CENG499 Homework 2

Full Name: Berk Ulutaş

Student Number: 2522084

1 Part 1

K	Distance Function	Accuracy	Confidence Interval
3	Cosine	0.96	[0.945, 0.975]
3	Minkowski (p=2)	0.888	[0.864, 0.913]
3	Minkowski (p=3)	0.91	[0.884, 0.936]
3	Mahalanobis	0.9	[0.877, 0.923]
5	Cosine	**0.967**	[0.953, 0.981]
5	Minkowski (p=2)	0.877	[0.85, 0.904]
5	Minkowski (p=3)	0.823	[0.793, 0.854]
5	Mahalanobis	0.883	[0.855, 0.911]
10	Cosine	0.898	[0.877, 0.92]
10	Minkowski (p=2)	0.86	[0.835, 0.885]
10	Minkowski (p=3)	0.838	[0.809, 0.868]
10	Mahalanobis	0.873	[0.853, 0.894]
30	Cosine	0.832	[0.804, 0.86]
30	Minkowski (p=2)	0.855	[0.829, 0.881]
30	Minkowski (p=3)	0.835	[0.805, 0.865]
30	Mahalanobis	0.8	[0.768, 0.832]
50	Cosine	0.843	[0.813, 0.874]
50	Minkowski (p=2)	0.837	[0.806, 0.867]
50	Minkowski (p=3)	0.828	[0.799, 0.858]
50	Mahalanobis	0.822	[0.793, 0.85]

Table 1: Accuracy and Confidence Intervals for Different Hyperparameter Configurations

In this part KNN method was evaluated using different hyperparameter configurations. The tested hyperparameters were as follows:

- Number of Neighbors $K = [3, 5, 10, 30, 50]$, 5 different parameters
- Distance Metrics: 4 different parameters
  - Cosine Distance
  - Minkowski Distance ($p = 2$): Equivalent to euclidean distance
  - Minkowski Distance ($p = 3$)

  - Mahalanobis Distance

Combination of 5 values of K and 4 distance functions resulted in $5 \cdot 4 = 20$ unique configurations. Each configuration was evaluated through a 10 fold cross validation process, repeated 5 times with shuffling to ensure robust results and to mitigate the effect of any particular partitioning of the dataset. Results can be seen on Table 1.

1.1 Comments on Results

- I have chosen the best hyperparameter configuration according to accuracy. The combination of $K = 5$ and the **Cosine distance** function achieved the highest accuracy **(0.967)** with a narrow confidence interval of [**0.953, 0.981**], indicating both high performance and consistency across the dataset.
- Smaller K values ($K = 3$ or $K = 5$), achieved higher accuracy, as the model can better capture local patterns in the data.
- Larger K values ($K = 30$ or $K = 50$), resulted in lower accuracy due to over smoothing, where the model begins to generalize excessively.
- Generally, Cosine distance consistently outperformed other distance functions. Minkowski showed reasonable performance, but its accuracy decreased with larger K values. Mahalanobis distance performed well for smaller K, but its performance declined with increasing K.
- The test accuracy of 0.9 was slightly lower than the best validation accuracy, which suggests minimal overfitting and good generalization capabilities of the chosen configuration.git

2 Part 2

In this part K Means and K Medoids methods were evaluated. For each method and Dataset tuple confidence intervals and plots are given. I commented on them individually but here I will make a general comment.

- For Dataset 1, both K Means and K-Medoids suggest $K = 6$ as the optimal cluster count based on the loss and silhouette score analysis.
- For Dataset 2, K Means does not provide a clear optimal K, but K-Medoids suggests $K = 4$ based on silhouette score.
- Many of the confidence intervals in this study are very narrow, particularly for the loss values. This suggests that the clustering results are consistent and reliable across multiple runs or cross validation folds.
- However, for silhouette scores, certain values of K ($K = 8, 9, 10$) exhibit slightly wider intervals, reflecting less stability in clustering quality.

2.1 K Means

2.1.1 Dataset 1

K	Average Loss	Confidence Interval
2	16873.235	(16873.235, 16873.235)
3	12211.566	(12207.469, 12215.662)
4	7645.710	(7607.390, 7684.029)
5	3950.311	(3950.311, 3950.311)
6	1010.421	(1010.421, 1010.421)
7	997.093	(996.850, 997.336)
8	984.508	(984.090, 984.927)
9	972.444	(971.894, 972.994)
10	962.101	(961.416, 962.786)

Table 2: Confidence Intervals for K Means Loss on Dataset 1

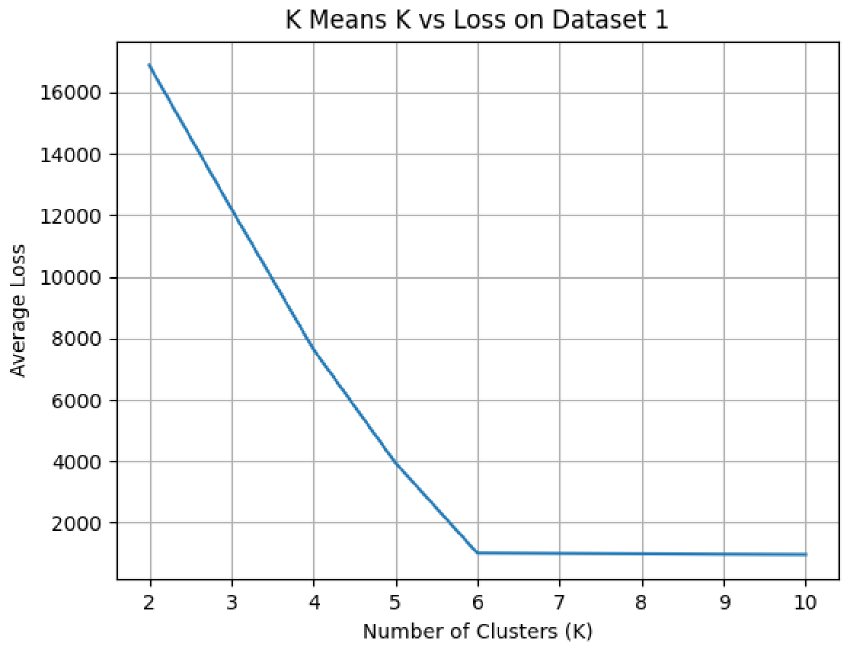


Figure 1: K Means: K vs Loss on Dataset 1.

- Figure 1 shows a sharp decrease in average loss while K increases from 2 to 6. This indicates that improvements in clustering.
- After $K = 6$, the curve flattens. This shows us that adding more clustering does not have big reductions in loss. This aligns with elbow method, which suggests $K = 6$ as the optimal number of clusters for Dataset 1.

K	Average Silhouette Score	Confidence Interval
2	0.348	(0.348, 0.348)
3	0.442	(0.442, 0.442)
4	0.566	(0.566, 0.566)
5	0.672	(0.671, 0.673)
6	0.781	(0.781, 0.781)
7	0.666	(0.661, 0.670)
8	0.635	(0.606, 0.663)
9	0.539	(0.537, 0.540)
10	0.480	(0.444, 0.515)

Table 3: Confidence Intervals for K Means Silhouette Scores on Dataset 1

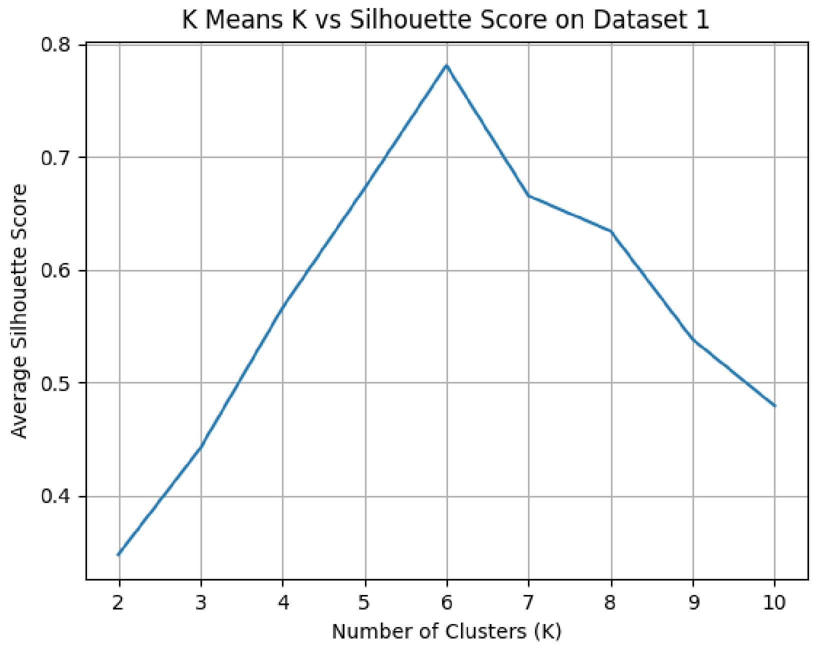


Figure 2: K Means: K vs Silhouette Score on Dataset 1.

- Figure 2 shows the silhouette score peaks at $K = 6$, which indicates that this configuration achieves the best balance in between other clustering configurations.
- The scores decrease beyond $K = 6$, this shows that adding more clusters leads to overlapping clusters or poorly seperated clusters.
- Also the results of K Means loss and Silhouette Score for Dataset 1 confirms each other. Both show that the best $K = 6$.

