CENG499
INTRODUCTION TO MACHINE LEARNING
THE2
REPORT

Erdem
Şamlıoğlu
2448843

1 Part1

Model	K	Distance Function	Confidence Interval
1	1	Cosine	(0.892, 0.938)
2	1	Minkowski	(0.846, 0.900)
3	1	Mahalanobis	(0.836, 0.881)
4	3	Cosine	(0.913, 0.954)
5	3	Minkowski	(0.887, 0.927)
6	3	Mahalanobis	(0.885, 0.922)
7	5	Cosine	(0.921, 0.962)
8	5	Minkowski	(0.869, 0.908)
9	5	Mahalanobis	(0.877, 0.916)
10	7	Cosine	(0.933, 0.971)
11	7	Minkowski	(0.865, 0.909)
12	7	Mahalanobis	(0.860, 0.906)
13	10	Cosine	(0.913, 0.960)
14	10	Minkowski	(0.862, 0.911)
15	10	Mahalanobis	(0.849, 0.904)
16	15	Cosine	(0.875, 0.922)
17	15	Minkowski	(0.852, 0.898)
18	15	Mahalanobis	(0.841, 0.896)
19	30	Cosine	(0.817, 0.873)
20	30	Minkowski	(0.815, 0.868)
21	30	Mahalanobis	(0.756, 0.814)

Table 1: KNN Experiment Results

I have tried the different hyperparameter configurations, which for K=[1,3,5,7,10,15,30], distance functions=[cosine, minkowski, mahalanobis]. As it is stated in the hw pdf, I allocated the 20% of the dataset as a test set to evaluate the generalization ability of the selected model. The remaining 80% is utilized for training and hyperparameter tuning using 10 fold stratified cross validation.
I ran my code multiple times to observe best performing hyperparameter, since the values slighlty differ from run to run due to the random initialization of the dataset shuffling and splitting in the cross validation process. Best configuration is with k=7 and with distance function as Cosine. The mean accuracies and confidence intervals for k=5 and 7 (both distance function cosine) were close so I also used f1 score, which was better for k=7 consistently , so I have chosen k=7. For the best hyperparameter configuration, the generalization performance was 96.7%.

2 Part2

2.1 Kmeans

I have used the scikitlearn implementation for K means method and silhouette score calculations as it is stated in the assignment pdf. I have used confidence intervals to see if the results were consistent.

2.2 Dataset1

K	Loss Mean	Loss CI	Silhouette Mean	Silhouette CI
2	17716.439	±280.868	0.324	±0.008
3	12766.717	±287.577	0.429	±0.007
4	7830.688	±168.392	0.559	±0.006
5	4181.870	±280.031	0.664	±0.008
6	1010.421	±0.000	0.781	±0.000
7	999.038	±0.973	0.661	±0.002
8	987.923	±1.556	0.563	±0.030
9	979.602	±2.266	0.468	±0.035
10	965.447	±2.242	0.369	±0.036

Table 2: Dataset1

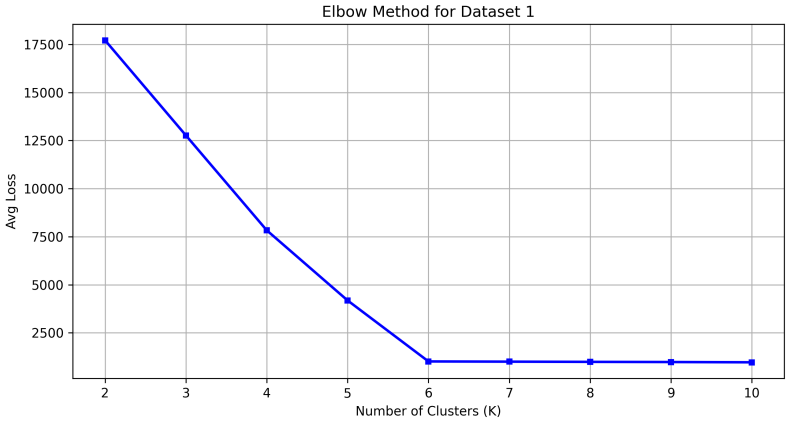

Figure 1: Kmeans Dataset 1 Elbow Method

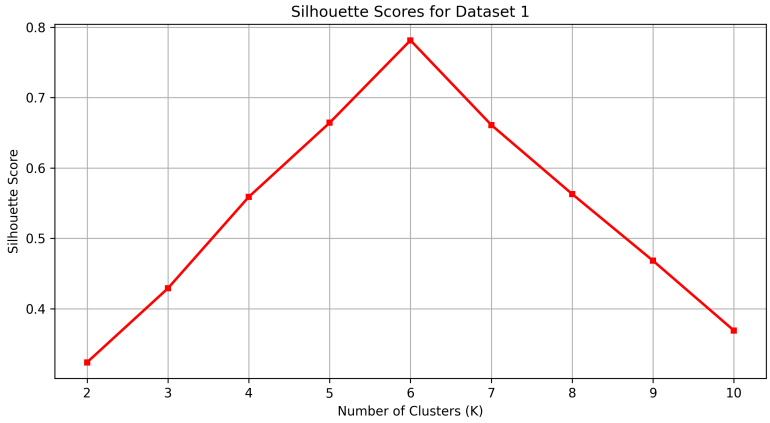

Figure 2: Kmeans Dataset 1 Silhouette Method

For dataset 1, the elbow method shows that the best k is 6. The loss curve shows a sharp decline until 6 and then the decrease isn't as much. Also the highest silhouette score is observed for K=6 with value 0.781. Both the silhouette and loss values' confidence interval is 0.

