CNG 409 Second Assignment

Talal Shafei

2542371

Part 1:

Hyperparameters to test for KNN:

- similarity function: Cosine, Minkowski, and Mahalanobis
- K: 5, 10, and 30

Note: for Cosine my function returns the angle, so the bigger the cosine the more similar two vectors, and the smaller the angle, I did it this way because I want my code to expect small distance value for similar vectors.

Total configurations: 9

Index	Distance	K value	Accuracy (%)	95% Confidence Interval
1	Cosine	5	93.867	[93.429, 94.304]
2	Cosine	10	94.533	[93.961, 95.106]
3	Cosine	30	94.400	[94.114, 94.686]
4	Minkowski	5	94.933	[94.647, 95.220]
5	Minkowski	10	95.067	[94.599, 95.534]
6	Minkowski	30	93.467	[93.029, 93.904]
7	Mahalanobis	5	90.400	[89.607, 91.193]
8	Mahalanobis	10	88.667	[87.621, 89.712]
9	Mahalanobis	30	82.933	[82.647, 83.220]

Best Hyperparameter corresponds to the hyperparameters index 5:

Distance function: Minkowski, and K value: 10

Mean accuracy: 95.067%

Configuration was chosen based on the highest accuracy score

Part 2:

First displaying tables of statistics obtained for datasets using Kmeans and Kmeans ++ then displaying graphs for K vs. Loss.

Kmeans:

For Dataset1:

K value	Loss	95% Confidence Interval
2	157.994	[157.994, 157.994]
3	73.475	[73.475, 73.475]
4	33.956	[33.956, 33.956]
5	10.893	[10.893, 10.893]
6	10.041	[9.993, 10.088]
7	9.262	[9.254, 9.270]
8	8.584	[8.531, 8.636]
9	7.885	[7.829, 7.941]
10	7.278	[7.231, 7.325]

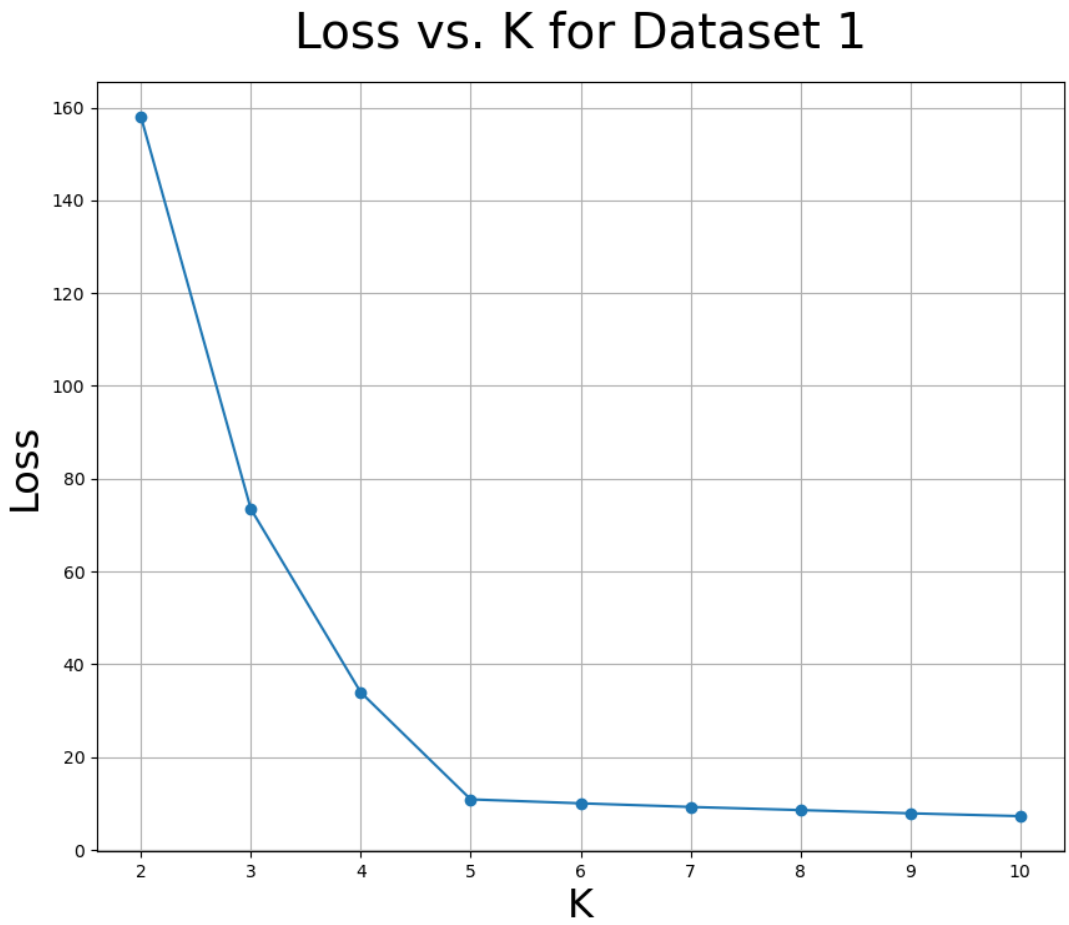
Notice that after cluster 5 the loss decreases slowly.

For Dataset2:

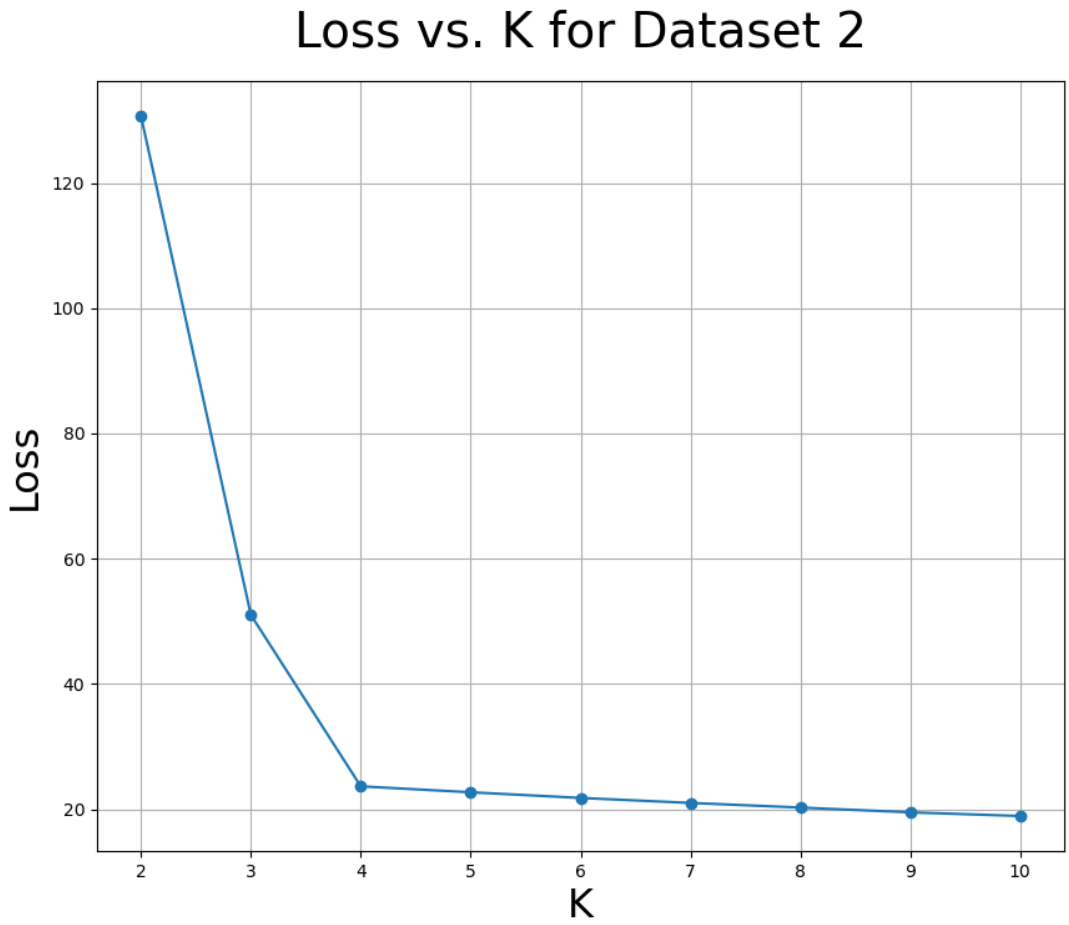
K value	Loss	95% Confidence Interval
2	130.692	[130.692, 130.692]
3	51.026	[51.026, 51.026]
4	23.652	[23.652, 23.652]
5	22.702	[22.696, 22.708]
6	21.799	[21.774, 21.824]
7	21.002	[20.947, 21.058]
8	20.262	[20.200, 20.324]
9	19.495	[19.466, 19.524]
10	18.915	[18.865, 18.965]

Notice that after cluster 4 the loss decreases slowly.

Graphs:



By the Elbow method we can see that the elbow is at cluster 5, in other words the loss function after cluster 5 starts decreasing linearly without sudden falls, in that case the best cluster for Dataset1 is 5.



By the Elbow method we can see that the elbow is at cluster 4, in other words the loss function after cluster 4 starts decreasing linearly without sudden falls, in that case the best cluster for Dataset2 is 4.

Kmeans++:

For Dataset1:

K value	Loss	95% Confidence Interval
2	157.994	[157.994, 157.994]
3	73.475	[73.475, 73.475]
4	33.956	[33.956, 33.956]
5	10.893	[10.893, 10.893]
6	9.991	[9.990, 9.991]
7	9.262	[9.253, 9.271]
8	8.542	[8.525, 8.559]
9	7.823	[7.796, 7.849]
10	7.169	[7.118, 7.221]

Notice that after cluster 5 the loss decreases slowly.

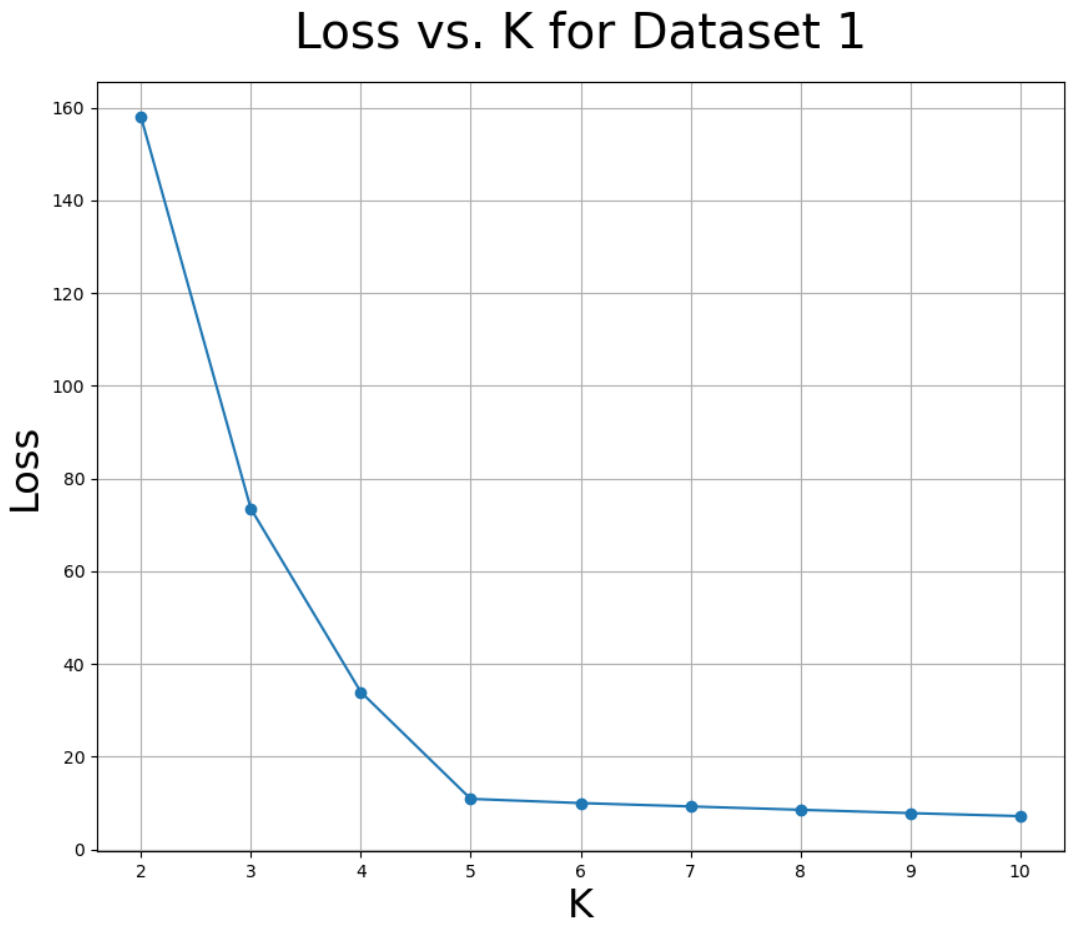
For Dataset2:

K value	Loss	95% Confidence Interval
2	130.692	[130.692, 130.692]
3	51.026	[51.026, 51.026]
4	23.652	[23.652, 23.652]
5	22.703	[22.695, 22.710]
6	21.807	[21.792, 21.822]
7	20.906	[20.883, 20.929]
8	20.113	[20.035, 20.191]
9	19.466	[19.422, 19.511]
10	18.842	[18.788, 18.897]

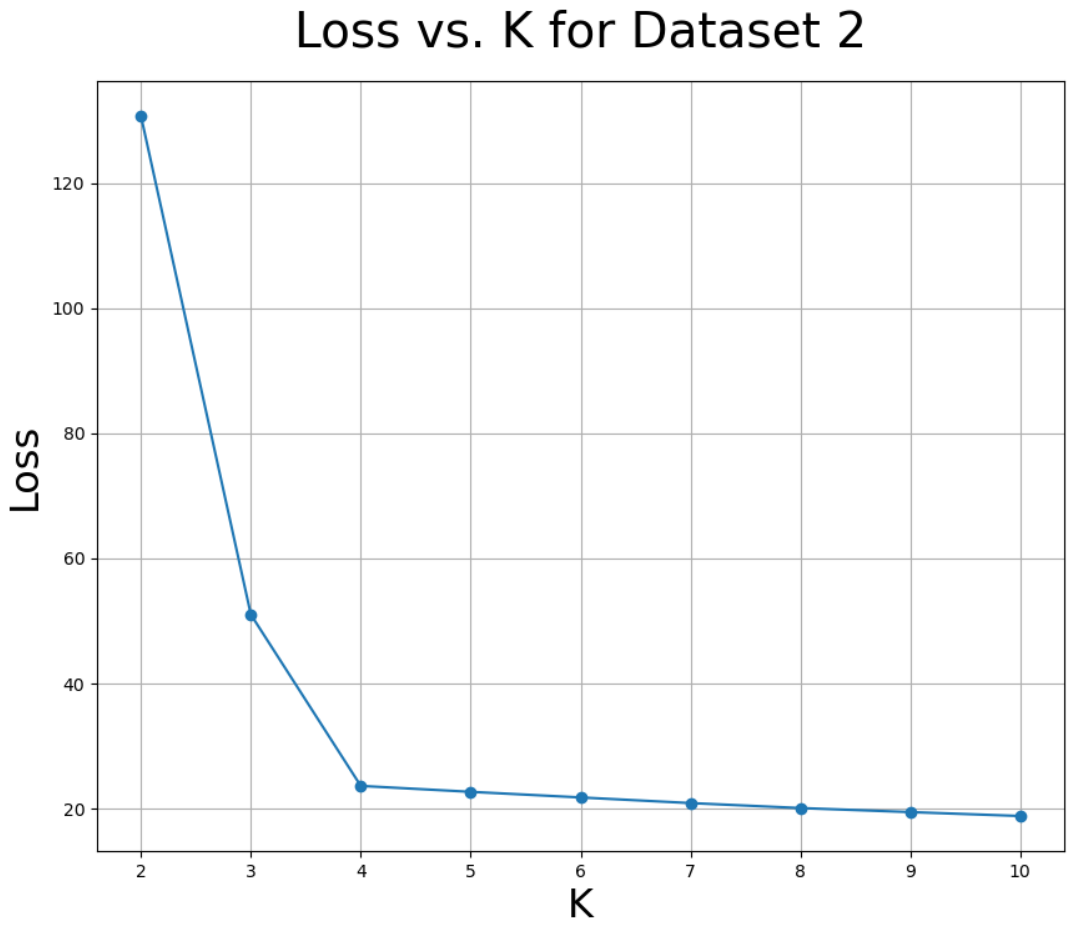
Notice that after cluster 4 the loss decreases slowly.

Notice that most of the Loss values for Kmeans++ when K is high was less than the Loss values of the Kmeans, and that due to the better initialized centroids.

Graphs:



By the Elbow method we can see that the elbow is at cluster 5, in other words the loss function after cluster 5 starts decreasing linearly without sudden falls, in that case the best cluster for Dataset1 is 5.



By the Elbow method we can see that the elbow is at cluster 4, in other words the loss function after cluster 4 starts decreasing linearly without sudden falls, in that case the best cluster for Dataset2 is 4.

Notice that in Kmeans and Kmeans++, we found the same optimal solution for the number of clusters with the Elbow method.

Dataset1: 5 clusters

Dataset2: 4 clusters

Time analysis for Kmeans:

F is the time for the run function

K number of clusters

I number of iterations until convergence

N number of data points in the dataset

D dimension of points in the dataset

c is constant time

$$F = Kc + I(NKDc + NDc) + c$$
$$= Kc + INKDc + INDc + c$$

- Kc is the initialization step, because initializing one cluster takes constant time.
- I represent the iteration until convergence, what inside the parentheses are the E and the M step.
- $NKDc$ is the E step where N the number points represented by the loop that iterates over all examples.
  KDc is the time for calculating the distance between a point and K centroids, where each distance needs Dc time to be calculated
- NDc appears in the M step when we try to calculate the centroids for K clusters, and for each cluster it takes $m_i * Dc$ time where m_i is the number of points in the cluster i, and Dc to sum the D dimensions, therefore

$$m_1 * Dc \ + \ m_2 * Dc \ + \ \ldots \ + \ m_k * Dc \ = \ Dc \left(\sum_i^K m_i \right)$$

