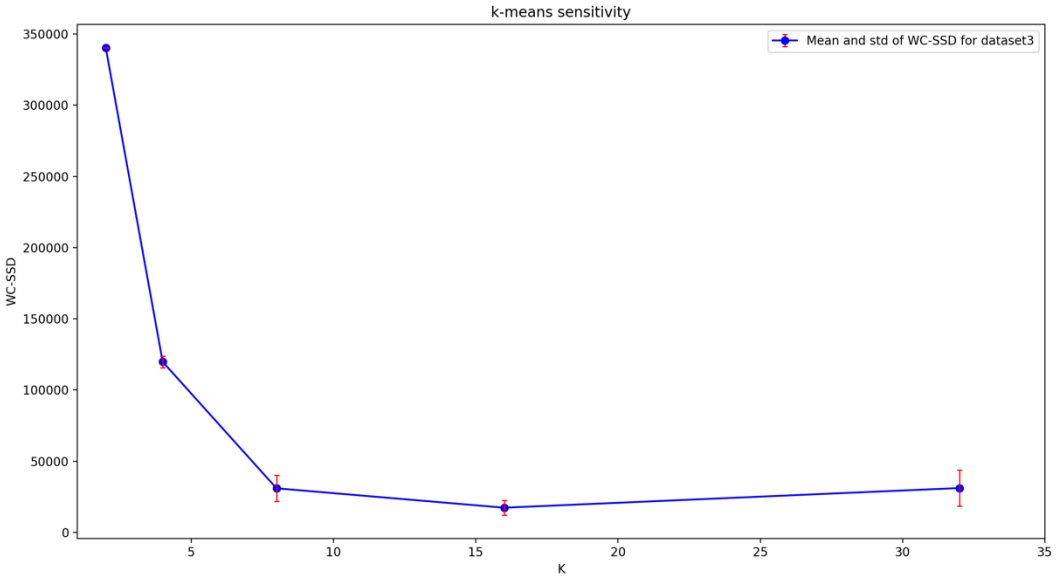

Figure13: Mean and Standard deviation of WC-SSD vs K for dataset 3

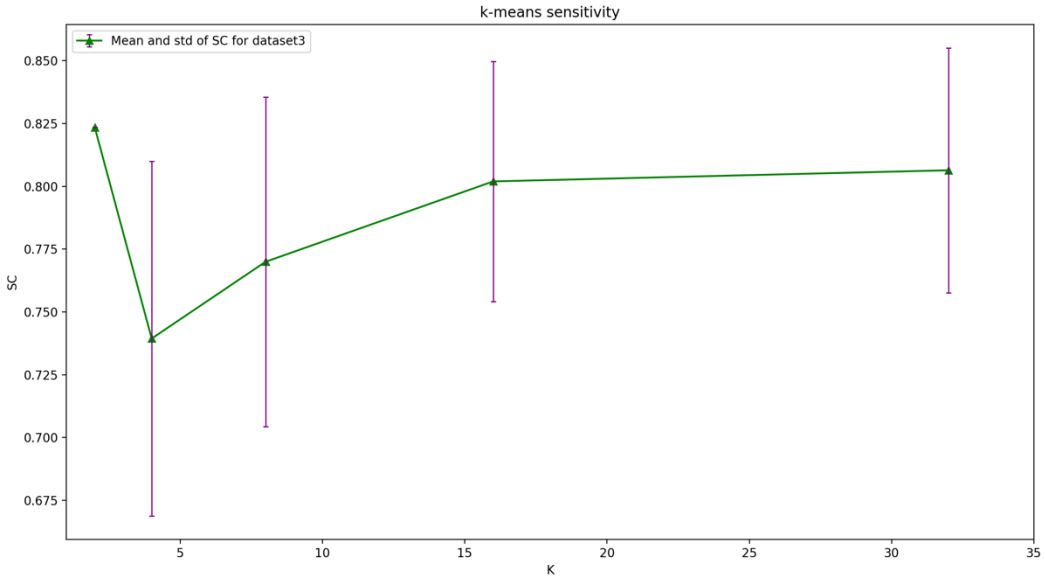

Figure14: Mean and Standard deviation of SC vs K for dataset 3

For dataset1 and dataset2, I got elbow for K = 4 with negligible standard error. For dataset3, though I observed elbow for K = 8, I would prefer to choose K = 4 because it has very minimal standard error. In case of silhouette coefficient, dataset1 and dataset2 have increasing curve with minimal error. But SC of dataset3 has higher standard error. The k-means algorithm updates the cluster centers by taking the average of all the data points that are closer to each cluster center. When all the points are packed nicely together (dataset1 and dataset2), the average makes sense. However, when we have outliers, this can affect the average calculation of the whole cluster. As a result, this will push the cluster center closer to

the outlier. We are running the k-means algorithm 10 times using 10 different random seeds, dataset3 has sparse data and it became more sensitive than others.