2.3 Dataset2

K	Loss Mean	Loss CI	Silhouette Mean	Silhouette CI
2	26046.084	±45.645	0.194	±0.009
3	22994.479	±0.412	0.167	±0.000
4	21622.326	±170.056	0.166	±0.009
5	20434.946	±231.719	0.144	±0.013
6	19504.350	±147.385	0.143	±0.007
7	18786.513	±85.976	0.131	±0.007
8	18307.103	±92.666	0.130	±0.006
9	17857.377	±56.527	0.126	±0.007
10	17520.718	±41.003	0.129	±0.006

Table 3: Dataset2

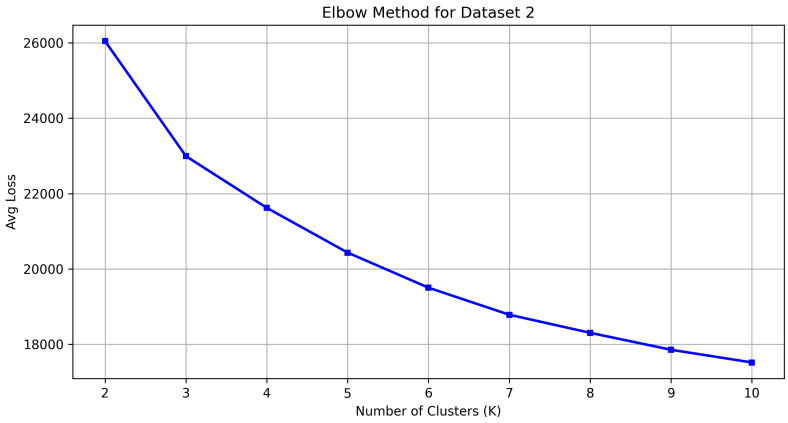

Figure 3: Kmeans Dataset 2 Elbow Method

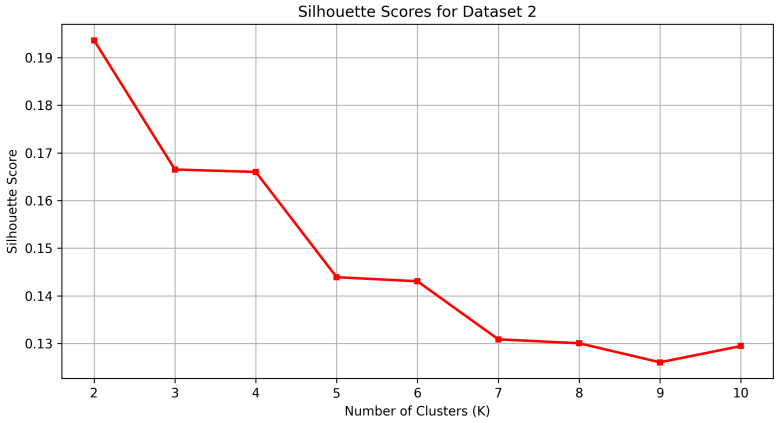

Figure 4: Kmeans Dataset 2 Silhouette Method

For dataset2, decrease slows after k=3, but the highest silhouette score is at k=2, at these points the confidence intervals are also high compared to dataset1 showing higher variability performance compared to dataset1. The elbow method showing k as 3, but silhouette as 2, might be due to noisy data or overlapping.

2.4 Kmedoids

I have used the scikitlearn implementation for K medoids method, silhouette score calculations and used Cosine similarity as the distance metric as it is stated in the assignment pdf.

2.5 Dataset1

K	Loss Mean	Loss CI	Silhouette Mean	Silhouette CI
2	376.845	±0.000	0.384	±0.000
3	284.587	±0.000	0.472	±0.000
4	184.493	±0.000	0.620	±0.000
5	100.722	±0.000	0.746	±0.000
6	19.453	±0.000	0.954	±0.000
7	19.296	±0.000	0.807	±0.000
8	19.159	±0.000	0.659	±0.000
9	18.984	±0.000	0.657	±0.000
10	18.673	±0.000	0.648	±0.000