But $\sum_i^K m_i = N$

Thus $K(Dc \ * \ \sum_i^K m_i) = NDc$

At the end we know that big-O sum of terms equals big-O to the max term, therefore F is

$$F = O(INKD)$$

Part 3:

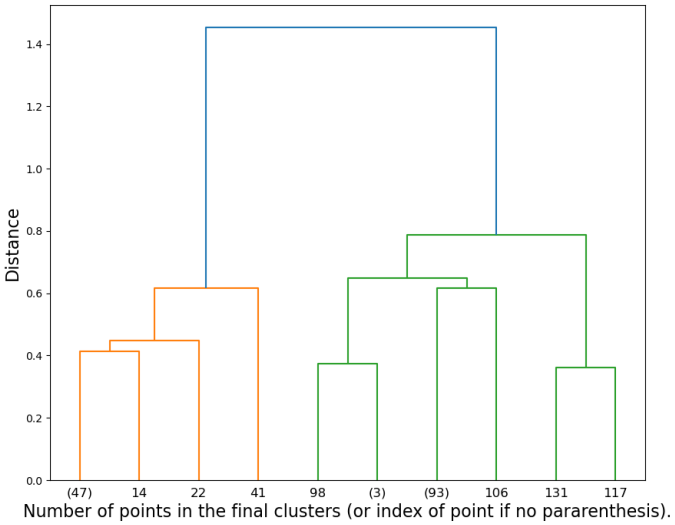
Hyperparameters to test for HAC:

- Linkage: Single, and Complete
- Distance: Euclidean, and Cosine

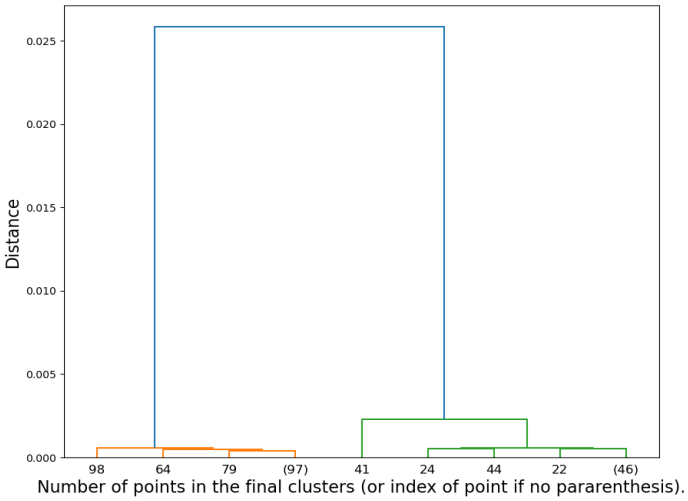
Total configurations: 4

Dendrograms representing the last 3 levels of the clustering:

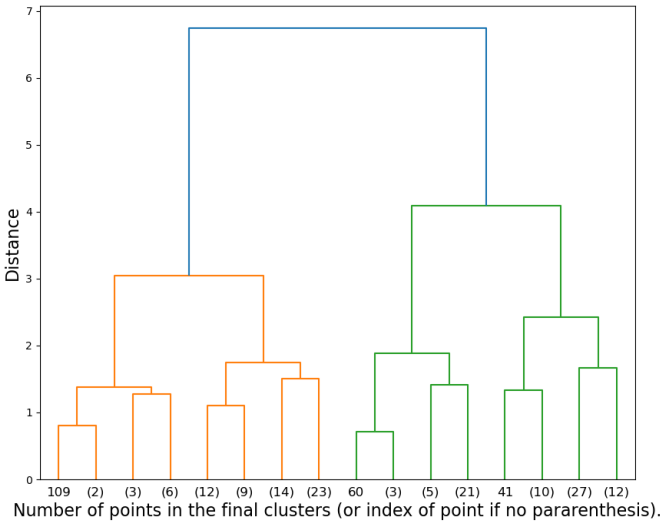
For single linkage and euclidean distance



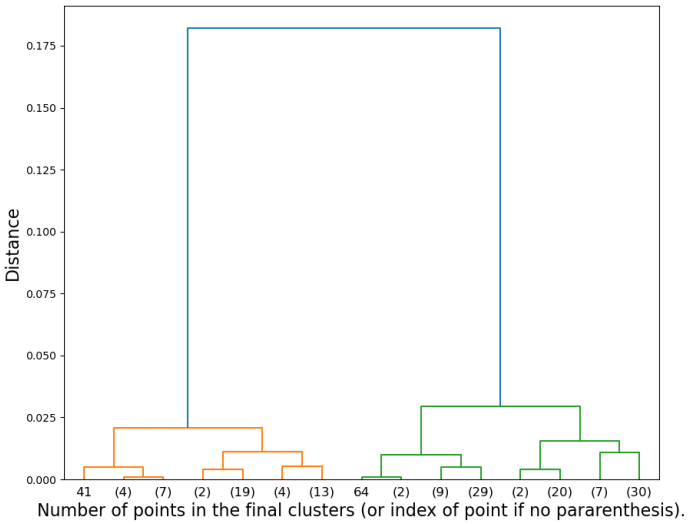
For single linkage and cosine distance



For complete linkage and euclidean distance



For complete linkage and cosine distance



From the dendrograms we can see that Complete linkage was joining points together in different clusters better than Single linkage, also we can see that we have two distinct clusters so K = 2 would be a reasonable choice.

