# Assignment 2 Report

Yıldırım Özen 2521862

November 30, 2024

## 0.1 Part 1

Here's my results for 10-fold knn repeated 5 times for 6 different hyperparameter configurations:

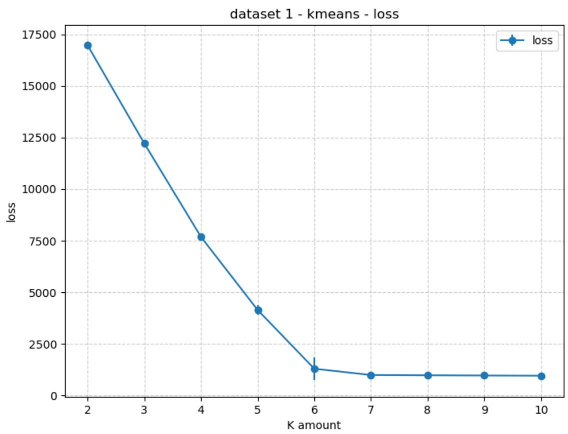
| k | metric | confidence- | confidence+ | average |
|---|---|---|---|---|
| 1 | cosine | 0.919 | 0.924 | 0.921 |
| 1 | minkowski p=2 | 0.888 | 0.896 | 0.892 |
| 3 | cosine | 0.948 | 0.954 | 0.951 |
| 3 | minkowski p=2 | 0.920 | 0.931 | 0.925 |
| 5 | cosine | 0.933 | 0.958 | 0.945 |
| 5 | minkowski p=2 | 0.890 | 0.904 | 0.897 |

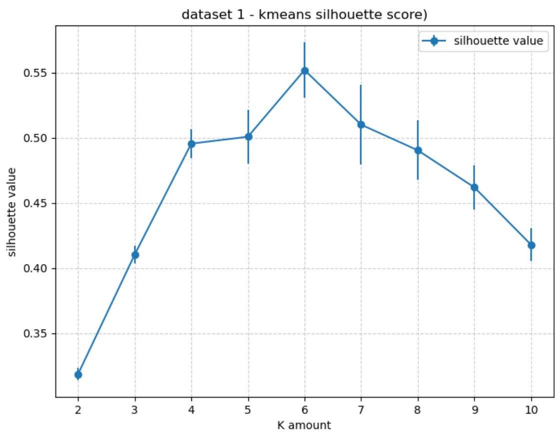
The best performing hyperparameters seem to be cosine with k=3 and cosine with k=5 based on the averages.

## 0.2 Part 2

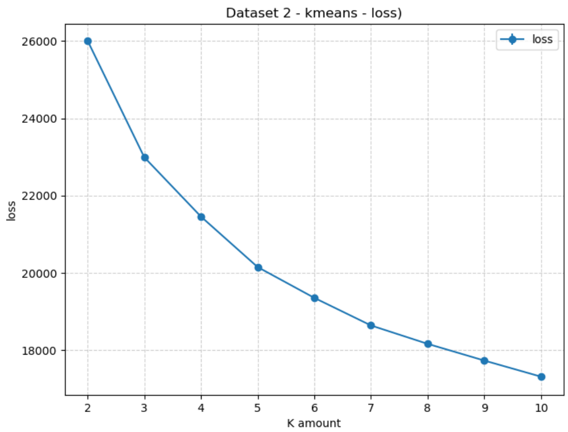
### 0.2.1 Kmeans

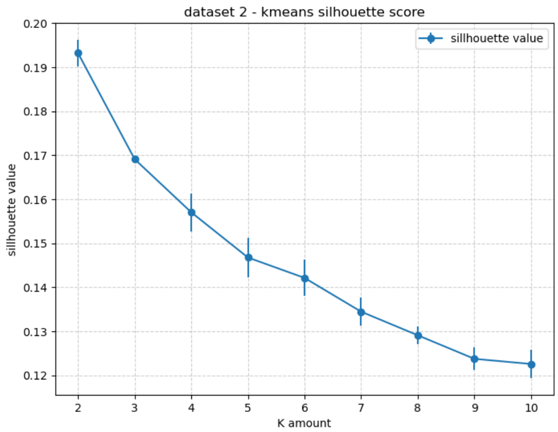
Here are the resulting plots for kmeans with confidence intervals:




As can be seen k=6 seem to be the best k value for this dataset according to both loss and silhouette scores.