Table 4: Dataset 1

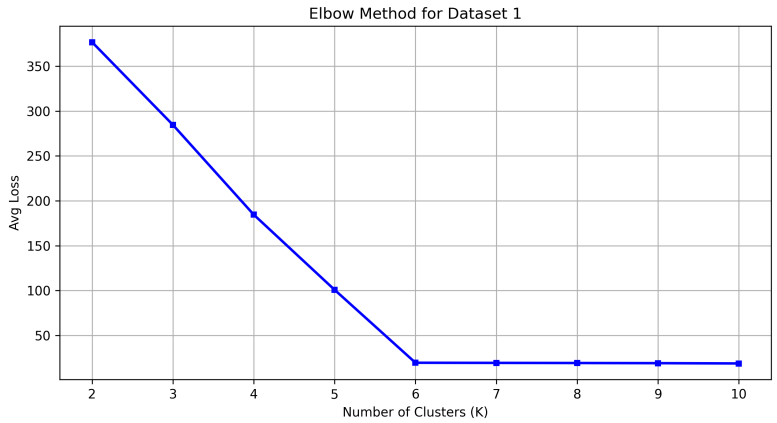

Figure 5: Kmedoids Dataset 1 Elbow Method

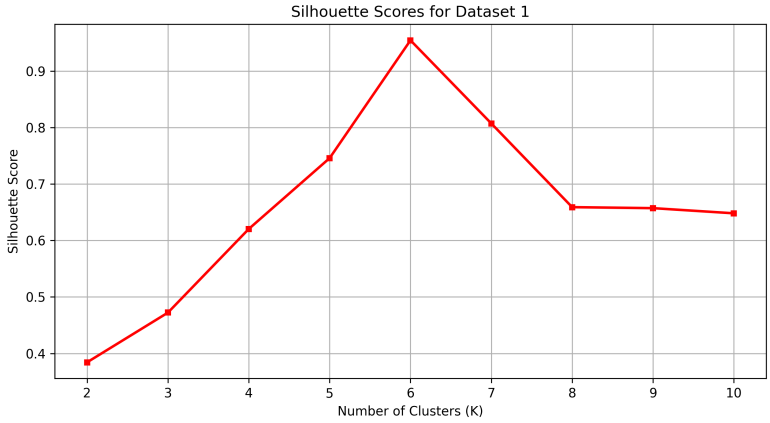

Figure 6: Kmedoids Dataset 1 Silhouette Method

For dataset1, with the elbow method, decrease slows a lot after k=6, and for the silhouette score, at k=6 it has the highest value. Based on both of these, k=6 is the best number of clusters for dataset1.

2.6 Dataset2

K	Loss Mean	Loss CI	Silhouette Mean	Silhouette CI
2	496.794	±0.000	0.112	±0.000
3	352.596	±0.000	0.233	±0.000
4	305.870	±0.000	0.274	±0.000
5	299.963	±0.000	0.220	±0.000
6	281.180	±0.000	0.189	±0.000
7	275.242	±0.000	0.186	±0.000
8	265.805	±0.000	0.191	±0.000
9	256.511	±0.000	0.197	±0.000
10	247.547	±0.000	0.195	±0.000

Table 5: Dataset2

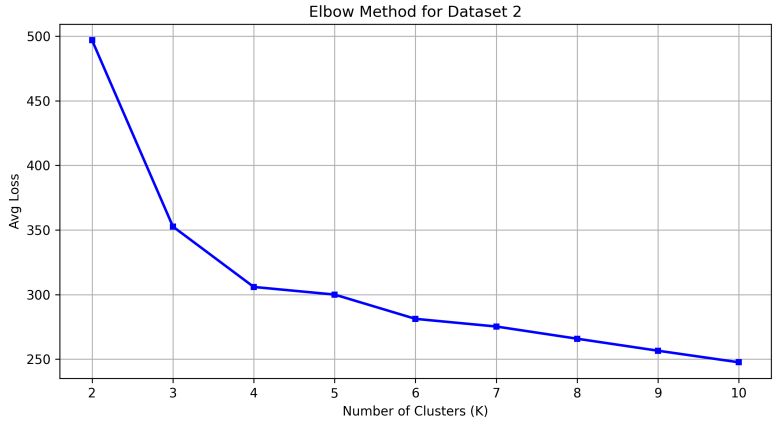

Figure 7: Kmedoids Dataset 2 Elbow Method

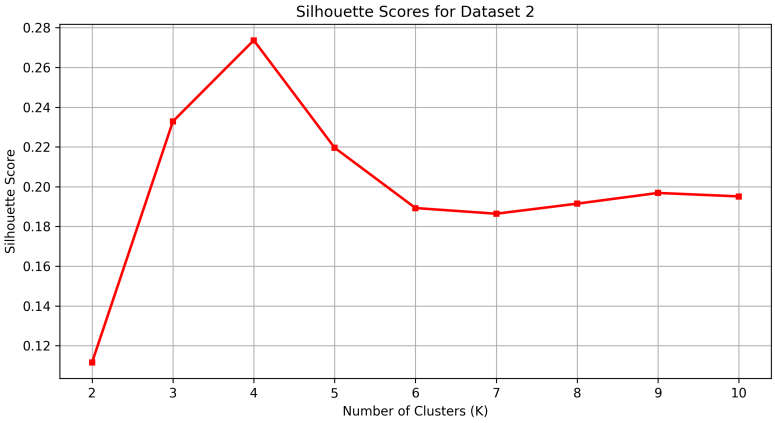

Figure 8: Kmedoids Dataset 2 Silhouette Method

For dataset2, with the elbow method, there are 2 points which the decrease get slower, k=3 and k=4 (after 4 it is slower) and checking the silhouette score, it is highest at k=4, based on k=4 is the best number of clusters for dataset2.

For an overall comparison of dataset 1 and 2, for kmedoids, silhouette score for dataset1 is much higher than dataset2, showing dataset1 has a more distinct separated well enough structure than dataset2.

For the confidence intervals, the value is 0 for both metrics loss and silhouette score, which is expected since K medoids is deterministic.