2.1.2 Dataset 2

K	Average Loss	Confidence Interval
2	26007.758	(26007.758, 26007.758)
3	22994.018	(22994.018, 22994.018)
4	21459.457	(21459.457, 21459.457)
5	20154.096	(20153.725, 20154.467)
6	19357.524	(19356.116, 19358.932)
7	18642.263	(18639.382, 18645.145)
8	18163.244	(18156.430, 18170.058)
9	17737.948	(17729.919, 17745.978)
10	17299.473	(17284.812, 17314.133)

Table 4: Confidence Intervals for K Means Loss on Dataset 2

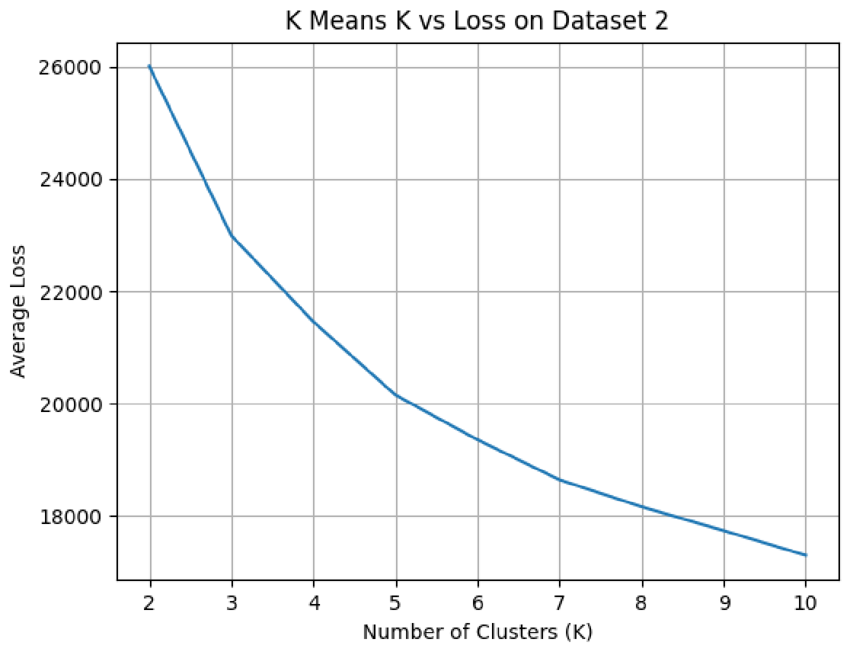


Figure 3: K Means: K vs Loss on Dataset 2.

- Figure 3 shows the loss decreases steadily without a clear elbow point. The absence of a sharp elbow suggests that clustering compactness improves consistently with higher K, but there is no clear elbow point. We cannot have clear groupings.
- This lack of a clear elbow point could be attributed to the curse of dimensionality. When we checked the shape of Dataset 2, it is (600, 784), where 784 represents a high dimensional

feature space. High dimensional data can make it challenging for K Means to form compact and distinct clusters, as the distance between points becomes less meaningful in higher dimensions.

- In contrast, Dataset 1 has a shape of (600, 40), which is much lower dimensional. This likely contributes to the clearer elbow point observed in the loss plot for Dataset 1, as the clusters can be more easily distinguished in a lower dimensional space.

K	**Average Silhouette Score**	**Confidence Interval**
2	0.205	(0.205, 0.205)
3	0.183	(0.176, 0.190)
4	0.174	(0.174, 0.174)
5	0.171	(0.170, 0.172)
6	0.158	(0.150, 0.167)
7	0.161	(0.156, 0.166)
8	0.146	(0.140, 0.151)
9	0.140	(0.138, 0.141)
10	0.139	(0.138, 0.140)

Table 5: Confidence Intervals for K Means Silhouette Scores on Dataset 2

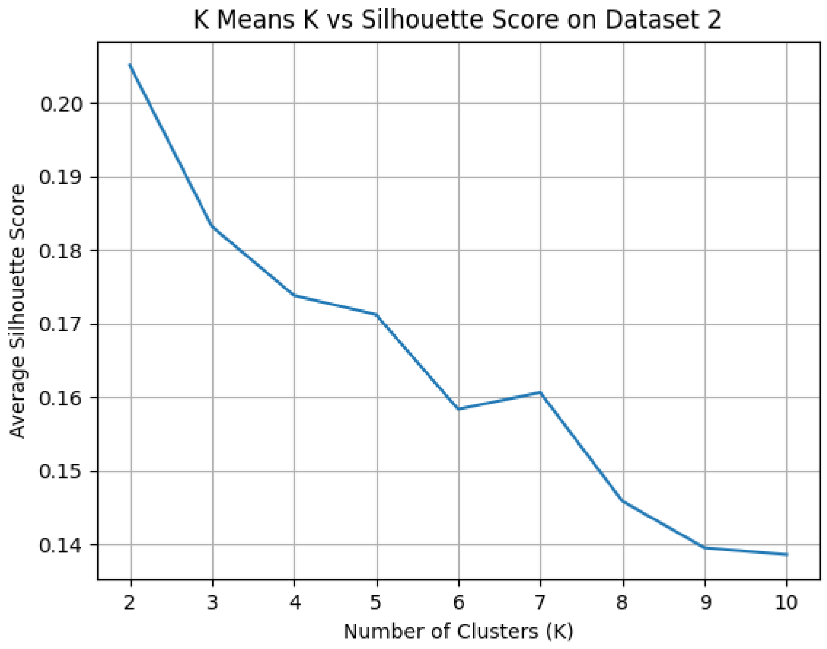


Figure 4: K Means: K vs Silhouette Score on Dataset 2.

- Figure 4 shows the silhouette score decreases as K increases, indicating that higher values of K do not improve the clustering quality.

- This is consistent with the loss plot, where a clear optimal K is not observed. This suggests Dataset 2 may have a more complex structure affecting clustering quality.

2.2 K Medoids

2.2.1 Dataset 1

K	Average Loss	Confidence Interval
2	376.845	(376.845, 376.845)
3	284.587	(284.587, 284.587)
4	184.493	(184.493, 184.493)
5	100.722	(100.722, 100.722)
6	19.453	(19.453, 19.453)
7	19.296	(19.296, 19.296)
8	19.159	(19.159, 19.159)
9	18.984	(18.984, 18.984)
10	18.673	(18.673, 18.673)

Table 6: Confidence Intervals for K Medoids Loss on Dataset 1

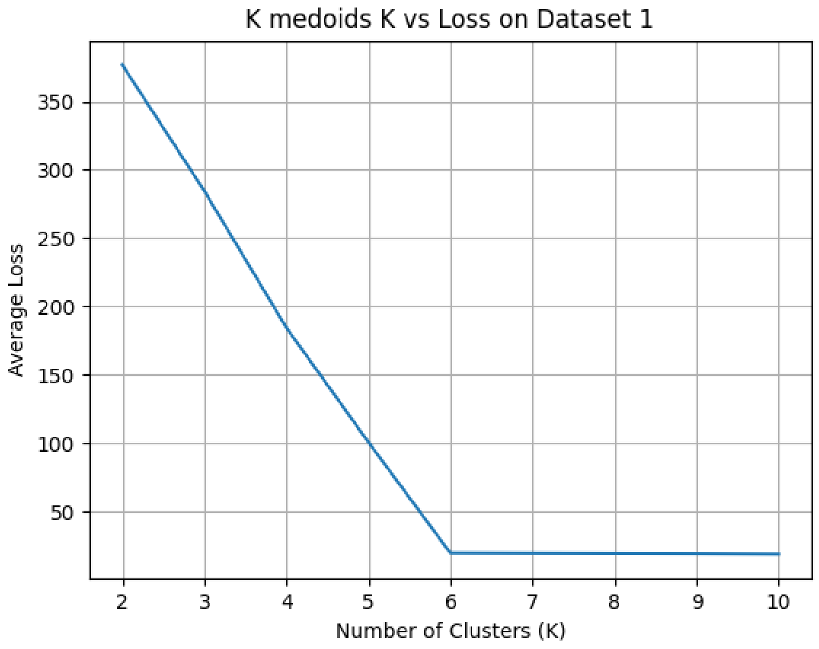


Figure 5: K Medoids: K vs Loss on Dataset 1.

- Similar to K Means, the loss decreases sharply until $K = 6$, after which it stabilizes as clearly seen in Figure 5 . This indicates that $K = 6$ provides the optimal clustering configuration.

- There is a clear elbow point on the given plot. This finding also supports to K means finding for Dataset 1.