Silhouette Analysis:

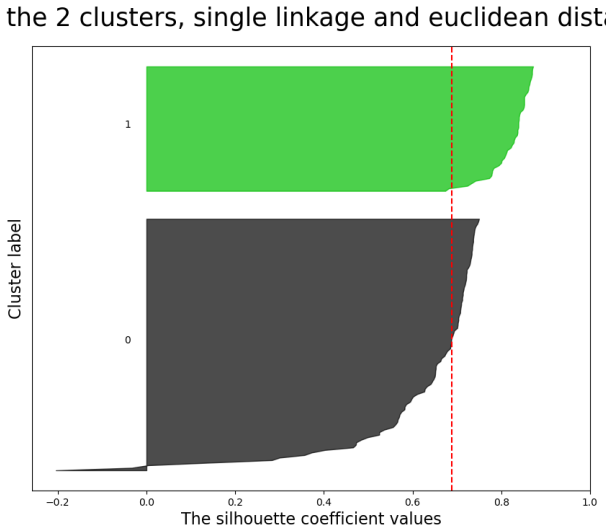
Hyperparameters to test for Agglomerative clustering:

- Linkage: Single, and Complete
- Distance: Euclidean, and Cosine
- Number of Clusters K: 2, 3, 4, and 5

Total configurations: 16

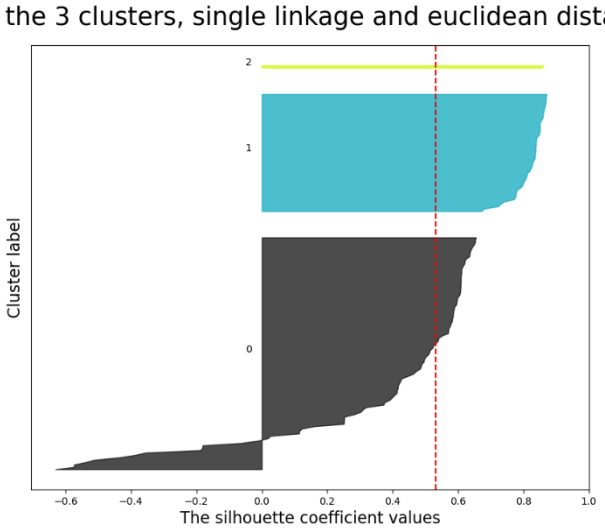
Note the Average Silhouette Score is presented as red dotted vertical line in the figures

1. For 2 clusters with Single linkage and Euclidean distance metric
   Average Silhouette Score: 0.6881



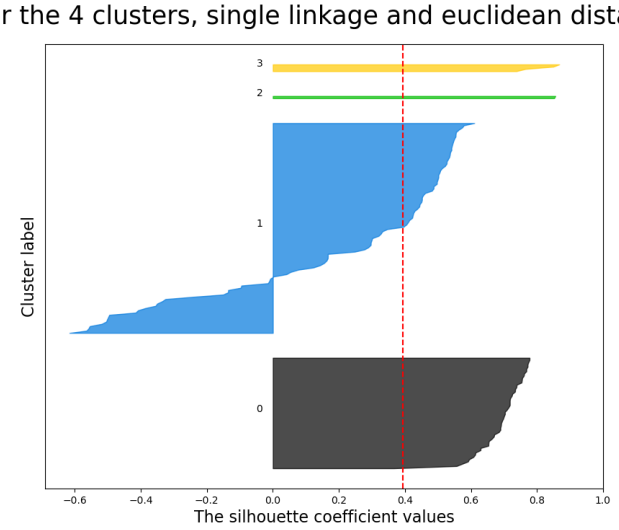
Comment: we can see a small portion in the first cluster (0) that have negative silhouette coefficient, which means they are probably in the wrong cluster.

2. For 3 clusters with Single linkage and Euclidean distance metric
   Average Silhouette Score: 0.5313



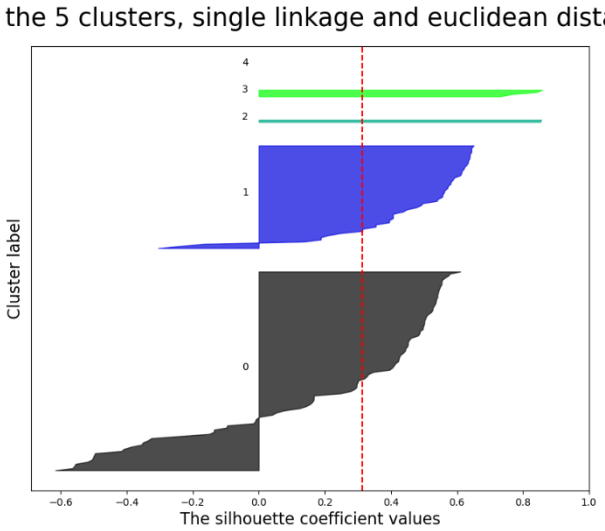
Comment: we can see a bigger portion in the first cluster (0) that have negative silhouette coefficient, which mean they are probably in the wrong cluster, also the last cluster (2) has small amount of points compared with the other clusters.

3. For 4 clusters with Single linkage and Euclidean distance metric
   Average Silhouette Score: 0.3944



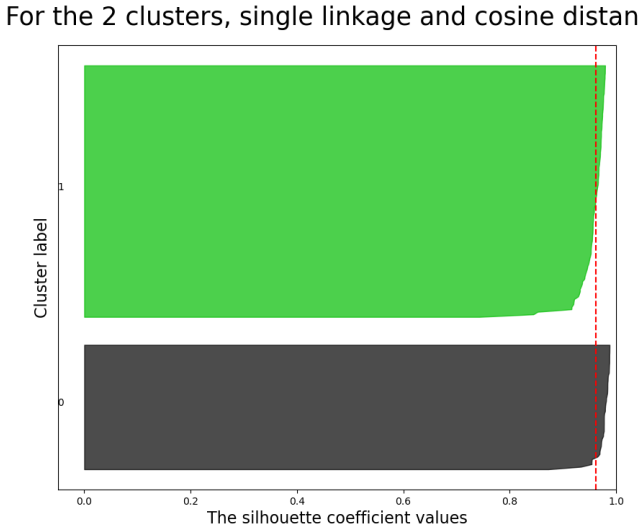
Comment: we can see that approximately quarter of (1) cluster that have negative silhouette coefficient, which mean they are in the wrong cluster, also the last clusters (2) and (3) are small compared with the other clusters.