2.7 Dimensionality Reduction

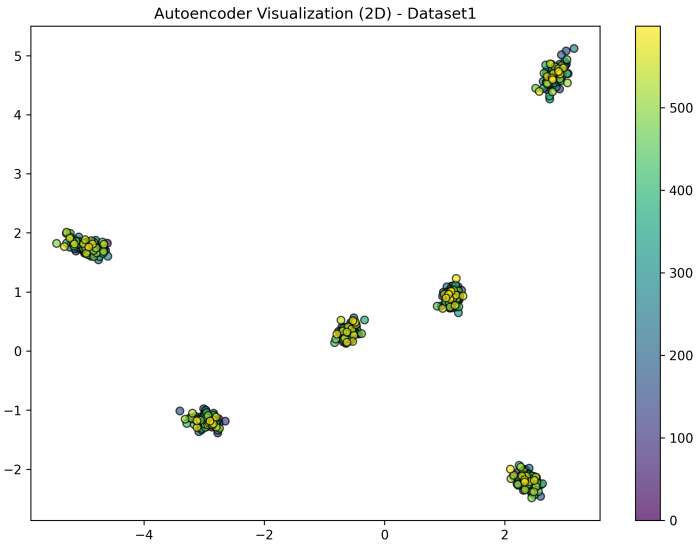

Figure 9: Dataset 1: Autoencoder 2D Visualization

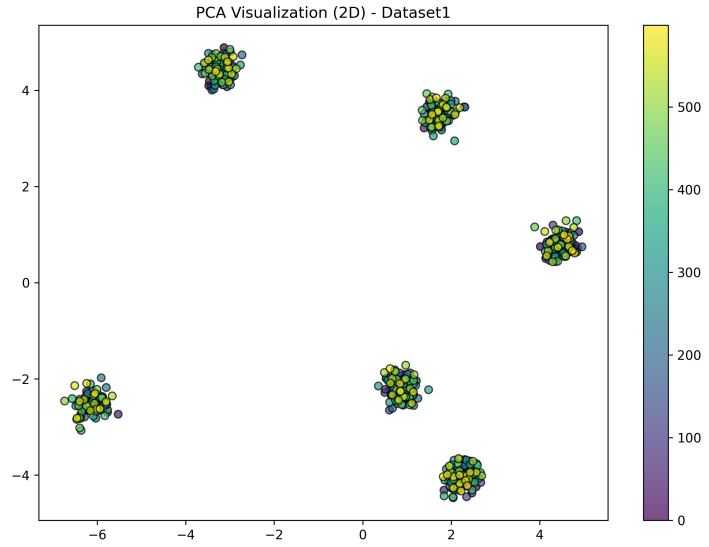

Figure 10: Dataset 1: PCA 2D Visualization

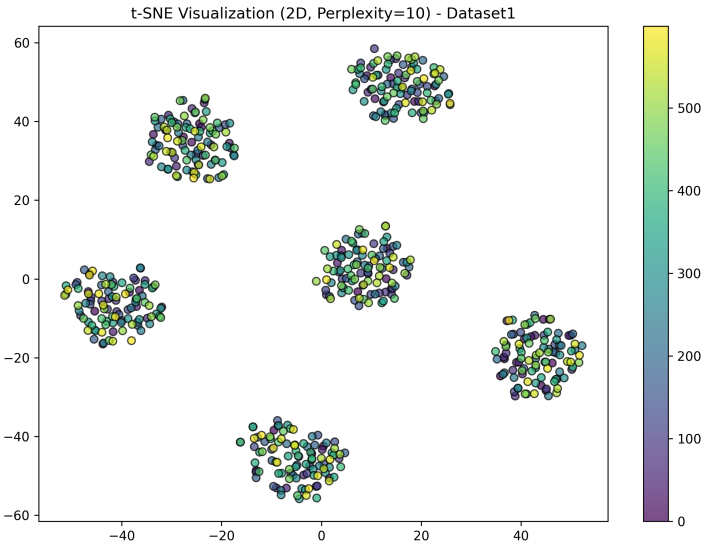

Figure 11: Dataset 1: t-SNE 2D Visualization (Perplexity = 10)

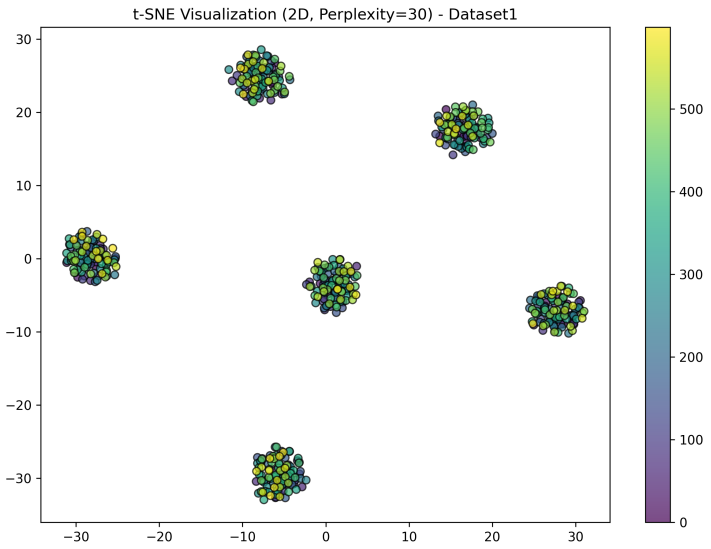

Figure 12: Dataset 1: t-SNE 2D Visualization (Perplexity = 30)