K	Average Silhouette Score	Confidence Interval
2	0.300	(0.300, 0.300)
3	0.393	(0.393, 0.393)
4	0.507	(0.507, 0.507)
5	0.627	(0.627, 0.627)
6	0.781	(0.781, 0.781)
7	0.656	(0.656, 0.656)
8	0.530	(0.530, 0.530)
9	0.529	(0.529, 0.529)
10	0.530	(0.530, 0.530)

Table 7: Confidence Intervals for K Medoids Silhouette Scores on Dataset 1

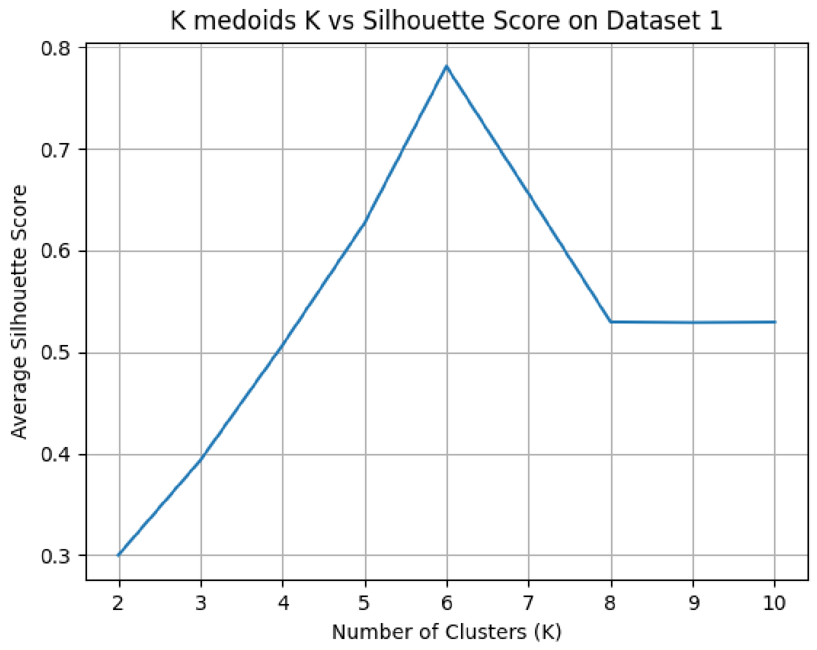


Figure 6: K Medoids: K vs Silhouette Score on Dataset 1.

- Figure 6 shows the silhouette score peaks at $K = 6$, consistent with the K Means results, confirming this as the optimal number of clusters.

2.2.2 Dataset 2

K	Average Loss	Confidence Interval
2	496.794	(496.794, 496.794)
3	352.596	(352.596, 352.596)
4	305.870	(305.870, 305.870)
5	299.963	(299.963, 299.963)
6	281.180	(281.180, 281.180)
7	275.242	(275.242, 275.242)
8	265.805	(265.805, 265.805)
9	256.511	(256.511, 256.511)
10	247.547	(247.547, 247.547)

Table 8: Confidence Intervals for K Medoids Loss on Dataset 2

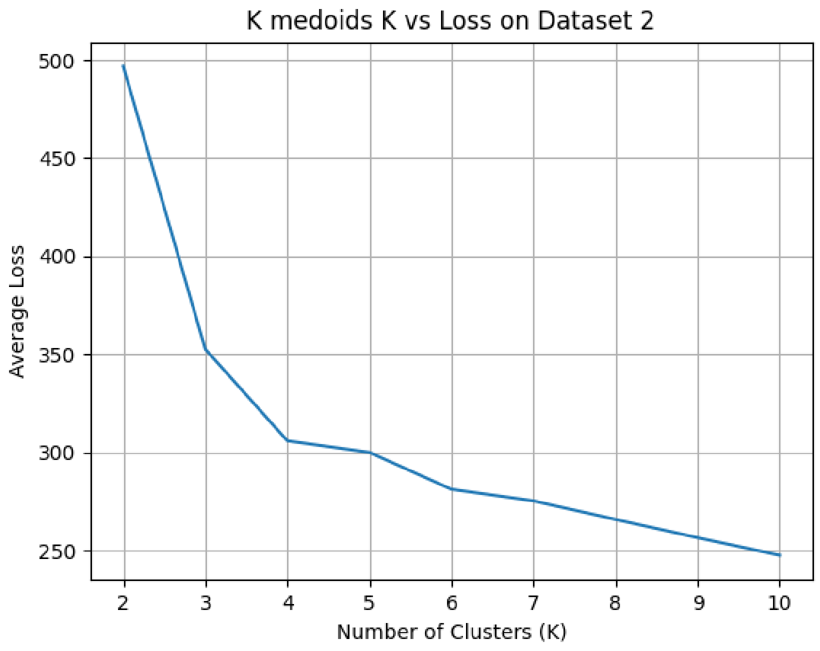


Figure 7: K Medoids: K vs Loss on Dataset 2.

- The loss decreases steadily with no clear elbow point, similar to K Means. This suggests a gradual improvement in compactness with higher K.
- Since our clustering test is between 2 and 10, we might not be able to find a clear elbow point for clustering more than 10 clusters. It is possible that the elbow point lies at some value of K greater than 10.

- Additionally, the lack of a sharp elbow within this range could also be influenced by the high dimensionality of the data (784 features), which impacts the effectiveness.

K	Average Silhouette Score	Confidence Interval
2	0.064	(0.064, 0.064)
3	0.096	(0.096, 0.096)
4	0.131	(0.131, 0.131)
5	0.097	(0.097, 0.097)
6	0.087	(0.087, 0.087)
7	0.082	(0.082, 0.082)
8	0.086	(0.086, 0.086)
9	0.096	(0.096, 0.096)
10	0.097	(0.097, 0.097)

Table 9: Confidence Intervals for K Medoids Silhouette Scores on Dataset 2

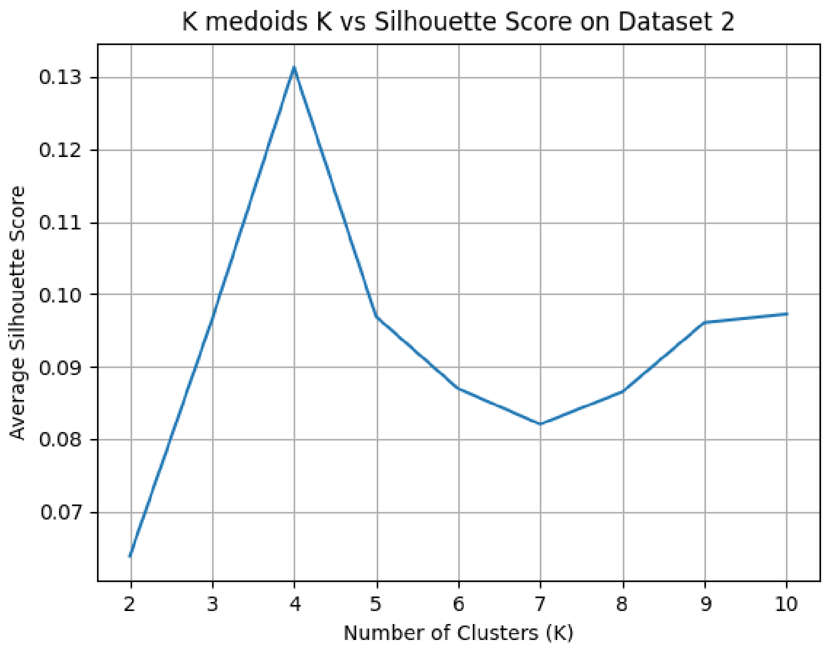


Figure 8: K Medoids: K vs Silhouette Score on Dataset 2.

- Figure 8 shows the silhouette score peaks at $K = 4$, differing from the K Means results. This could indicate that K-Medoids captures a different underlying cluster structure in Dataset 2.
- Further investigation, such as visualizing the data in a lower-dimensional space (e.g., using PCA, t-SNE, UMAP, autoencoders) could help clarify why K-Medoids identifies $K = 4$ as the optimal cluster number.

2.3 Worst Case Runtime Analysis

2.3.1 K Means

The worst-case running time of K Means depends on:

- N: Number of data points
- d: Dimension of the data sample vectors
- K: Number of clusters
- I: Number of iterations until convergence

Steps in K Means:

1. **Distance Computation:**
   - Each data point computes its distance to all K centroids. The cost of computing the distance is $\mathcal{O}(d)$.
   - Total cost for one iteration: $\mathcal{O}(N \cdot K \cdot d)$.
2. **Centroid Update:**
   - Updating the centroids involves averaging the coordinates of all points in a cluster.
   - Cost per cluster: $\mathcal{O}(N \cdot d)$.
   - Total cost for K clusters: $\mathcal{O}(K \cdot N \cdot d)$.

Total Running Time:

- The algorithm runs for I iterations.
- Total cost: $\mathcal{O}(I \cdot N \cdot K \cdot d)$.

2.3.2 K-Medoids

The worst-case running time of K-Medoids depends on:

- N: Number of data points
- d: Dimension of the data sample vectors
- K: Number of clusters
- I: Number of iterations until convergence

Steps in K-Medoids:

1. **Distance Computation:**
   - Each data point computes its distance to all K medoids, with a cost of $\mathcal{O}(d)$ per distance.

- Total cost per iteration: $\mathcal{O}(N \cdot K \cdot d)$.

2. **Medoid Update:**

   - For each cluster, the algorithm considers swapping the current medoid with another data point to minimize the total cost.
   - Cost of evaluating all possible swaps: $\mathcal{O}(N \cdot d)$ per cluster.
   - Total cost for K clusters: $\mathcal{O}(K \cdot N^2 \cdot d)$.

Total Running Time:

- The algorithm runs for I iterations.
- Total cost: $\mathcal{O}(I \cdot K \cdot N^2 \cdot d)$.

2.4 Dimensionality Reduction

Since the homework text states that the definition of "best" is up to you, In this report, I chose the "best" dimensionality reduction results are defined as those where the scatter plots clearly show distinct clusters with minimal overlapping points and clear boundaries between clusters.

2.4.1 PCA

PCA method does not have any hyperparameter.