4. For 5 clusters with Single linkage and Euclidean distance metric
   Average Silhouette Score: 0.3122



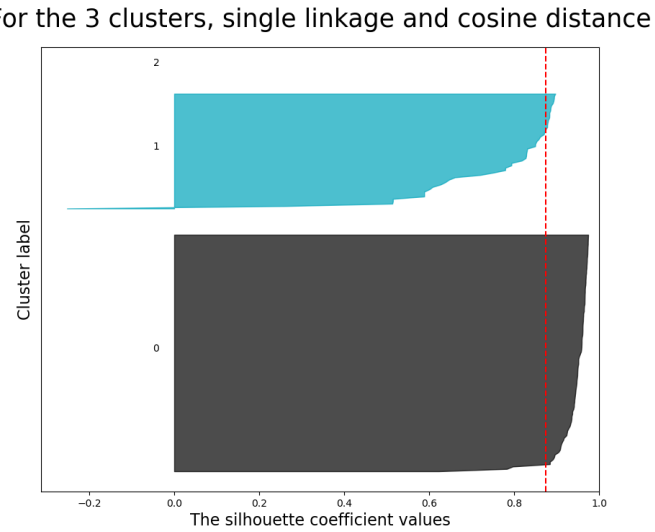
Comment: big amount of the first cluster (0) and small amount of (1) have negative silhouette coefficient, which mean they are in the wrong cluster, also the last cluster (4) is empty, also (3) and (2) are small compared with the other clusters.

5. For 2 clusters with Single linkage and Cosine distance metric
   Average Silhouette Score: 0.9625



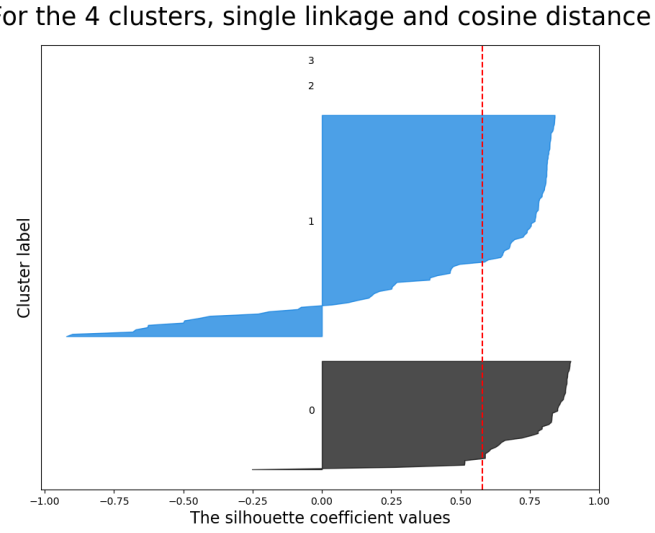
Comment: very high Silhouette Average and very high silhouette coefficient for all the values, resulting in a good clustering because all values in the right centroids with very high score (silhouette coefficient).

6. For 3 clusters with Single linkage and Cosine distance metric
   Average Silhouette Score: 0.8747



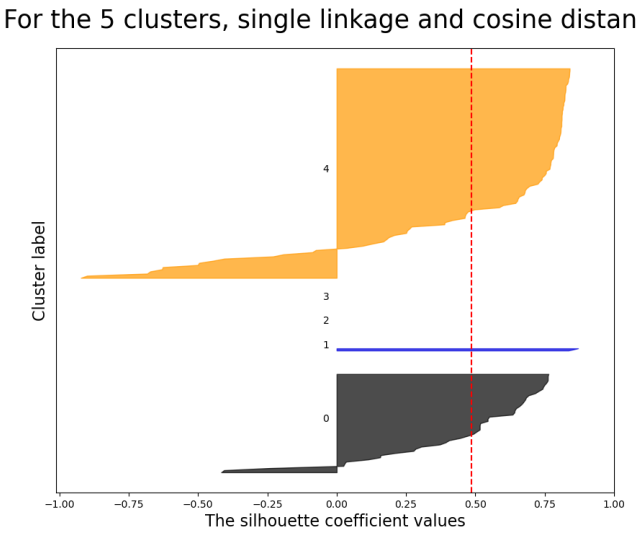
Comment: high Silhouette Average and high silhouette coefficient for all the values except some negative silhouette coefficient for (1) cluster, also the last cluster (2) is empty, even though it has high score, but we have empty cluster so it would be better for K to equal 2.

7. For 4 clusters with Single linkage and Cosine distance metric
   Average Silhouette Score: 0.5786



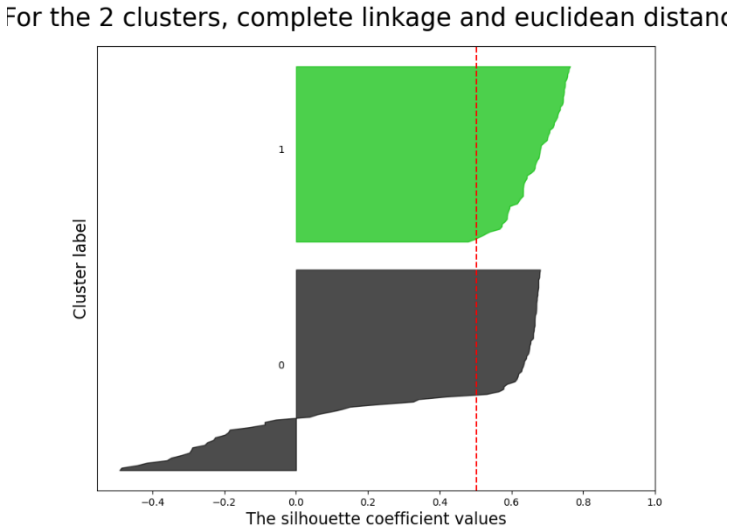
Comment: the last clusters are empty, and there is a portion that has high negative scores in cluster (1), and small one in the first cluster (0).

8. For 5 clusters with Single linkage and Cosine distance metric
   Average Silhouette Score: 0.4871



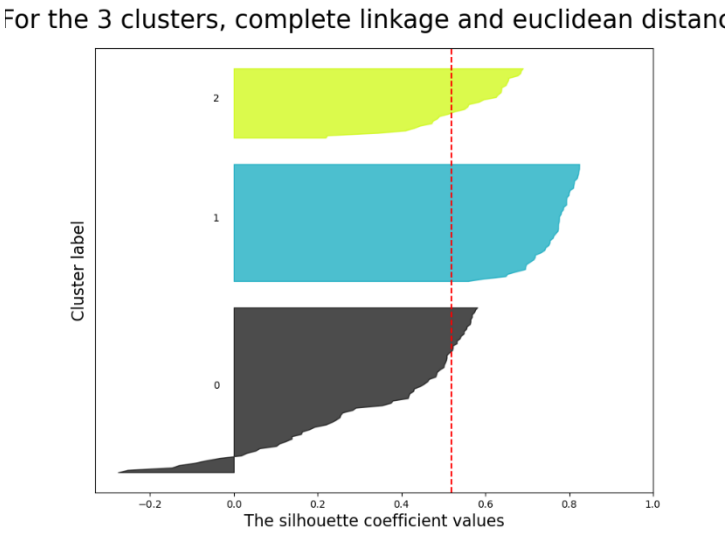
Comment: clusters (3) and (2) are empty, cluster (1) has a few points, and cluster (4) has high negative portion, and cluster (0) has some negative portion.