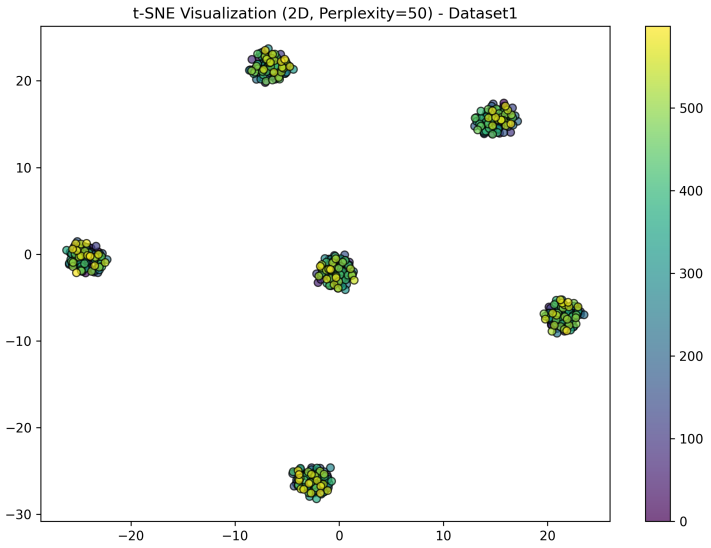

Figure 13: Dataset 1: t-SNE 2D Visualization (Perplexity = 50)

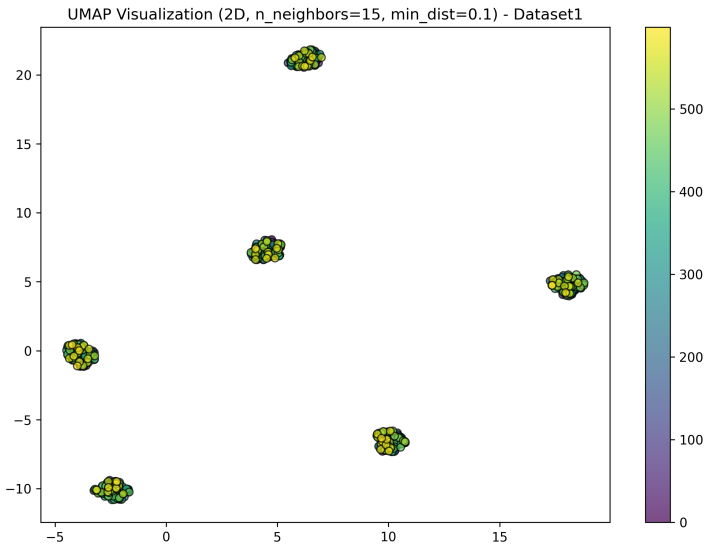

Figure 14: Dataset 1: UMAP 2D Visualization (Neighbors = 15, Min Distance = 0.1)

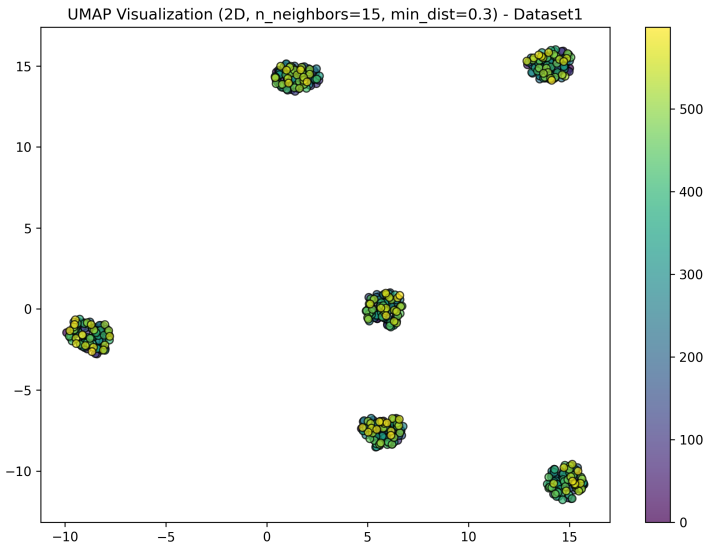

Figure 15: Dataset 1: UMAP 2D Visualization (Neighbors = 15, Min Distance = 0.3)

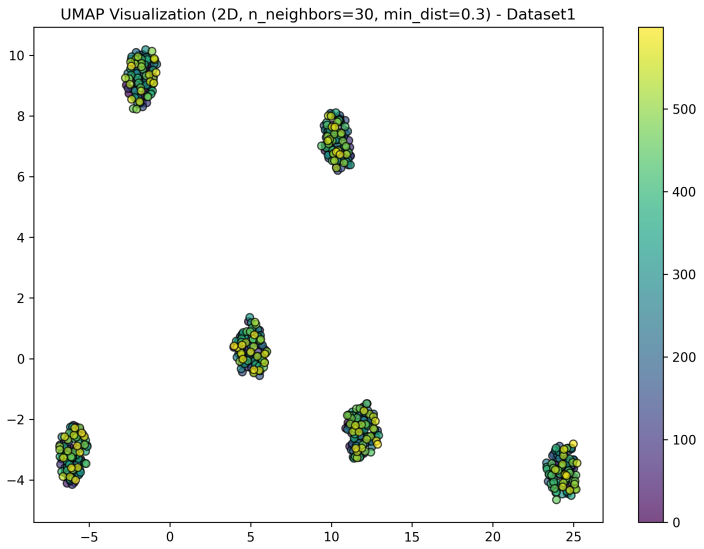

Figure 16: Dataset 1: UMAP 2D Visualization (Neighbors = 30, Min Distance = 0.3)