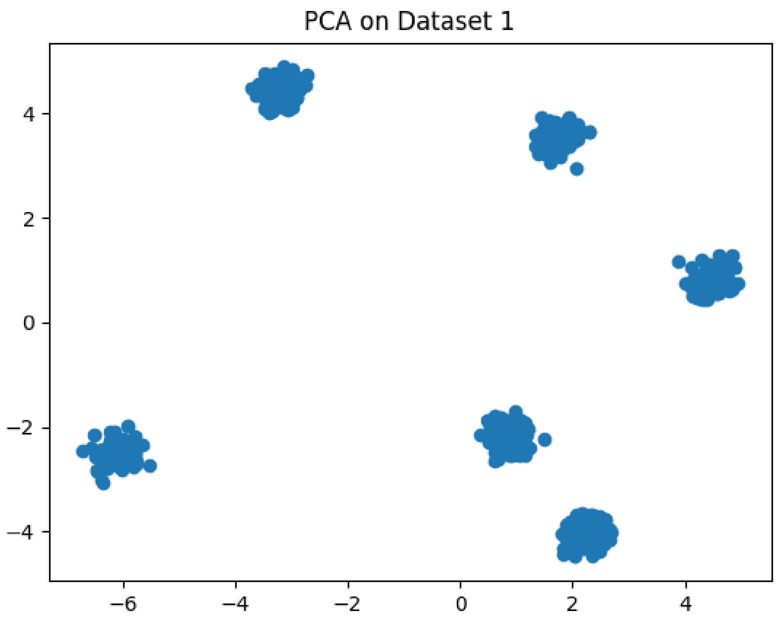


Figure 9: PCA on Dataset 1

- PCA performs well on Dataset 1, reducing its dimensionality to two principal components while preserving the underlying cluster structure.
- The six distinct clusters are clearly visible, which aligns with the findings from K Means and K-Medoids clustering using silhouette and loss scores.
- This clear cluster separation also validates the reliability of clustering results from both K Means and K-Medoids.

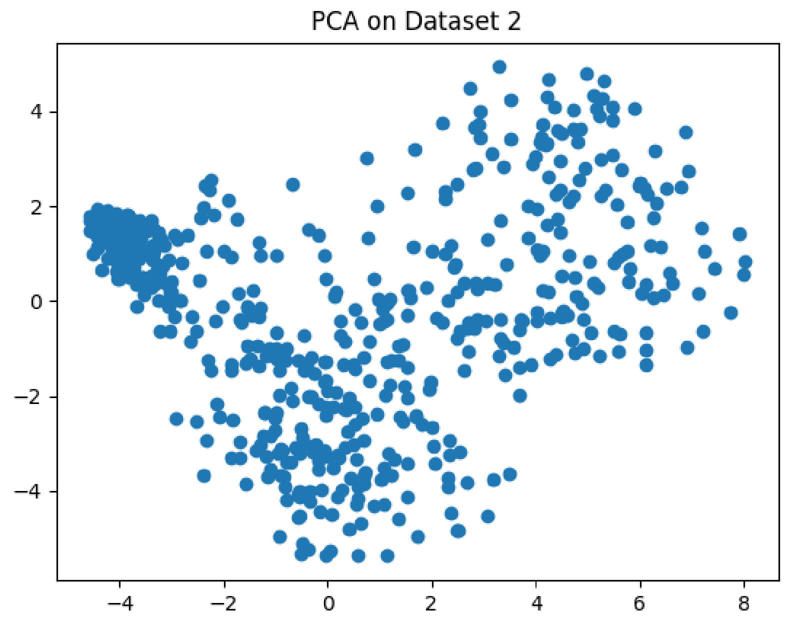


Figure 10: PCA on Dataset 2

- Unlike Dataset 1, the PCA projection for Dataset 2 appears noisy, with overlapping points and less distinct cluster separation.
- It looks like there are three clusters, but boundaries are not well defined. This shows also the difficulty encountered in clustering this dataset using K Means and K-Medoids.
- Further dimensionality reduction techniques such as t-SNE or UMAP can help to better separate the data, as these methods may handle non-linear relationships better than PCA.

2.4.2 Autoencoder

For autoencoder I have chosen following parameters as fixed. Learning rate = 0.01, iteration count = 500, number of hidden layers = 2, activation functions = ReLU optimizer utilized = Adam

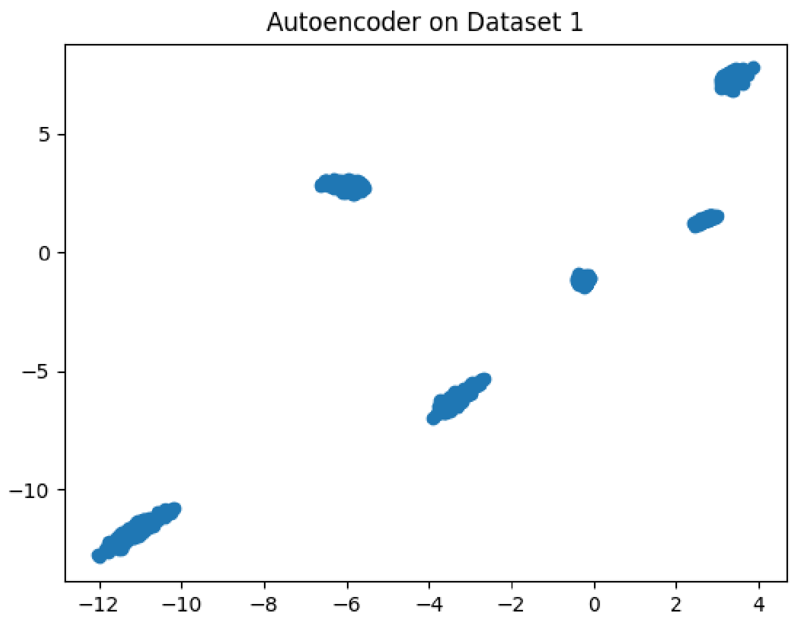


Figure 11: Autoencoder on Dataset 1

- The autoencoder effectively reduces Dataset 1 to a lower dimensional space while preserving the cluster structure.
- Similar to the PCA results, six distinct clusters are visible with well defined boundaries and minimal overlap, just like the findings from K Means and K-Medoids clustering methods.
- This result suggests that Dataset 1 is well suited for dimensionality reduction techniques, both linear (PCA) and non-linear (autoencoder).

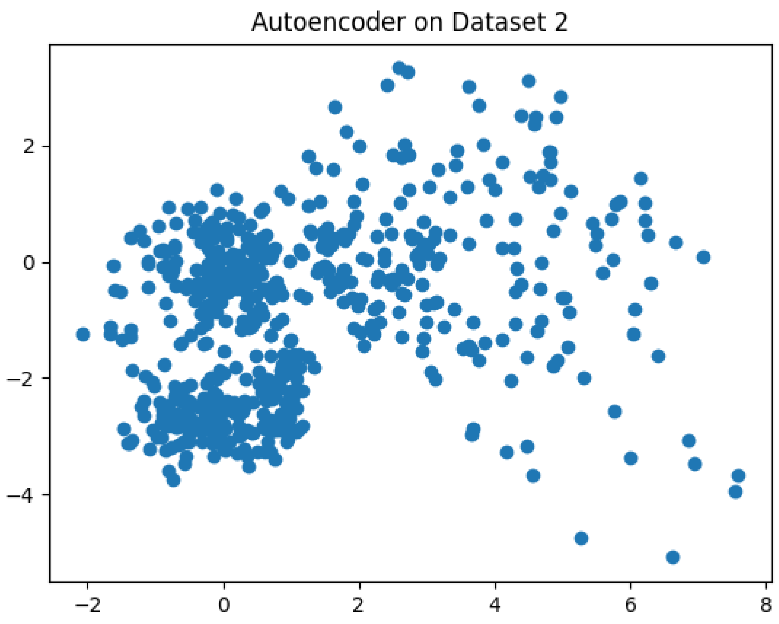


Figure 12: Autoencoder on Dataset 2

- Unlike Dataset 1, the autoencoder results for Dataset 2 are noisier, with overlapping points and less distinct cluster separation.
- While there is a hint of two denser regions, the boundaries are not clearly defined, making it difficult to interpret the underlying cluster structure.
- The high dimensionality of Dataset 2 (784 features) create challenges for clustering and dimensionality reduction techniques. Even with the autoencoder's non linear representation, the clusters remain ambiguous.

2.4.3 T-SNE

I experimented with different perplexity values (30 and 50), and both produced similar results, which are quite good. The clusters are distinct and well-separated in both cases.

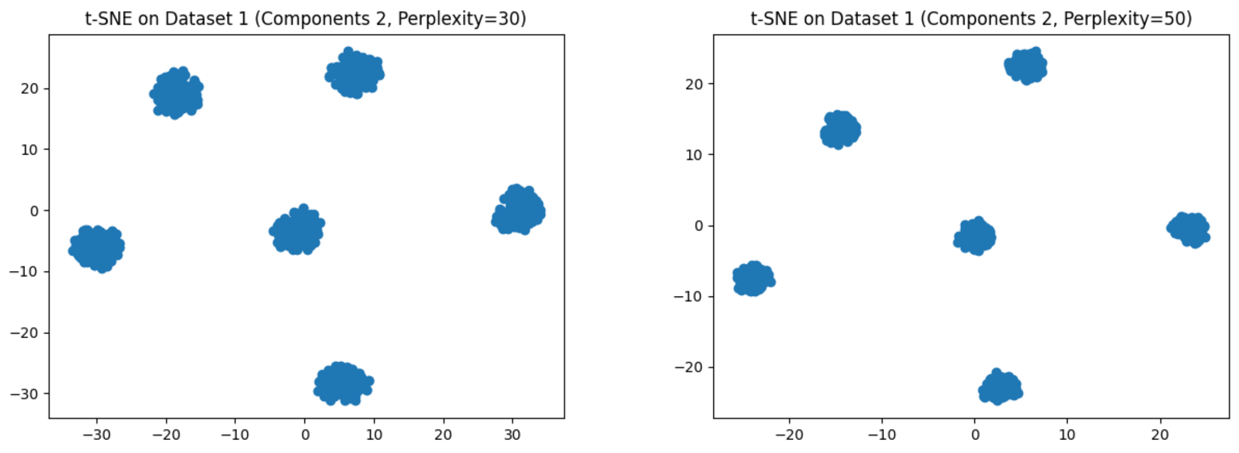


Figure 13: t-SNE on Dataset 1

- The t-SNE visualization with perplexity 30 and 50 shows six well-separated clusters, similar to PCA and autoencoder results for Dataset 1.
- This suggests that t-SNE is well-suited for visualizing the low-dimensional structure of Dataset 1 with this perplexity setting.