2.2.4) NMI calculation:

Run the code:
python kmeans_2.2.4.py digits-embedding.csv
Output:
N_M_I for dataset 1: 0.63
N_M_I for dataset 2: 0.5
N_M_I for dataset 3: 0.45

Visualization:
Instead of 1000 randomly selected examples, I took all examples of each dataset to visualize.
Here, Black pentagram (☆) represents the centroid of each cluster.

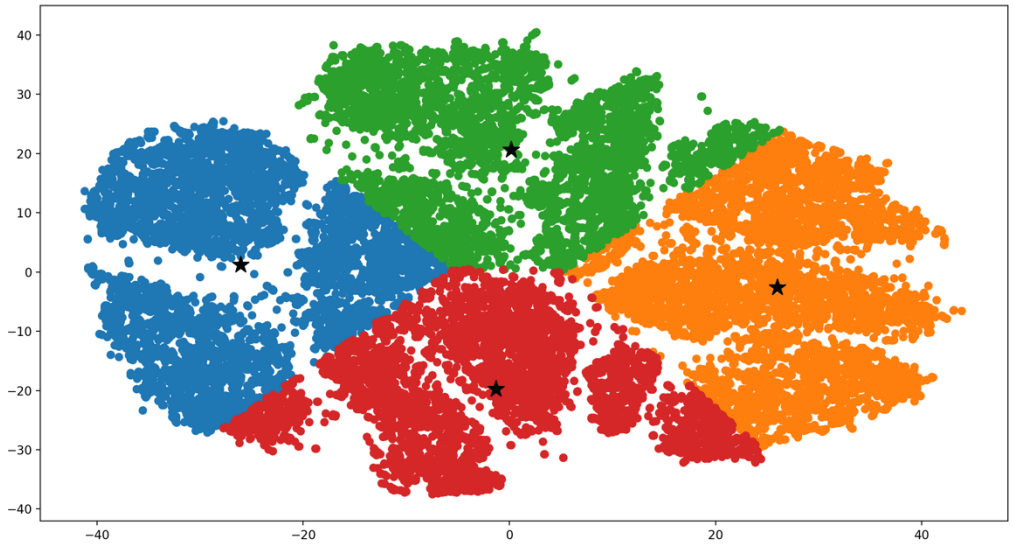

Figure15. Visualization for dataset 1

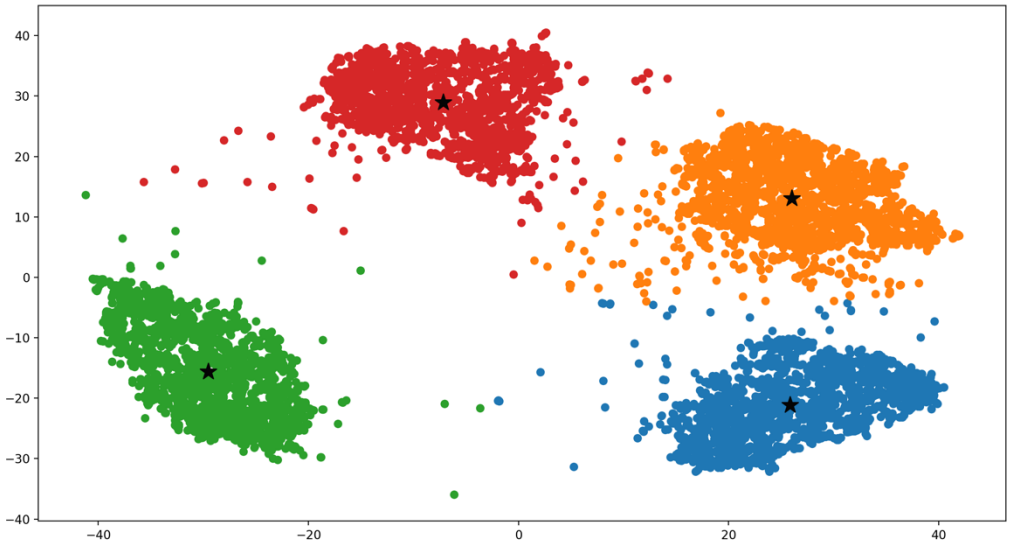

Figure16. Visualization for dataset 2

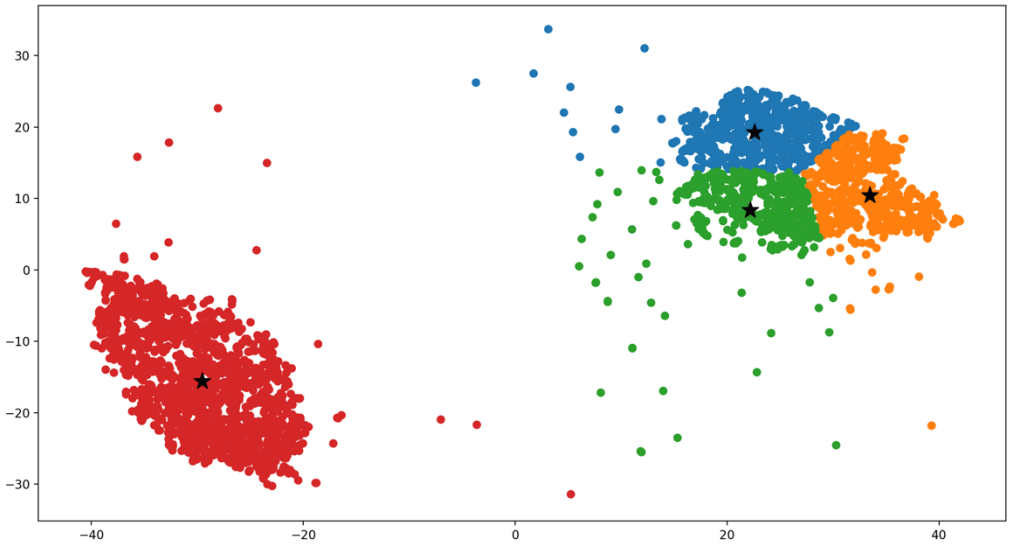

Figure17. Visualization for dataset 3