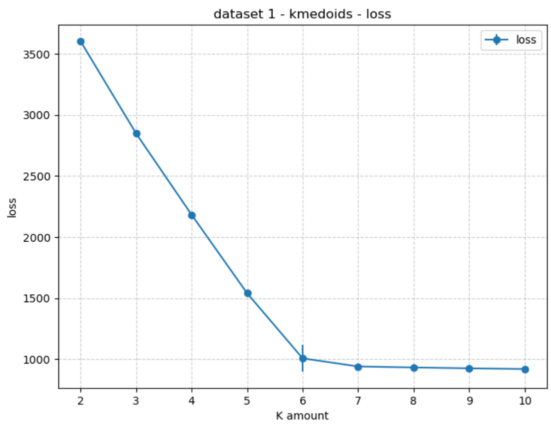


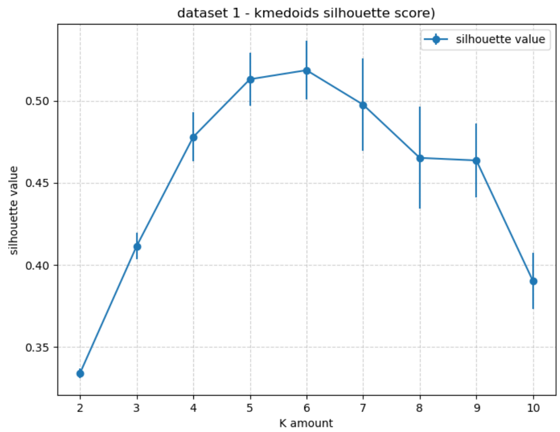




We can see from the dataset2 graphs that increasing k does not decrease loss a lot and decreases silhouette value, thus the clusters seem to be poorly seperated with every increasing k value. Thus I would think k=2 would be best for this dataset.

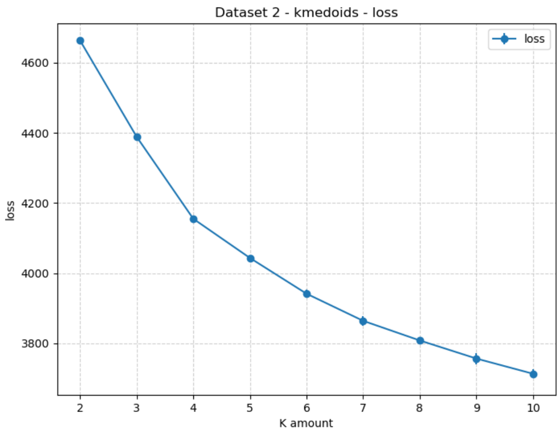
### 0.2.2 Kmedoids

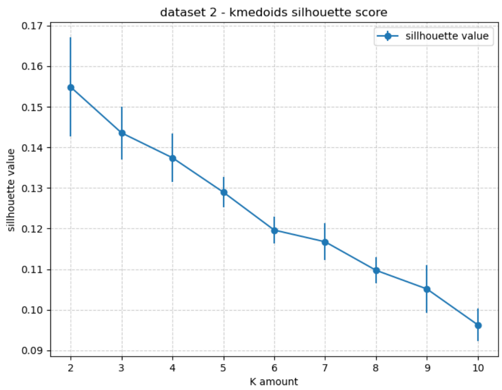
Here are the resulting plots for kmedoids with confidence intervals:






It can be seen that the elbow point is at k=6 and silhouette score tops at that point as well, thus it is our optimal k.