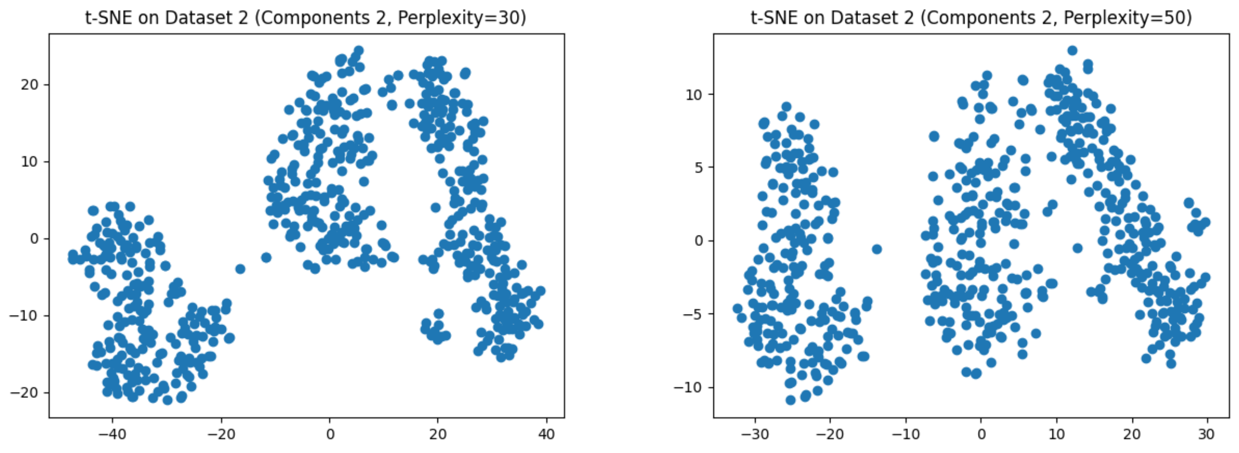


Figure 14: t-SNE on Dataset 2

- The t-SNE visualization for Dataset 2 with perplexity 30 and 50 suggests the presence of three clusters, though the boundaries are not as well-defined as in Dataset 1. On the other hand for Dataset 2 this method has given the best results so far.
- t-SNE outperforms PCA and autoencoders in visualizing Dataset 2 by providing better separation of clusters.

- t-SNE is a non-linear dimensionality reduction technique Unlike linear methods such as PCA. This allows t-SNE to uncover complex non-linear structures that linear methods might miss.

2.4.4 UMAP

I experimented with different distance metrics (cosine and euclidean). Similar to Dataset 1, there is no major difference between cosine and Euclidean distances for Dataset 2. While cosine distance might perform slightly better, the overall cluster structure remains consistent across both metrics.

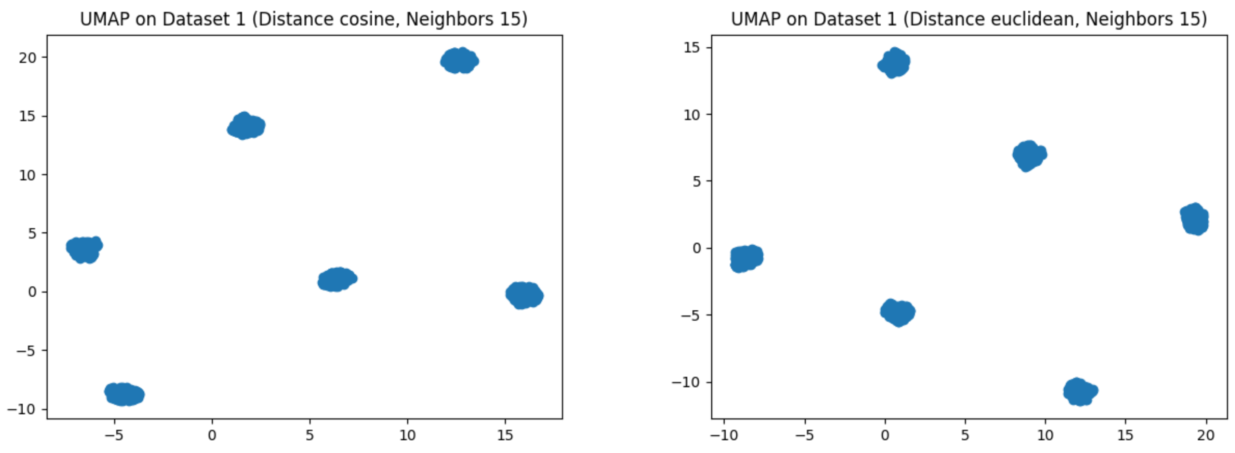


Figure 15: UMAP on Dataset 1

- The UMAP projection for Dataset 1 with cosine, euclidean distance and 15 neighbors shows six well-separated clusters with minimal overlap, similar to the results from t-SNE, PCA, and autoencoders.
- Both distance metrics (cosine and Euclidean) perform well for Dataset 1, indicating that UMAP is robust in preserving the inherent cluster structure of this dataset.

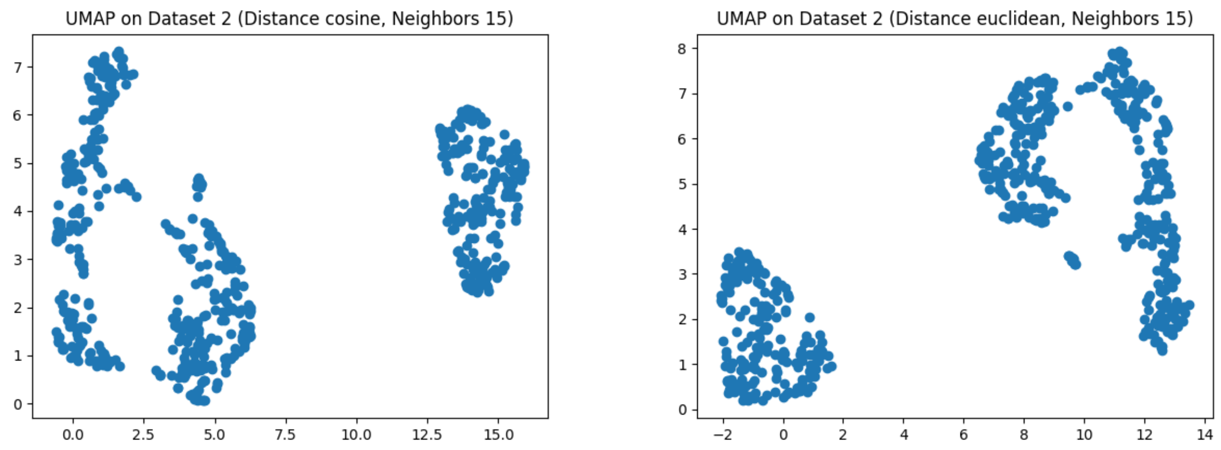


Figure 16: UMAP on Dataset 2

- The UMAP projection for Dataset 2 with both distance metrics reveals three distinct clusters, which is consistent with the observations from t-SNE for this dataset.
- The separation is relatively clear, but some points near the cluster boundaries exhibit overlap, indicating potential noise or complexity in the data. But this is not a big deal. We can say that UMAP generalizes well.
- Additionally compared to t-SNE, UMAP is faster and often provides similar results, making it a better choice for large datasets.

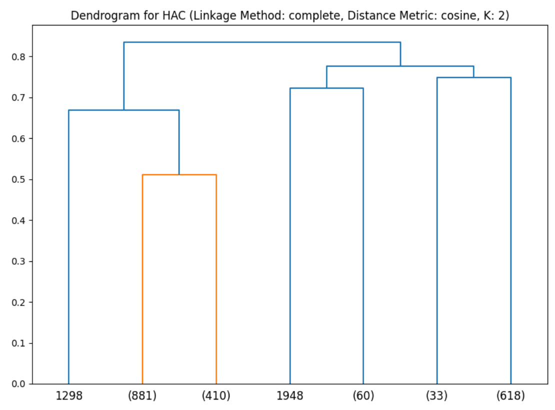
3 Part 3

3.1 HAC

In this part, I analyzed the performance of Hierarchical Agglomerative Clustering (HAC) using different configurations:

- Linkage Methods = [single, complete]
- Similarity Metric = [euclidean, cosine]

I think the generated plots, including dendrograms and K vs Silhouette score graphs, do not provide significant insights into the clustering quality due to low silhouette scores.