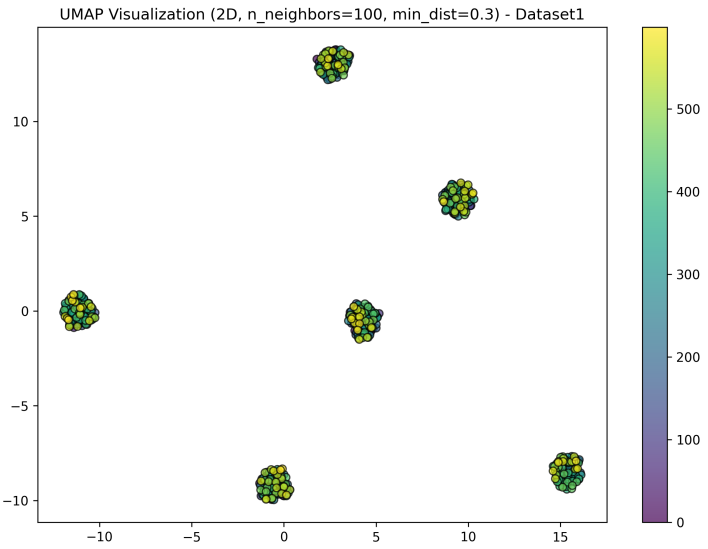

Figure 17: Dataset 1: UMAP 2D Visualization (Neighbors = 100, Min Distance = 0.3)

Z

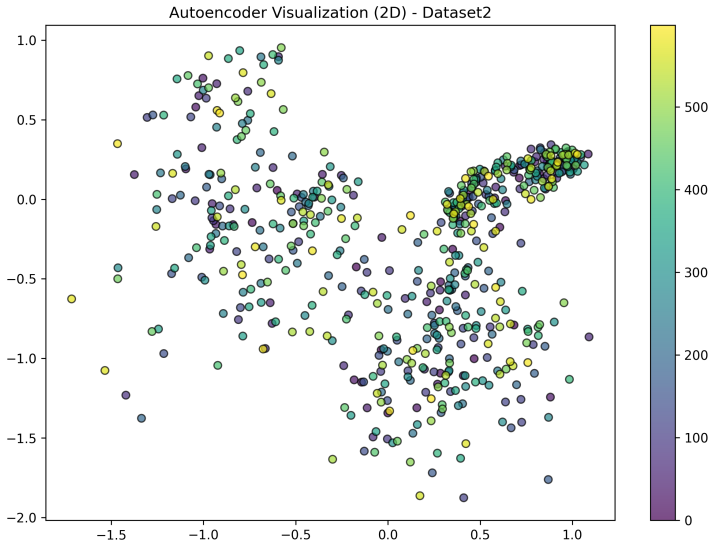

Figure 18: Dataset 2: Autoencoder 2D Visualization

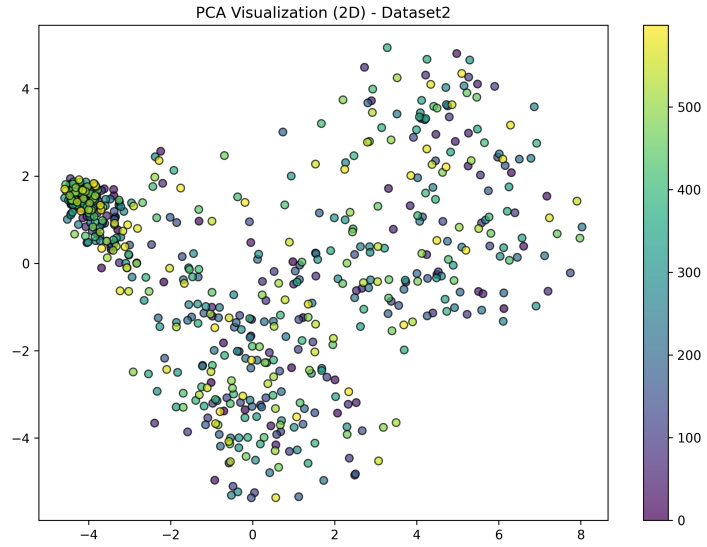

Figure 19: Dataset 2: PCA 2D Visualization

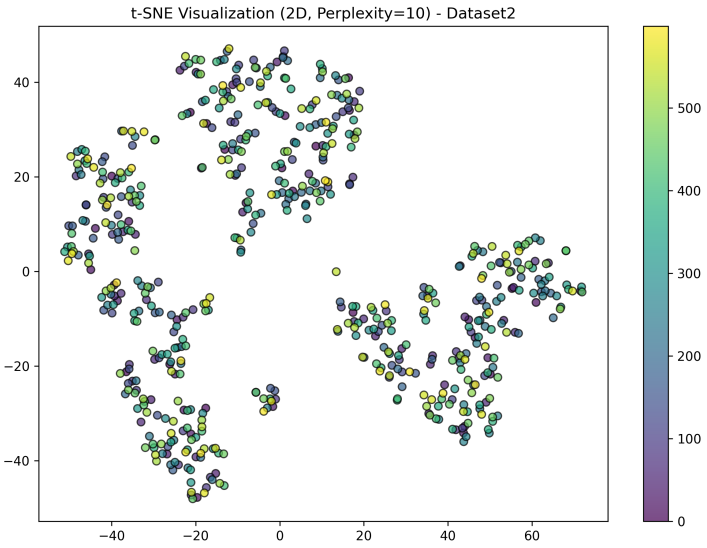

Figure 20: Dataset 2: t-SNE 2D Visualization (Perplexity = 10)