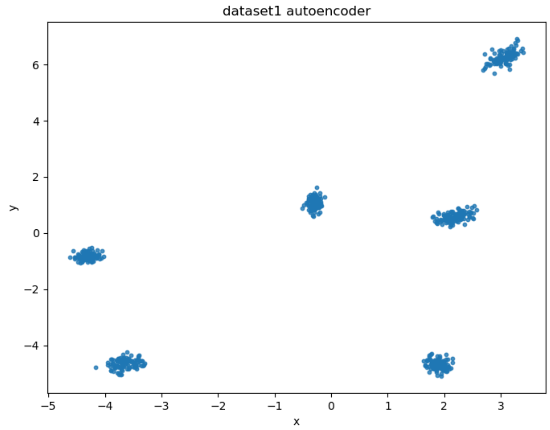


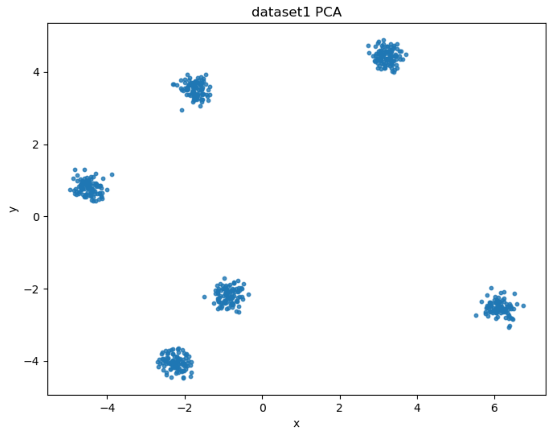



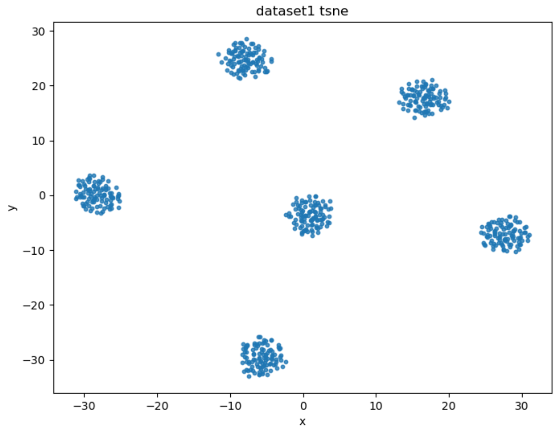

It can be seen that although loss is decreasing at a slow rate, silhouette is also decreasing, thus the clusters are poorly separated with every increasing k value. Thus I would think k=2 would be best for this dataset.

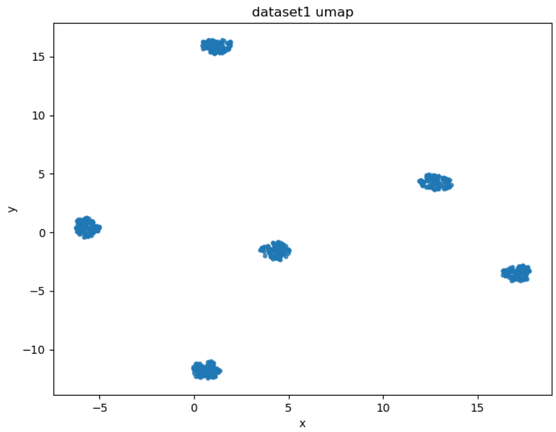
### 0.2.3 Dimensionality Reduction

Here are the plots of the reduction methods I've used for Dataset1:


dataset1 autoencoder
y
x

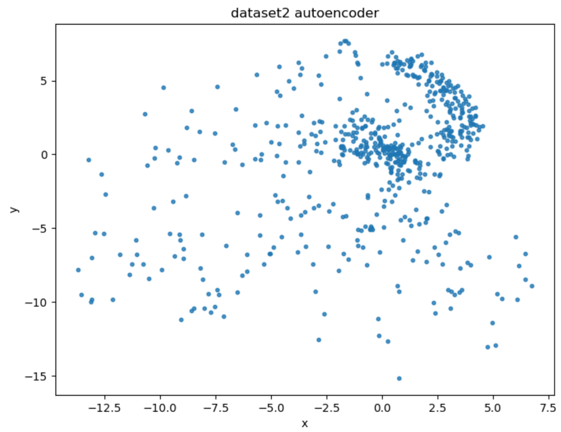

dataset1 PCA
y
x

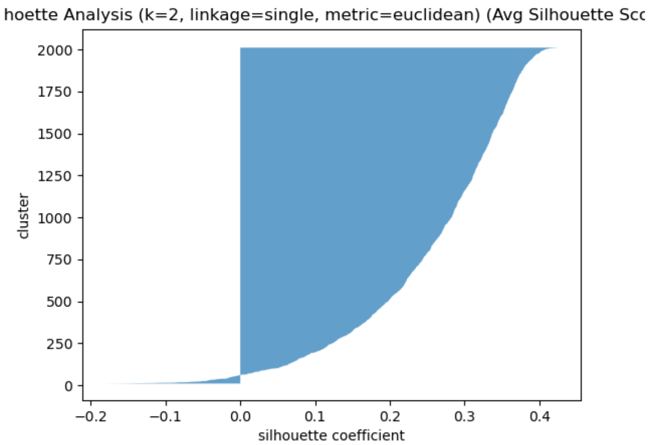


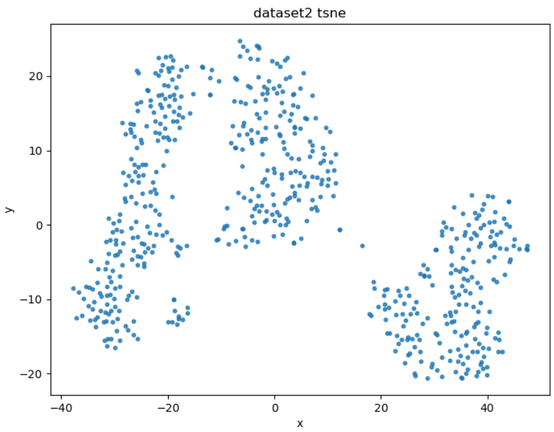



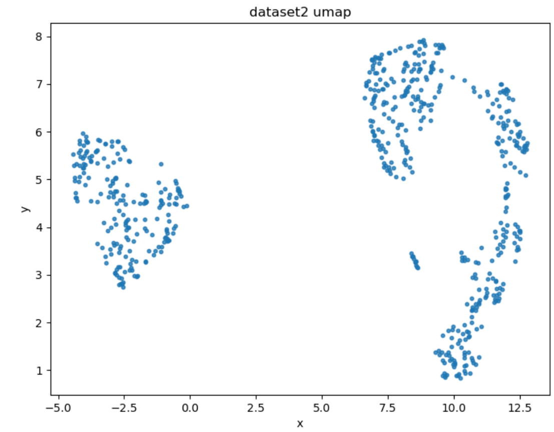

We can see that every method gave nearly the same results with 6 distinct clusters. Thus our result from elbow method was accurate.

Here are the plots of the reduction methods I've used for Dataset2:


dataset2 autoencoder
y
x
5
0
−5
−10
−15
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5


hoette Analysis (k=2, linkage=single, metric=euclidean) (Avg Silhouette Scc
cluster
silhouette coefficient
2000
1750
1500
1250
1000
750
500
250
0
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4




It’s harder to spot clusters with this dataset, however it can be said that there are 2 clusters, one on the left side and one on the right side. Thus our guess from the elbow method was also correct.