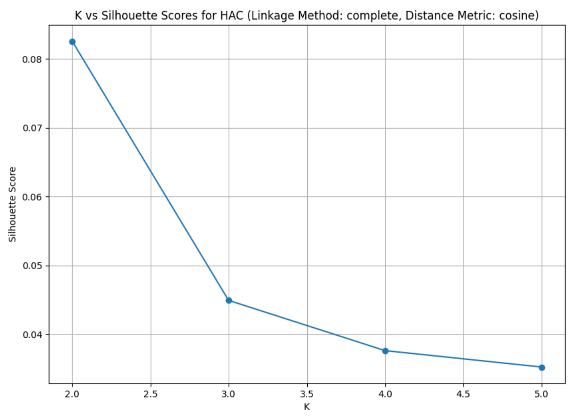
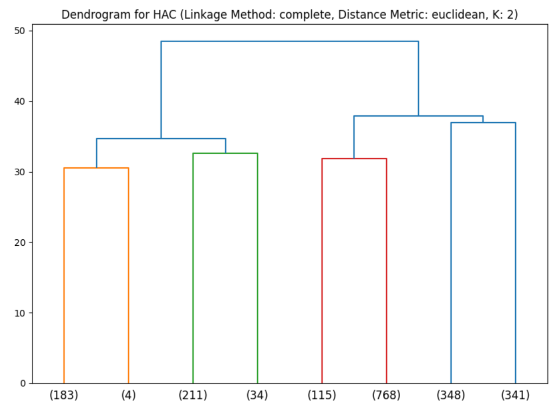


Figure 17: Dendogram and K vs Silhouette Plot for (Complete, Cosine)



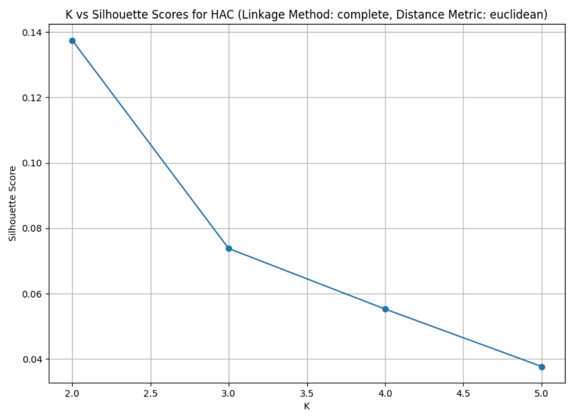


Figure 18: Dendogram and K vs Silhouette Plot for (Complete, Euclidean)

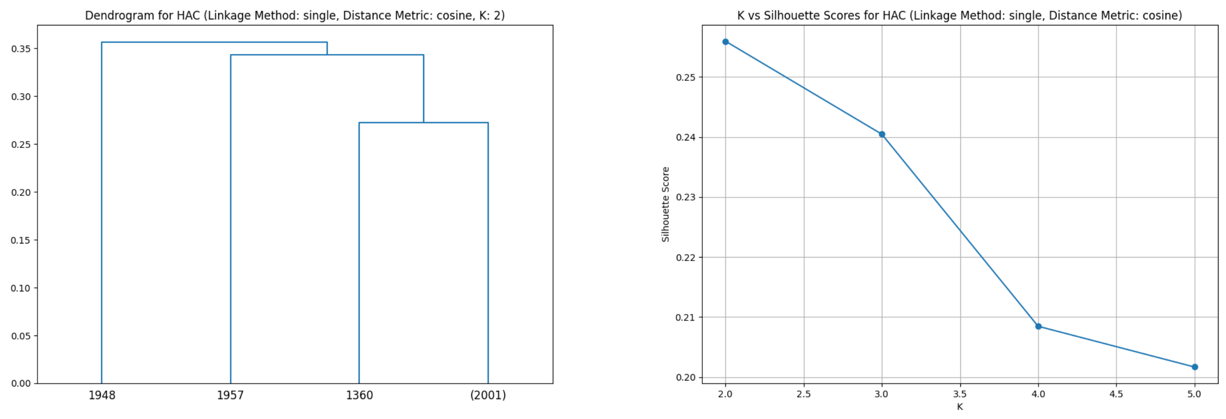


Figure 19: Dendogram and K vs Silhouette Plot for (Single, Cosine)

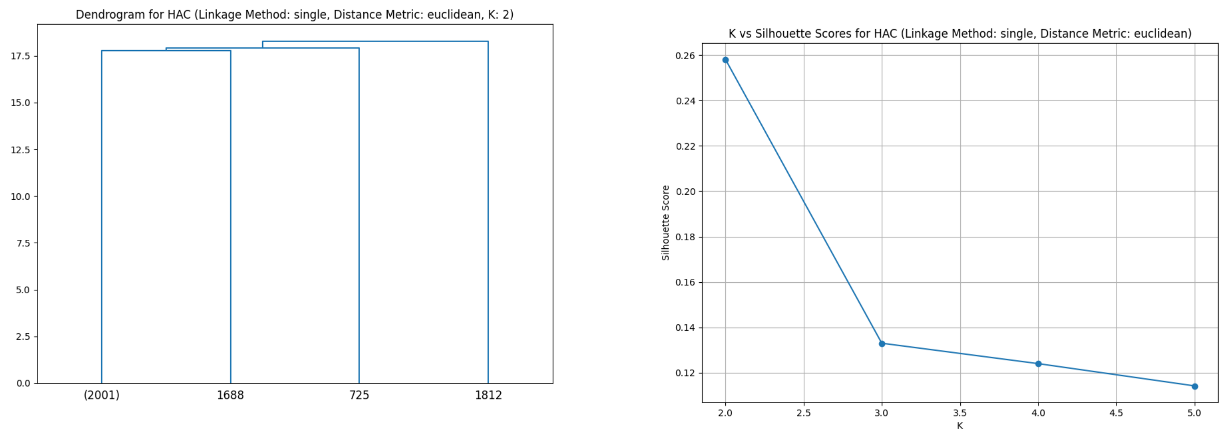


Figure 20: Dendogram and K vs Silhouette Plot for (Single, Euclidean)

3.1.1 Comments on Results

Among all Single Linkage Euclidean Distance was the best

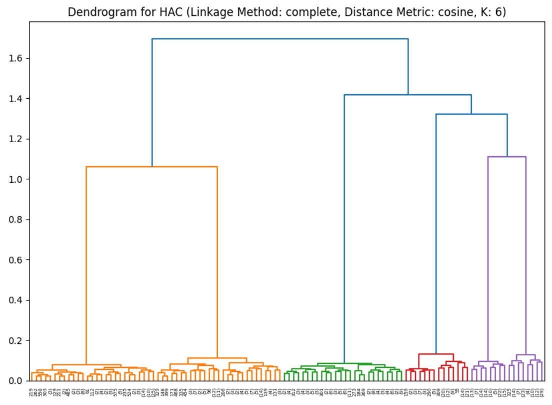
- Complete Linkage (Cosine and Euclidean):
  - The dendrograms show the hierarchical merging of clusters but do not provide clear, interpretable structures.
  - For $K = 2$, clusters are formed, but their sizes and merging points seem arbitrary, indicating potential issues with the data structure or algorithm suitability.
- Single Linkage (Cosine and Euclidean Distances):

  - The single linkage dendrograms show early merging of clusters. May be it is because of the chaining effect of single linkage method.
  - The clusters for $K = 2$ re not distinct or meaningful, suggesting that single linkage may not be suitable for this dataset.

- The choice of distance metric (cosine vs. Euclidean) does not significantly impact the visual interpretation of the dendrograms.
- Silhouette scores do not identify an optimal K value, as there is no peak or significant improvement for any K.

3.1.2 Optional: Test HAC Code with Dataset 1 from part 2

Since I suspected that my implementation was wrong, I tried my code with dataset 1 from part 2. As we observed in part 2, my code was able to find 6 clusters. In this case, I think my implementation is correct.



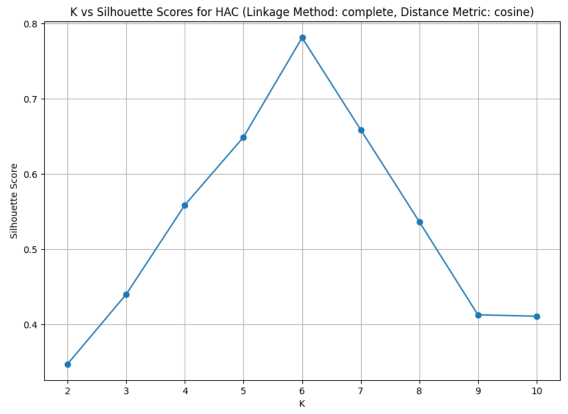


Figure 21: Dendogram and K vs Silhouette Plot for (Complete, Cosine)

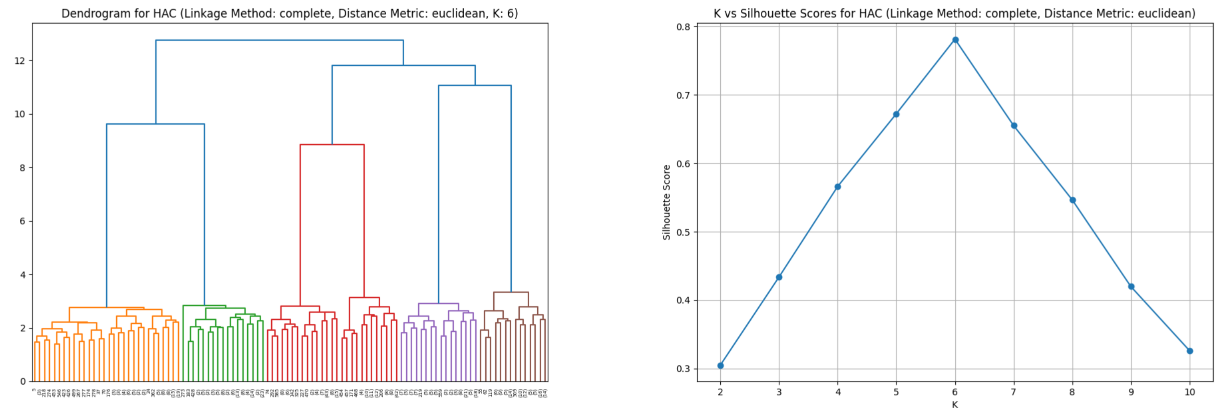


Figure 22: Dendogram and K vs Silhouette Plot for (Complete, Euclidean)