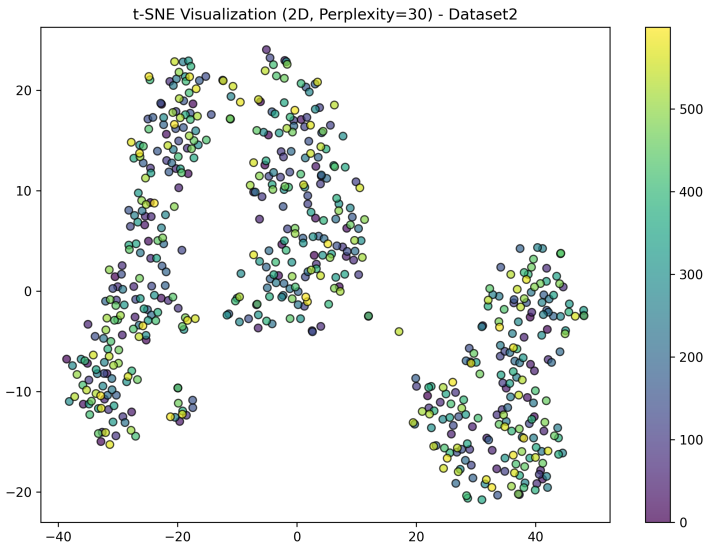

Figure 21: Dataset 2: t-SNE 2D Visualization (Perplexity = 30)

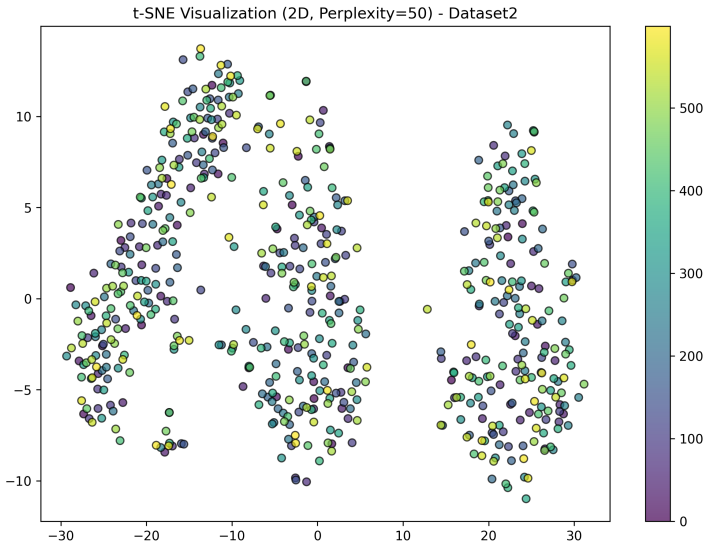

Figure 22: Dataset 2: t-SNE 2D Visualization (Perplexity = 50)

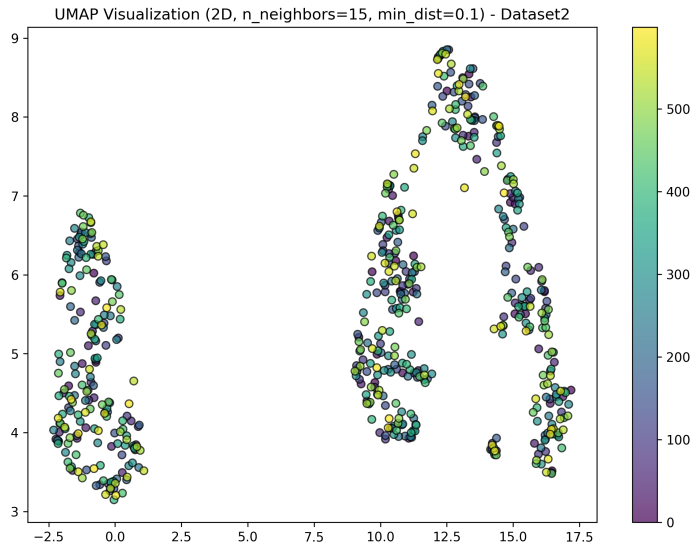

Figure 23: Dataset 2: UMAP 2D Visualization (Neighbors = 15, Min Distance = 0.1)