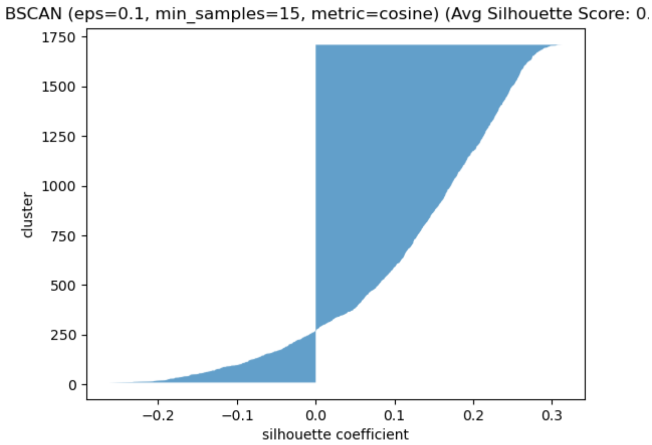
### 0.2.4 Worst-case runtime analysis

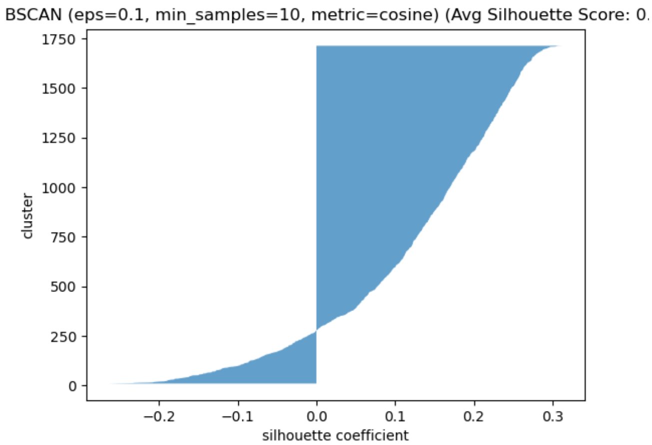
For kmeans, in every iteration, every point(N) is compared with each cluster centroid(K) to find the nearest one. Computing the distances between them depends on vector dimension(d). This is repeated I times so complexity is O(K * N * I * d) Updating the centroids have the same complexity. For kmedoids, assigning centers to points is the same as kmeans, however when you want to update the medoids, you have to compute the cost for every n data points instead of calculating only once, thus the complexity becomes O($n^2$). The final complexity becomes O(K* $N^2$ * I * d).

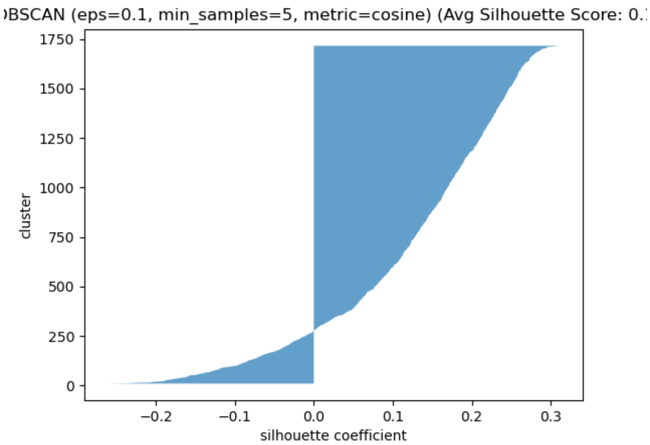
## 0.3 Part 3

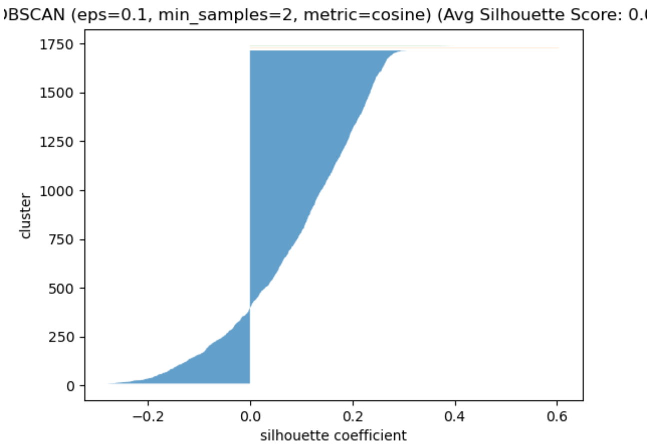
These are the four highest 4 silhouette score DBSCANs:

As can be seen, they all have eps=0.1 and metric = cosine. Three of them have min samples = 5 and one 2. Their average silhouette scores are very similar and they seem to fit for only one cluster.


BSCAN (eps=0.1, min_samples=15, metric=cosine) (Avg Silhouette Score: 0.
cluster
silhouette coefficient


BSCAN (eps=0.1, min_samples=10, metric=cosine) (Avg Silhouette Score: 0.
cluster
silhouette coefficient