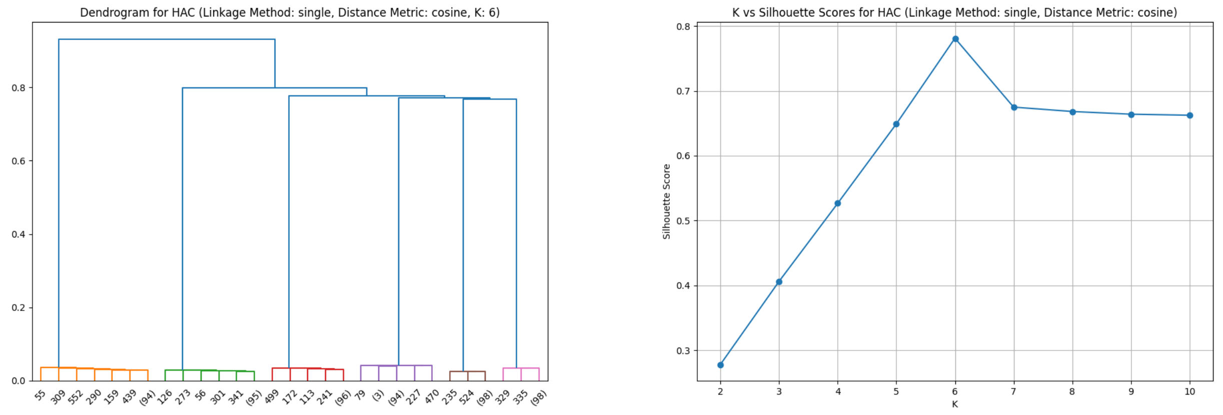


Figure 23: Dendogram and K vs Silhouette Plot for (Single, Cosine)

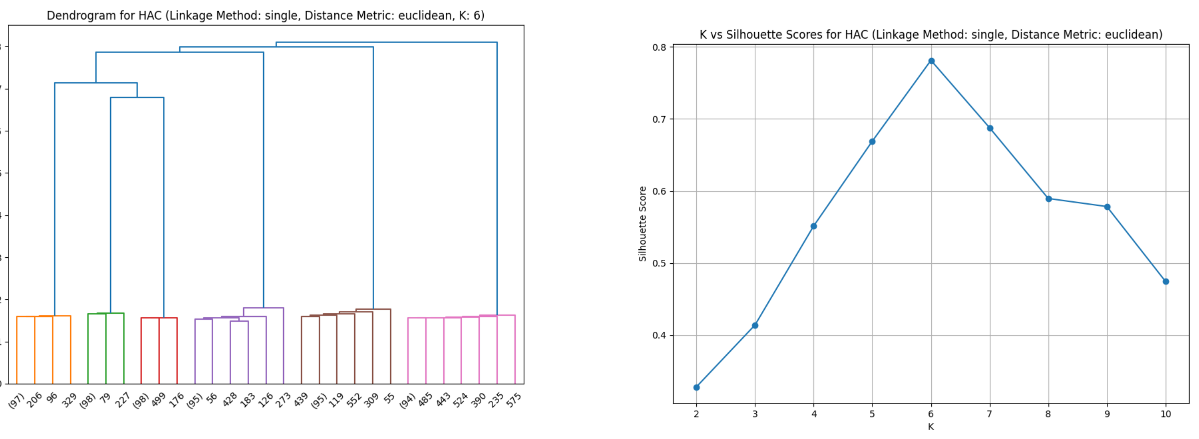


Figure 24: Dendogram and K vs Silhouette Plot for (Single, Euclidean)

3.2 DBSCAN

In this part, I analyzed the performance of DBSCAN using different configurations:

- Min sample values = [2, 5, 10, 15, 30]
- Euclidean Distance with [13, 14, 15, 16, 17, 18, 19, 20] epsilon values
- Cosine Distance with [0.1, 0.12, 0.14, 0.16, 0.18, 0.2] epsilon values

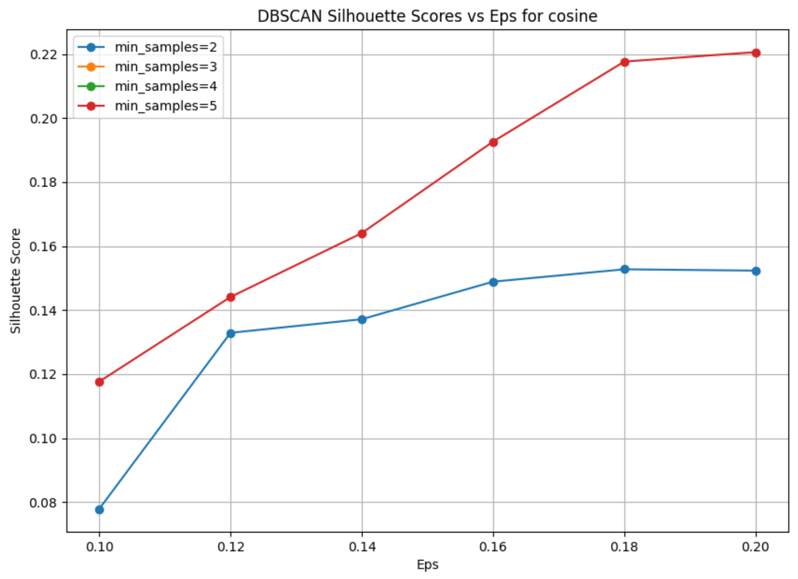


Figure 25: DBSCAN Silhouette Scores vs Eps for cosine

- As epsilon increases, the silhouette scores improve slightly, but they remain low overall (maximum around 0.22).
- Different min samples values (2–5) produce similar trends, with higher min samples slightly improving the silhouette scores.

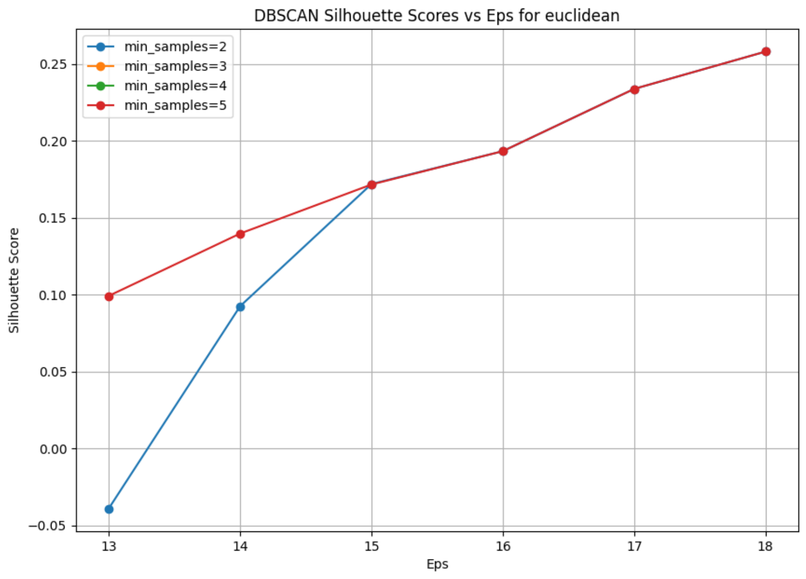


Figure 26: DBSCAN Silhouette Scores vs Eps for euclidean

- The scores remain consistent across different min samples values, indicating that min samples has a minor effect compared to epsilon.
- The highest score, achieved at epsilon=18.0 with min samples = 2, suggests that the clusters formed at this configuration are the most cohesive (although still suboptimal)

Best 4 results are followings

- Rank 1: eps=18.00, min samples=2, metric=euclidean, silhouette score=0.2582, K=2
- Rank 2: eps=18.00, min samples=5, metric=euclidean, silhouette score=0.2582, K=2
- Rank 3: eps=18.00, min samples=10, metric=euclidean, silhouette score=0.2582, K=2
- Rank 4: eps=18.00, min samples=15, metric=euclidean, silhouette score=0.2582, K=2

3.2.1 General Comments

- The consistently low silhouette scores across configurations confirm that the current hyperparameters are not optimal for the dataset.
- I tried many more hyperparameters but these are best ones among others.
- Epsilon significantly affects the results. Smaller epsilon values fail to group enough points into clusters, while larger values merge points into overly simplistic clusters, resulting in $K = 2$ clusters across all top configurations.

3.3 Dimensionality Reduction

For this part of the analysis, I experimented with various configurations across multiple dimensionality reduction techniques, including PCA, t-SNE, and UMAP. Despite exploring a wide range of hyperparameter settings, the results did not show consistently well-defined clusters based on silhouette scores. However, I will present and explain my best 4 results (ranked by silhouette scores) below.