9. For 2 clusters with Complete linkage and Euclidean distance metric
   Average Silhouette Score: 0.5023



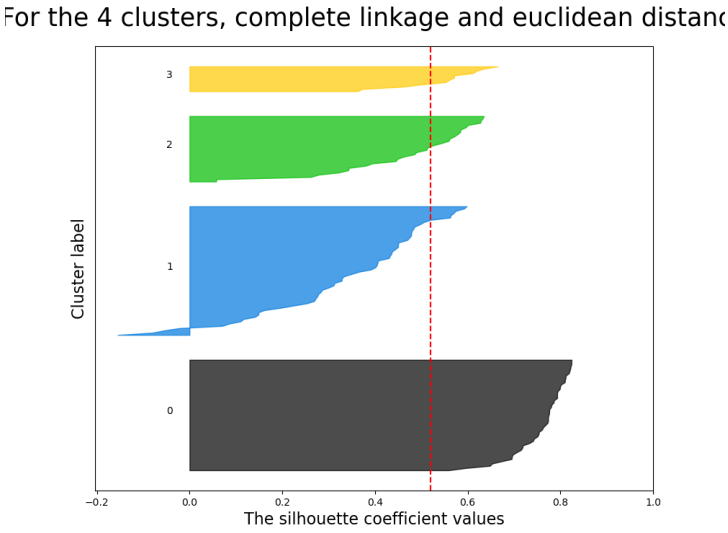
Comment: cluster (1) looks good, but cluster (0) has some negative values.

10. For 3 clusters with Complete linkage and Euclidean distance metric
    Average Silhouette Score: 0.5191



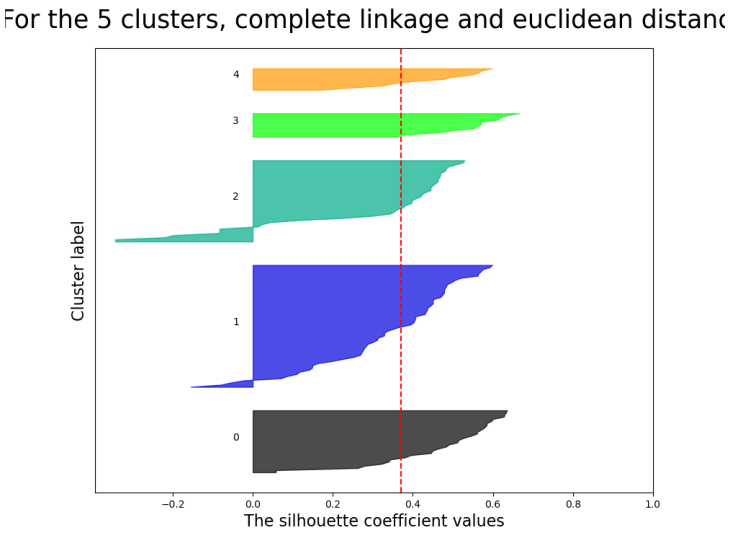
Comment: the first cluster (0) has some negative scores

11. For 4 clusters with Complete linkage and Euclidean distance metric
    Average Silhouette Score: 0.5199



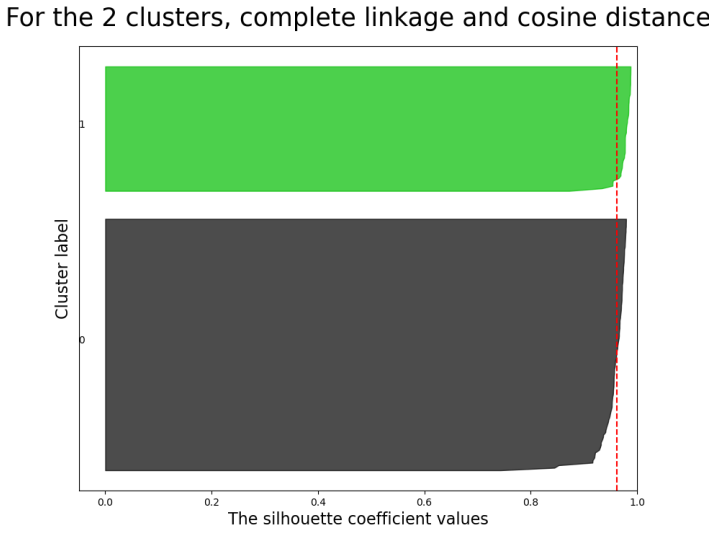
Comment: the first cluster (0) has most of the points and most of them has high score, cluster (1) has some negative and near 0 scores, and cluster (3) does not have points as much as the other clusters.

12. For 5 clusters with Complete linkage and Euclidean distance metric
    Average Silhouette Score: 0.3702



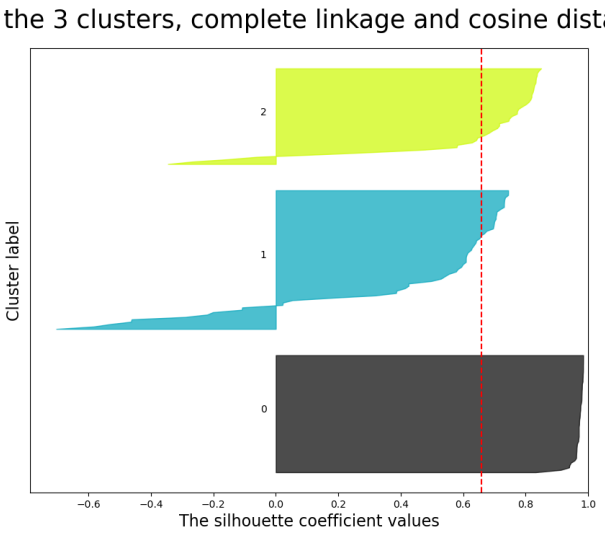
Comment: cluster (2) has some negative scores, and cluster (1) has few.