In fig.15, we can see dataset1 is clustered well in 4 clusters (K = 4). For dataset 2 and dataset 3, I observed outliers. NMI of dataset1 is larger than others. If NMI for a clustering is higher than the other clustering, it means we would prefer that clustering.

3. Hierarchical Clustering:

3.1) And

3.2) Run the code:

python hierarchical_3.1and3.2.py digits-embedding.csv

Plot dendrogram for single linkage:

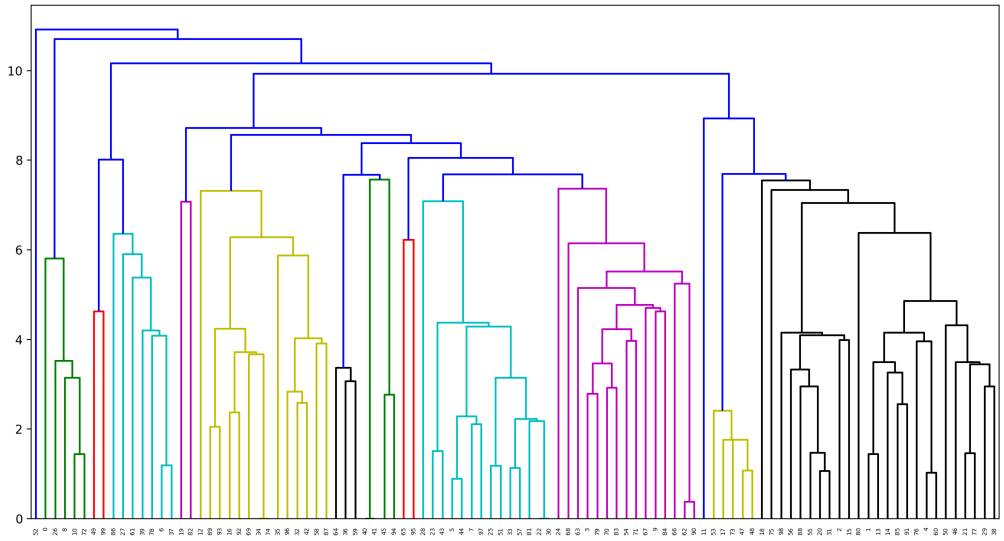

Figure18: Dendrogram for single linkage

Plot dendrogram for complete linkage:

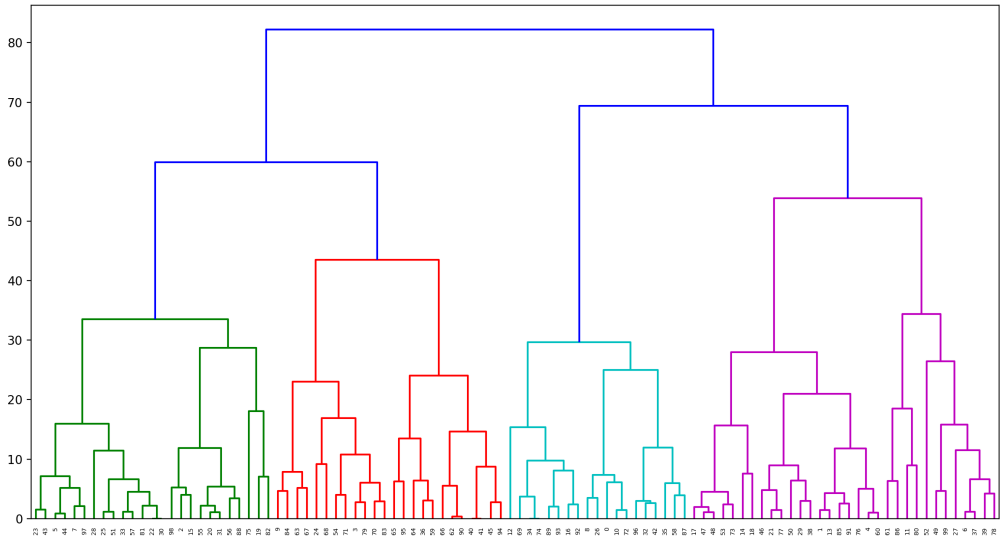

Figure19: Dendrogram for complete linkage

Plot dendrogram for average linkage:

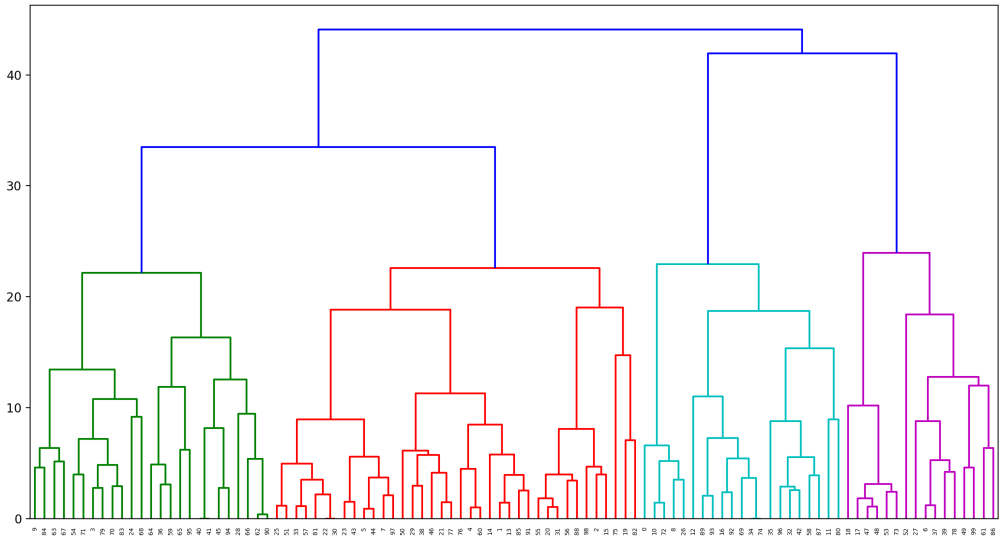

Figure20: Dendrogram for average linkage

3.3) I cut each of the dendrograms at successive levels of the hierarchy to produce partitions of different sizes K = [2, 4, 6, 8].
I showed the cut for K = 4 and K= 8 (for each of single, complete, and average linkage) in 3 different figures (fig.21, fig.22, and fig.23) and I wrote the code in separate file named "hierarchical_3.3_successive_cut.py".

Run the code:
python hierarchical_3.3_successive_cut.py digits-embedding.csv

This will show following figures:

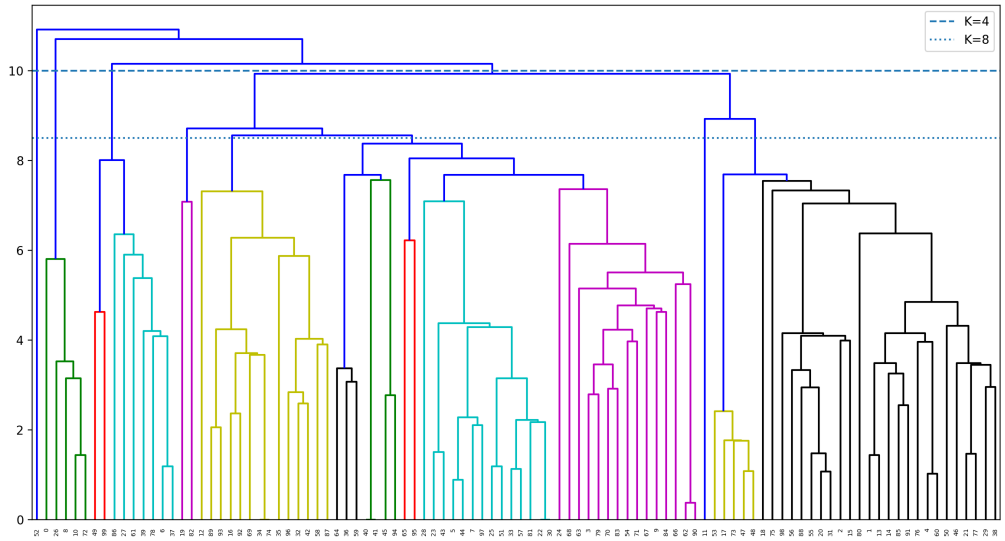

Figure21: Dendrogram cut for single linkage

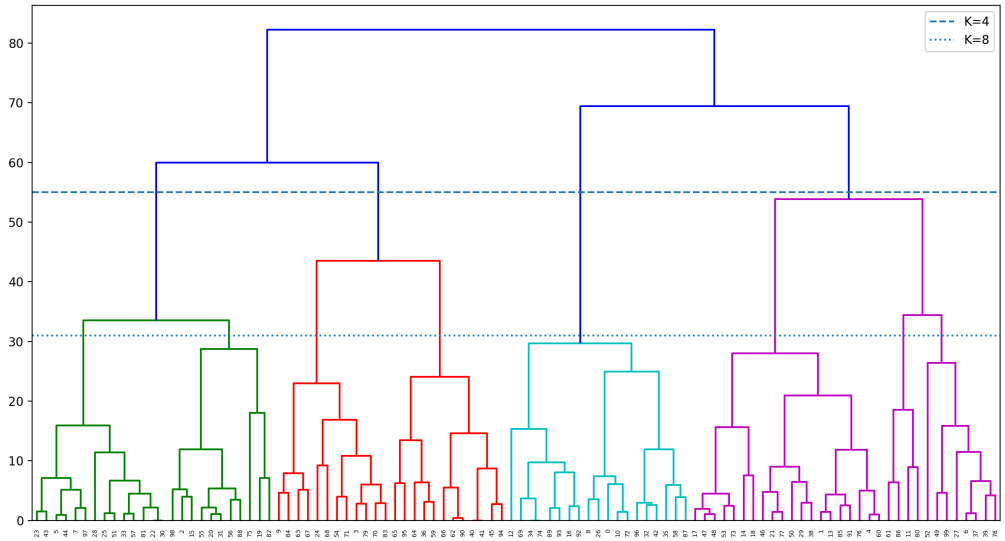

Figure22: Dendrogram cut for complete linkage

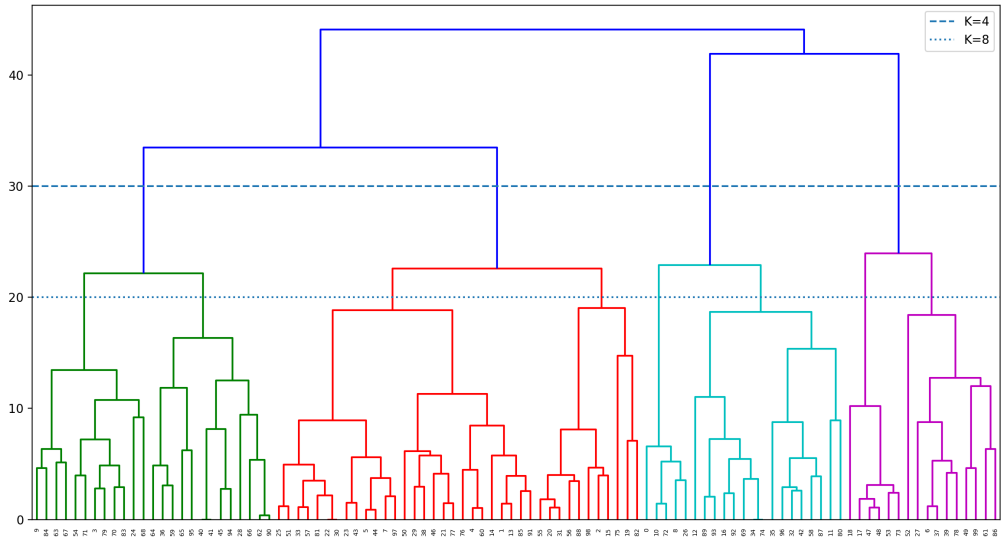

Figure23: Dendrogram cut for average linkage

I calculated within-cluster sum of squared distances (WC-SSD) and silhouette coefficient (SC) as a function of K.

Run the code:

python hierarchical_3.3.py digits-embedding.csv

Plot WC-SSD and SC as a function of K for single linkage:

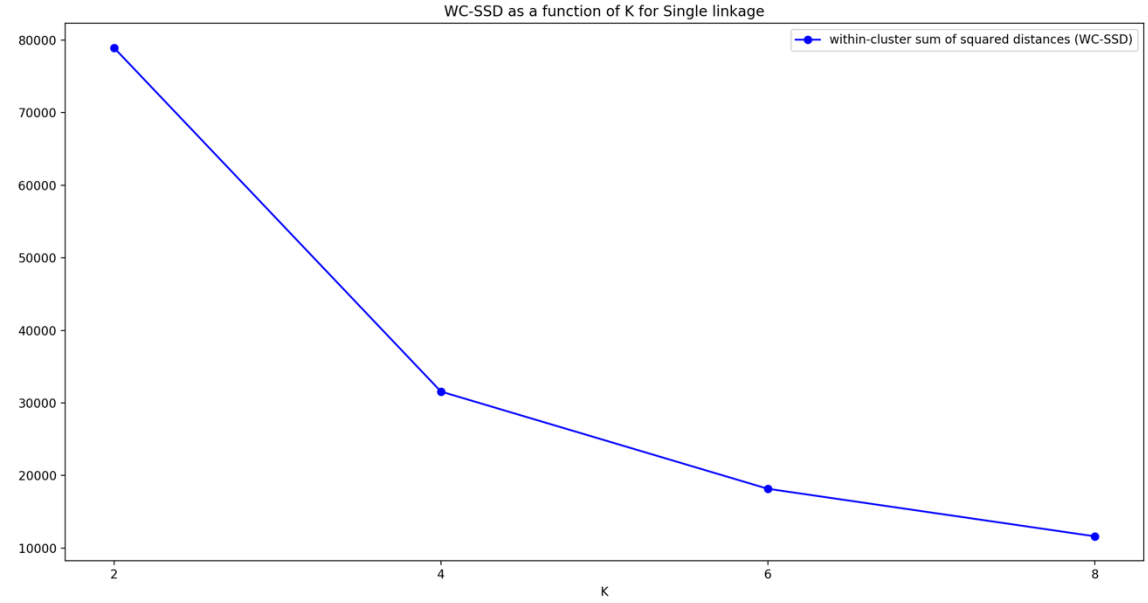

Figure24: WC-SSD vs K for single linkage

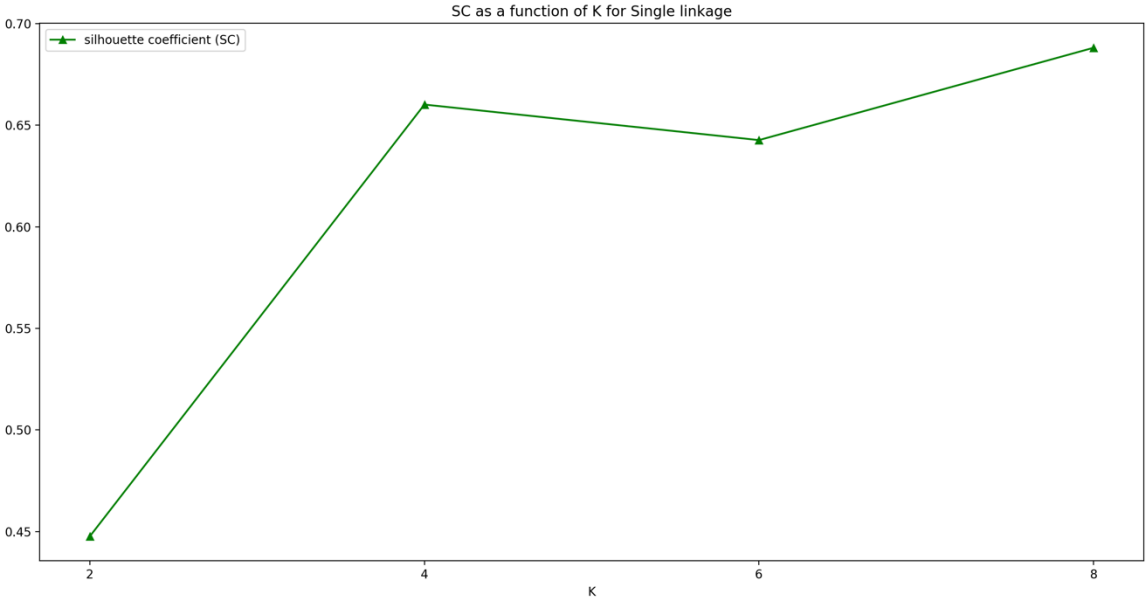

Figure25: SC vs K for single linkage

Plot WC-SSD and SC as a function of K for complete linkage:

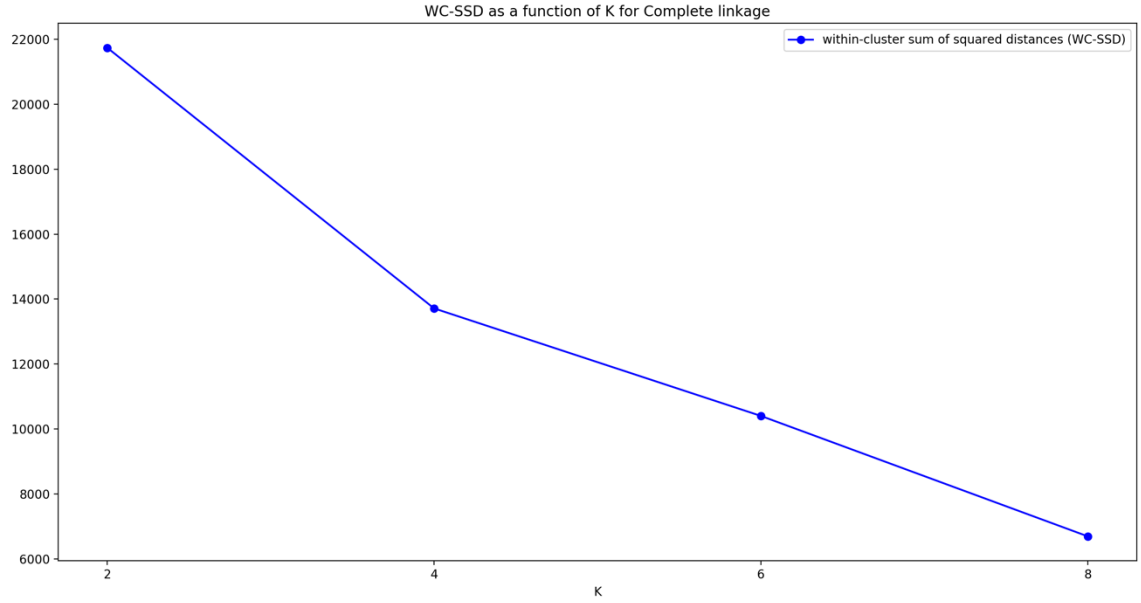

Figure26: WC-SSD vs K for complete linkage

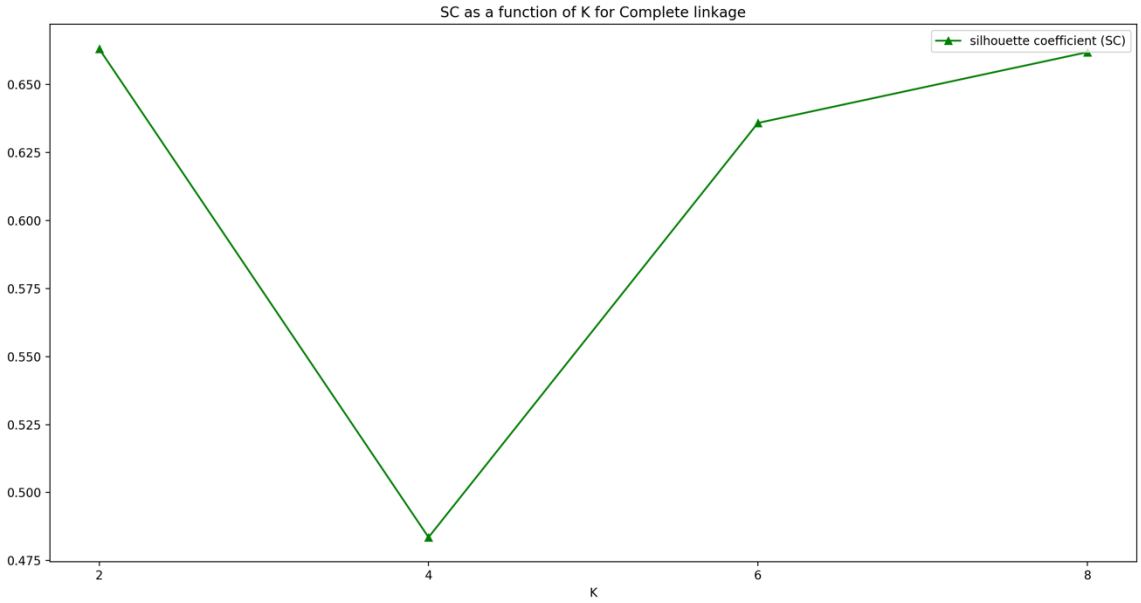

Figure27: SC vs K for complete linkage

Plot WC-SSD and SC as a function of K for average linkage:

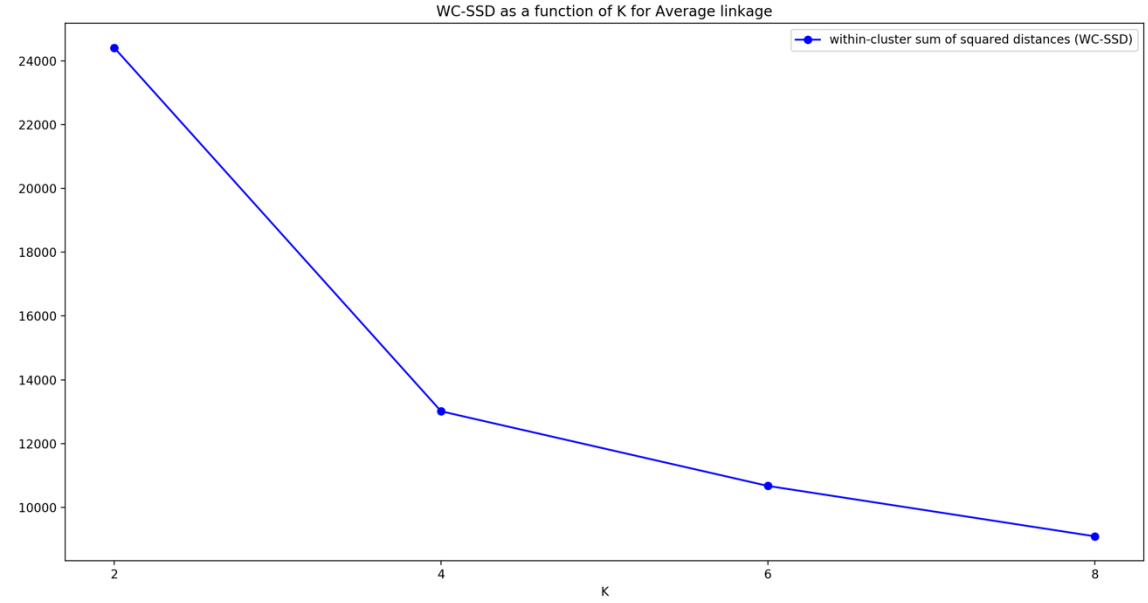

Figure28: WC-SSD vs K for average linkage

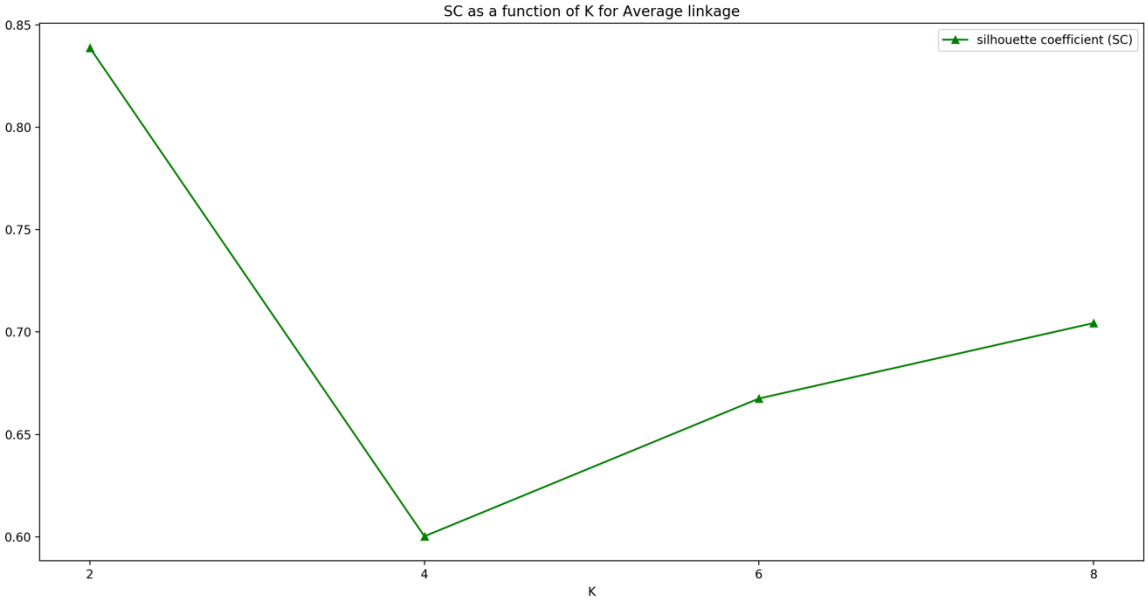

Figure29: SC vs K for average linkage

3.4) Choose value of K:

In K means clustering, we have to define the number of clusters to be created beforehand, which is sometimes difficult to say. Whereas in Hierarchical clustering data is automatically formed into a tree shape form (dendrogram) and we can choose which trees are significant. I cut each of the dendrograms at successive levels of the hierarchy to produce partitions of different sizes K. From the figures (fig.21, fig.22, and fig.23) of cut dendrogram cut we can easily choose K. I chose K = 4.

According to Elbow method I can also choose K = 4. From the figures (fig.24, fig.26, and, fig.28) we can see, for different value of K (for each of single, complete, and average linkage) for WC-SSD the number of clusters is chosen at this point, hence the "elbow criterion".

For K-means I chose K = 4 for Dataset 1 in Section 2. In K-means clustering, since we start with random choice of clusters, the results produced by running the algorithm multiple times might differ. While results are reproducible in Hierarchical clustering.

3.5) For choice of K = 4:

Run the code:
python hierarchical_3.5.py digits-embedding.csv

Output:
N_M_I for Single Linkage: 0.83
N_M_I for Complete Linkage: 0.49
N_M_I for Average Linkage: 0.6

Normalized Mutual Information (NMI) is a normalization of the Mutual Information (MI) score to scale the results between 0 (no mutual information) and 1 (perfect correlation).

In single-link (or single linkage) hierarchical clustering, we merge in each step the two clusters whose two closest members have the smallest distance (or: the two clusters with the smallest minimum pairwise distance). We will get higher NMI for single linkage.
In complete-link (or complete linkage) hierarchical clustering, we merge in each step the two clusters whose merger has the smallest diameter (or: the two clusters with the smallest maximum pairwise distance). We will get smaller NMI for complete linkage.
Average linkage measures the average dissimilarity over all pairs. It is a compromise between the sensitivity of complete-link clustering to outliers and the tendency of single-link clustering to form long chains that do not correspond to the intuitive notion of clusters as compact, spherical objects. We are expecting to get NMI in between single linkage and complete linkage.

Results from K-means for Dataset 1 in Section 2, I got NMI 0.63 for dataset 1 which is similar to the NMI (0.6) that I got from average linkage of agglomerative clustering. In both algorithms, I chose K = 4.