Here is my best 4 results:

- KMeans on TSNE perplexity=30 with 2 clusters - Silhouette Score: 0.48
- HAC on TSNE perplexity=30 with 2 clusters - Silhouette Score: 0.47
- KMeans on UMAP number of neighbors=5 euclidean with 2 clusters - Silhouette Score: 0.47
- KMeans on UMAP number of neighbors=15 euclidean with 2 clusters - Silhouette Score: 0.47

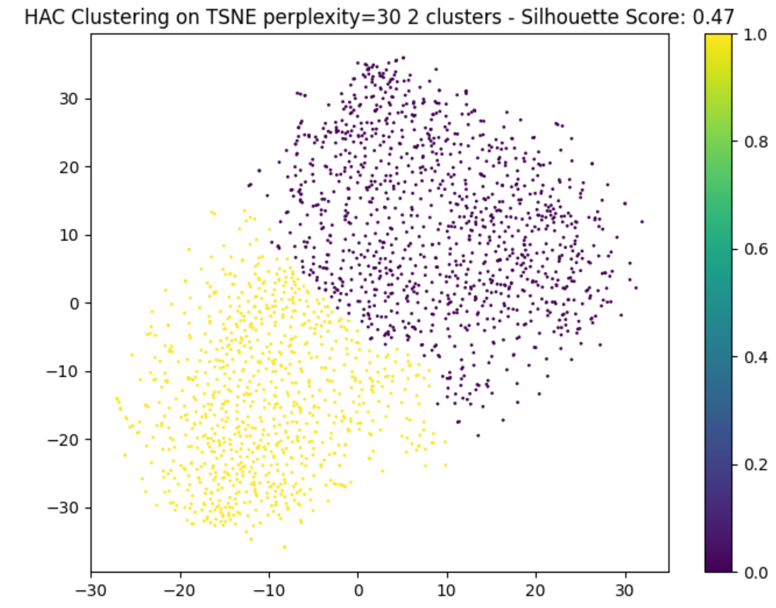


Figure 27: HAC TSNE

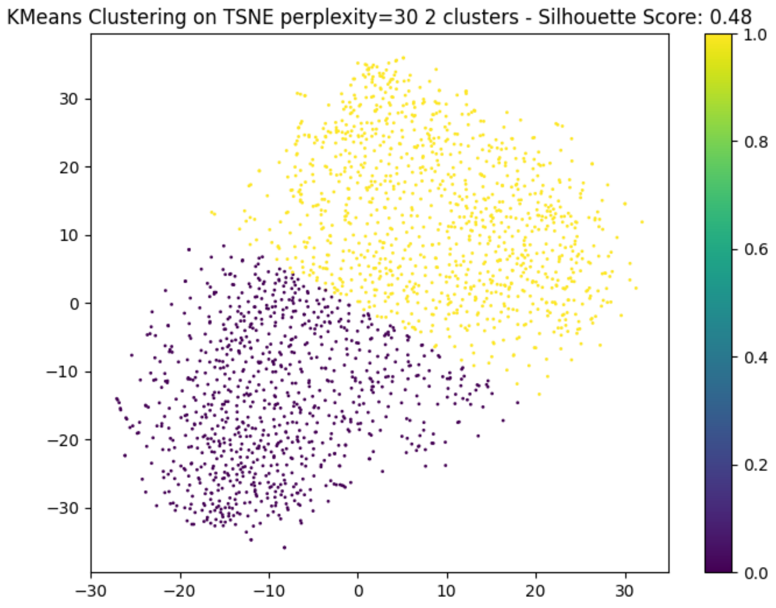


Figure 28: KMeans TSNE

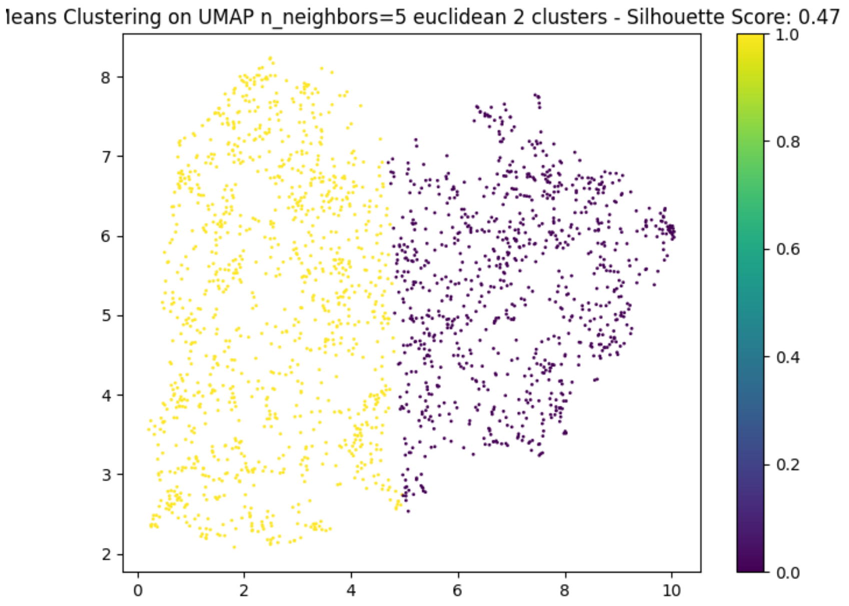


Figure 29: KMeans UMAP n neighbors=5

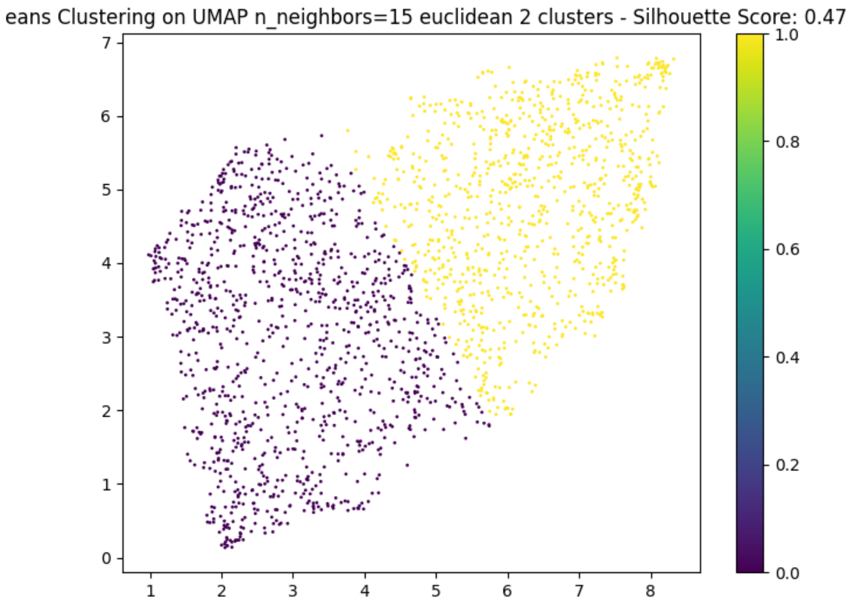


Figure 30: KMeans UMAP n neighbors=15

3.3.1 Comments

- While the silhouette scores for these configurations were the best among the experiments, they still indicate moderate clustering quality, suggesting that the optimal configuration might not have been found.
- The cluster shapes in the visualizations further indicate that the data might not naturally cluster into 2 groups or that the dimensionality reduction might not fully capture the cluster structure.
- The clustering results, even with the best configurations, do not appear to reflect the true underlying structure of the dataset. Further experimentation with different techniques or preprocessing steps, such as feature scaling, outlier removal, or using alternative metrics, could improve the clustering outcomes.

3.4 Worst-Case Runtime Analysis

3.4.1 Worst-Case Time Complexity of HAC

- **Distance Matrix Calculation:**
  - HAC starts by calculating pairwise distances between all N data points.
  - The complexity of this step is $\mathcal{O}(N^2 \cdot D)$, where D is the dimensionality of the data.
- **Merging Clusters:**
  - In each iteration, HAC merges the two closest clusters, updating the proximity matrix.
  - This requires $\mathcal{O}(N^2)$ operations for $N-1$ iterations.
- **Total Complexity:**
  - Combining both steps, the total complexity is $\mathcal{O}(N^2 \cdot D + N^3)$.
  - For high dimensional data, the $\mathcal{O}(N^2 \cdot D)$ term dominates.

3.4.2 Comparison of HAC and K Means for Large Datasets

Dataset Characteristics:

- **Number of Data Points (N):** 1 million which is 10^6
- **Dimension (D):** $120,000$ (200x200 RGB image).

K Means Time Complexity:

- We have computed K Means has a worst-case complexity of $\mathcal{O}(I \cdot N \cdot K \cdot D)$, where:
  - I: Number of iterations.
  - K: Number of clusters.

HAC Complexity for the Given Dataset:

- For $N = 10^6$ and $D = 120,000$, the worst-case runtime is:

$$\mathcal{O}(N^2 \cdot D + N^3) = \mathcal{O}(10^{12} \cdot D + 10^{18}),$$

  10^{18} dominates which is computationally very costly.

K Means Complexity for the Given Dataset:

- For $N = 10^6$, $D = 120,000$, the worst-case runtime is:

$$\mathcal{O}(I \cdot K \cdot D \cdot 10^6) = \mathcal{O}(I \cdot K \cdot 1.2 \times 10^{11}).$$

- While $I \cdot K \leq 10^7$ which is the typical case. I searched about this topic internet and found that big numbers for I and K is not practical in general. Dominating term around 10^11 to 10^15

3.4.3 Preferred Clustering Method

For a dataset with 1 million data points, each having 120,000 dimensions:

- **K Means:**
  - Preferred since it has linear dependency on N, which scales better than HAC's quadratic dependency.
- **HAC:**
  - Not feasible due to its quadratic dependency on N, which leads to longer computation times.