13. For 2 clusters with Complete linkage and Cosine distance metric
    Average Silhouette Score: 0.9625



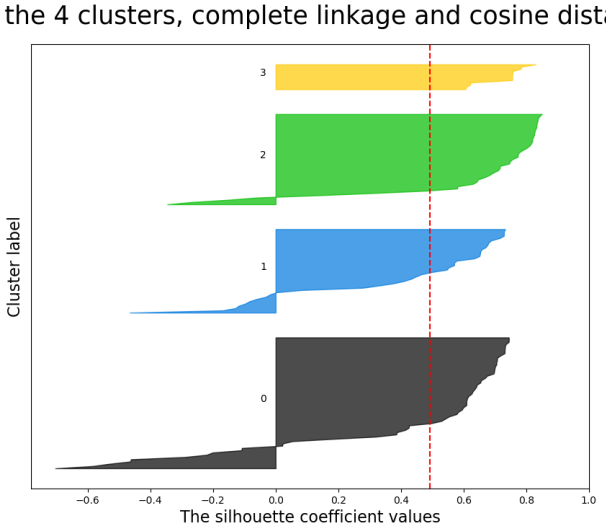
Comment: very high Silhouette Average and very high silhouette coefficient for all the values, resulting in a good clustering, (same as the fifth configuration).

14. For 3 clusters with Complete linkage and Cosine distance metric
    Average Silhouette Score: 0.6597



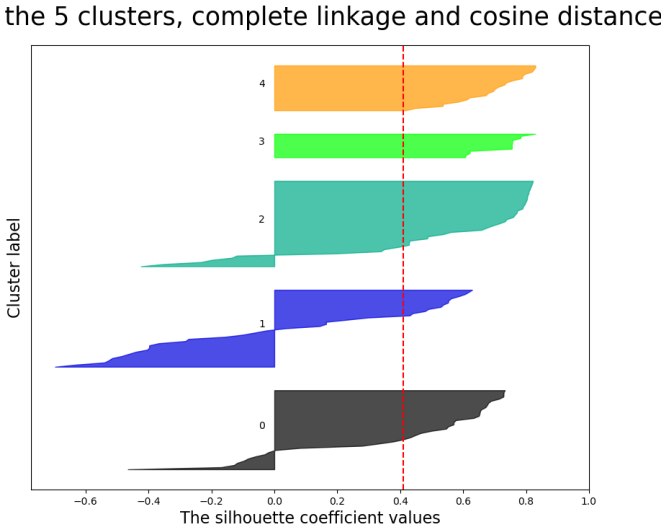
Comment: very high Silhouette Average and very high silhouette coefficient for all the values, resulting in a good clustering, (same as the fifth configuration).

15. For 4 clusters with Complete linkage and Cosine distance metric
    Average Silhouette Score: 0.4913



Comment: the last cluster (3) has few points compared to the rest, also the rest has some negative scores, with the first cluster having the highest negative scores.

16. For 5 clusters with Complete linkage and Cosine distance metric
    Average Silhouette Score: 0.4088



Comment: half of cluster (1) has negative scores, and (0) and (2) have some negative scores, cluster (3) is small with respect to the others.

Highest Average Silhouette value: 0.9625445604324341
The Best Hyperparameter Configuration Based on Average Silhouette score is the 5th configuration, with 2 Clusters Single Linkage and Cosine Distance.

Note: even though the 13th configuration also had 0.9625 Average Silhouette score but if we consider digits more than 4 after the decimal point, we find that the 5th had the highest score.

Time analysis for HAC:

F is the time for the HAC Algorithm

N number of data points in the dataset

D dimension of points in the dataset

c is constant time

when computing the distances whether we use Single or Complete linkage we need to find the distance between all data points to find the nearest to each other, and so for N points we need to find $N-1$ distances that needs time Dc each which results in $N * Dc(N-1)$ and its worst-case time is $O(N^2D)$

and now after we store the distances we found from before in a $N \times N$ similarity matrix we need to exhaustively search for the minimum distance for N data points each search will cost N^2 and we will do it N times, so the total cost will be N^3c and, and worst-case is $O(N^3)$

Therefore

$$F = O(N^3 + N^2D)$$

In normal cases $D \ll N$ so F is

$$F = O(N^3)$$

Note (Manning, 2008) there are more efficient ways to do the searching procedure with the priority queue that will store the distances in the similarity matrix in decreasing order, which will result F to be

$$F = O(N^2 \log N)$$

Also **only** for Single Linkage we can optimize even more by introducing a new matrix that will store the next best merge for the cluster because we already computed the similarity matrix in the first step, so we can append the next best option to this matrix so we can use it later when we want to merge the cluster instead of doing a search, and that works for Single linkage because it is concerned with the smallest distances between the points only which was already computed, and not like the other linkages that have other requirements, thus after eliminating the searching part we have

$$Single\ F = O(N^2)$$

Reference

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. URL http://nlp.stanford.edu/IR-book/.

Additional Question:

comment on which clustering method (Kmeans or HAC) you would prefer to use with a dataset consisting of 1 million data points each of which has a dimension of 120000 (i.e 200x200 RGB image)

Answer:

based on the formulas we constructed above, if we consider time complexity for Kmeans it will be as below where C is constant multiple

$$O(10^6 * 12 * 10^4 * K * I) = C * 1.2 * 10^{11}$$

because we can safely assume that $K * I << 1.2 * 10^{11}$

For HAC we can consider all the formulas from worst to best:

$$O(N^3 + N^2 D) = C * 1.12 * 10^{18}$$

$$O(N^3) = C * 10^{18}$$

$$O(N^2 \log N) = C * 6 * 10^{12}$$

Finally for the optimized Single Linkage Algorithm:

$$O(N^2) = C * 10^{12}$$

Therefore, for this task **I would prefer to use Kmeans** because it is much faster than the HAC, because it depends on linear N not like HAC, but if we had to run it many times Kmeans performance for this task would be close or even worse than the Single linkage optimized Algorithm, but in general Kmeans would be better than HAC for this task.