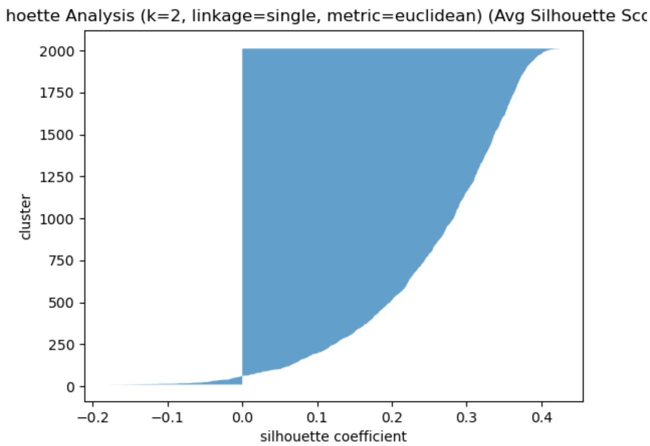


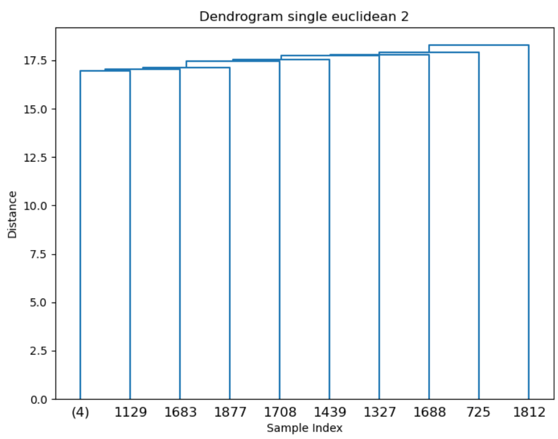



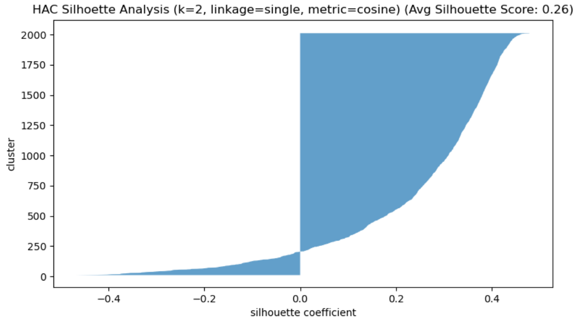

These are the HAC hyperparameter configurations and their average silhouette scores:
Running HAC with linkage=single, metric=euclidean, k=2
Average Silhouette Score: 0.25815263390541077
Running HAC with linkage=single, metric=euclidean, k=3
Average Silhouette Score: 0.1330309808254242
Running HAC with linkage=single, metric=euclidean, k=4
Average Silhouette Score: 0.1240428239107132
Running HAC with linkage=single, metric=euclidean, k=5
Average Silhouette Score: 0.11423469334840775
Running HAC with linkage=single, metric=cosine, k=2
Average Silhouette Score: 0.25596165657043457
Running HAC with linkage=single, metric=cosine, k=3
Average Silhouette Score: 0.24046701192855835
Running HAC with linkage=single, metric=cosine, k=4

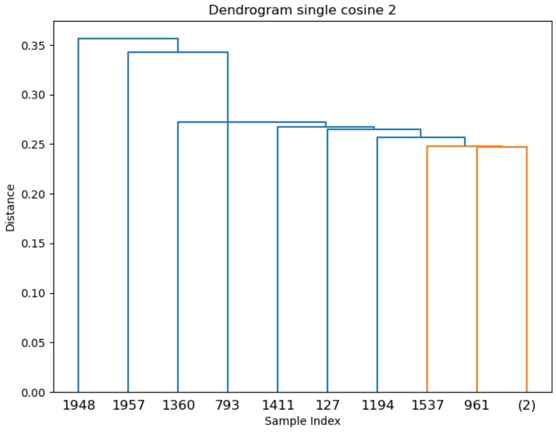
Average Silhouette Score: 0.2084643840789795
Running HAC with linkage=single, metric=cosine, k=5
Average Silhouette Score: 0.20167870819568634
Running HAC with linkage=complete, metric=euclidean, k=2
Average Silhouette Score: 0.13750071823596954
Running HAC with linkage=complete, metric=euclidean, k=3
Average Silhouette Score: 0.07379759103059769
Running HAC with linkage=complete, metric=euclidean, k=4
Average Silhouette Score: 0.055272724479436874
Running HAC with linkage=complete, metric=euclidean, k=5
Average Silhouette Score: 0.037721045315265656
Running HAC with linkage=complete, metric=cosine, k=2
Average Silhouette Score: 0.08257857710123062
Running HAC with linkage=complete, metric=cosine, k=3
Average Silhouette Score: 0.044918715953826904
Running HAC with linkage=complete, metric=cosine, k=4
Average Silhouette Score: 0.0376201793551445
Running HAC with linkage=complete, metric=cosine, k=5
Average Silhouette Score: 0.035240765661001205
The highest Average Silhouette Score is with linkage=single, metric=euclidean, k=2

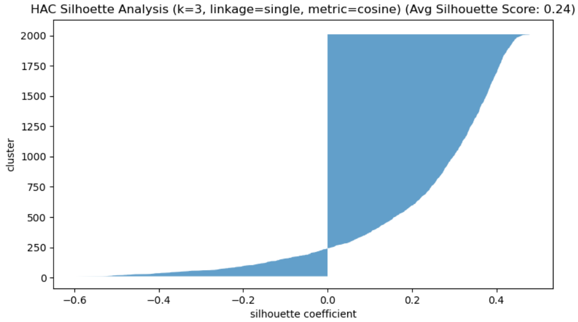
Here are the Dendograms and silhouette analysises for the best four HAC silhouette scores:

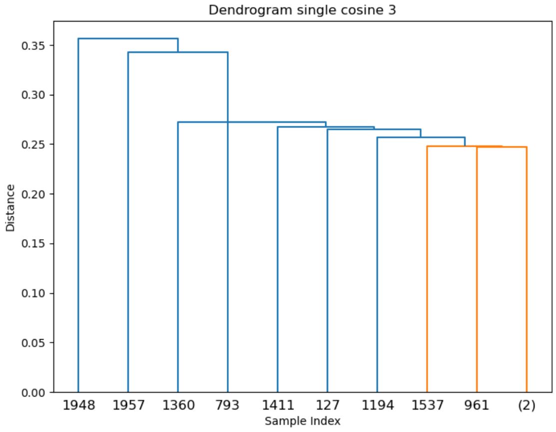


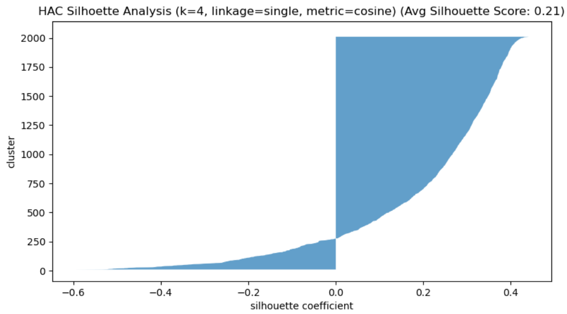



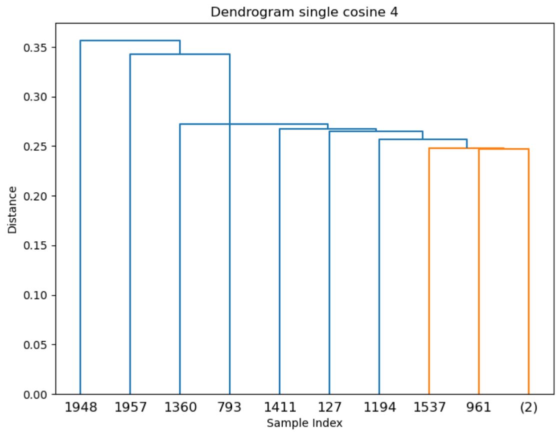

HAC Silhoette Analysis (k=2, linkage=single, metric=cosine) (Avg Silhouette Score: 0.26)
cluster
silhouette coefficient


Dendrogram single cosine 2
Distance
Sample Index
1948
1957
1360
793
1411
127
1194
1537
961
(2)


HAC Silhouette Analysis (k=3, linkage=single, metric=cosine) (Avg Silhouette Score: 0.24)
cluster
silhouette coefficient
2000
1750
1500
1250
1000
750
500
250
0
−0.6
−0.4
−0.2
0.0
0.2
0.4


Dendrogram single cosine 3
Distance
Sample Index
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1948
1957
1360
793
1411
127
1194
1537
961
(2)




As can be seen, these silhouette plots are also nearly identical and they all have single linkage. They also have small k values (2, 3 and 4). Three of them have cosine metric and one has euclidan. The last three dendograms show two clusters while the first one shows only one dendogram.

### 0.3.1 Worst-case runtime analysis

HAC starts by calculating distances between all pairs of data points, which takes O($n^2 * D$). Then merges the clusters which takes O($n^2$). This is done n times so the complexity is O($n^3$). If the algorithm utilizes a heap it can be optimized to O($n^2 logn$) Thus the final equation becomes O($n^2 * d + n^2 logn$). For N = 1 million and d = 120000 kmeans finishes in around t = $10^{17}$ while HAC finishes in $10^18$ or $10^17$, depending on whether heap is utilized or not. Thus kmeans would be faster if a naive approach is implemented in HAC.