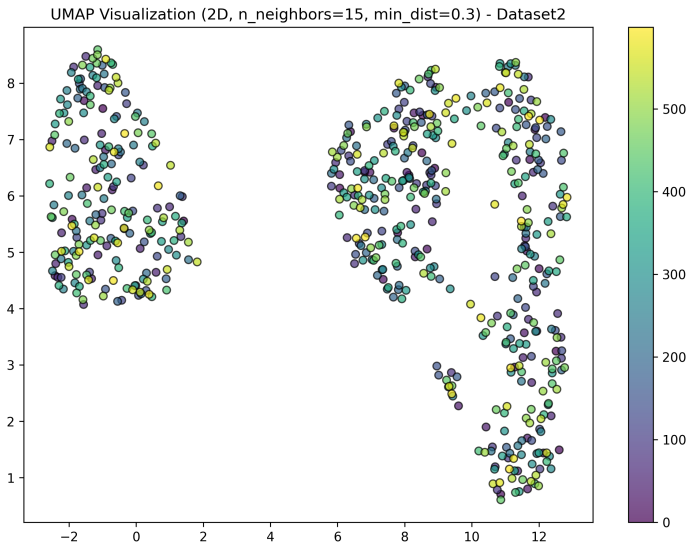

Figure 24: Dataset 2: UMAP 2D Visualization (Neighbors = 15, Min Distance = 0.3)

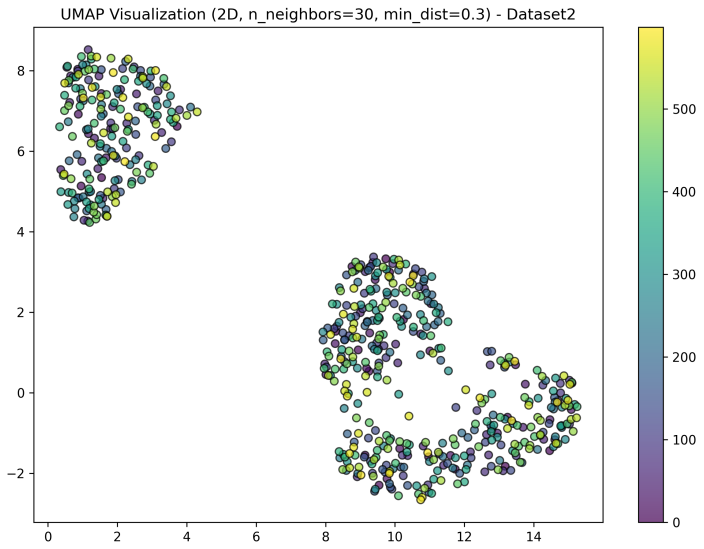

Figure 25: Dataset 2: UMAP 2D Visualization (Neighbors = 30, Min Distance = 0.3)

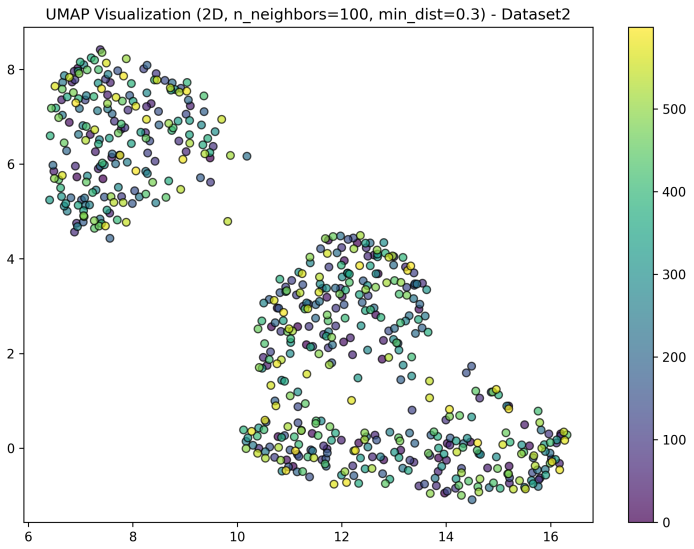

Figure 26: Dataset 2: UMAP 2D Visualization (Neighbors = 100, Min Distance = 0.3)

I implemented the PCA and Autoencoders myself and used scikitlearn implementation for t-SNE and umap library for UMAP. The dimensionality reduction is done to visualize the dataset and understand the structure of the clusters in lower dimensional space. I have tried several configurations including 2D and 3D. (3D didnt change the visuals significantly so I didn't include them in the report.) For t-SNE perplexity values I have tested [10, 30, 50], for umap n neighbors I tested [10, 15, 30, 100] and for min dist I tested [0.1, 0.3, 0.5] values.(Again I have included not all of them but some of them to compare.)

For every dimensionality reduction method, they performed better on dataset 1 then dataset 2.

For t-SNE's different values of perplexity, higher values separated the data better than lower.

for UMAP, smaller n neighbors values, and lower min dist values separated the data better.

For the best dimensionality reduction results for dataset1, all of them performs good, and specifically. For Autoencoder, Figure 9. For PC, Figure 10. For t-SNE, Figure 13. For UMAP figure 14. We get 6 clusters for dataset1 which was the expected result.

For the best dimensionality reduction results for dataset2, it is hard to get a "best" since data isn't separated well enough like dataset1. But for autoencoder, figure 18. For PCA, figure 19. For t-SNE, figure 22. For UMAP, figure 23. We get 3(?) clusters from dataset 2 which wasn't expected. Expected cluster number was 4.

2.8 Runtime

The number of data points (N), data sample vector dimension (d), cluster number(K), and the number of iterations (I)

Kmeans time complexity= $O(I \cdot N \cdot K \cdot d)$
Kmedoids time complexity= $O(I \cdot N^2 \cdot K \cdot d)$ (I use kmedoids++ as parameter in scikitlearn implementation)

3 Part 3

3.1 HAC and DBSCAN

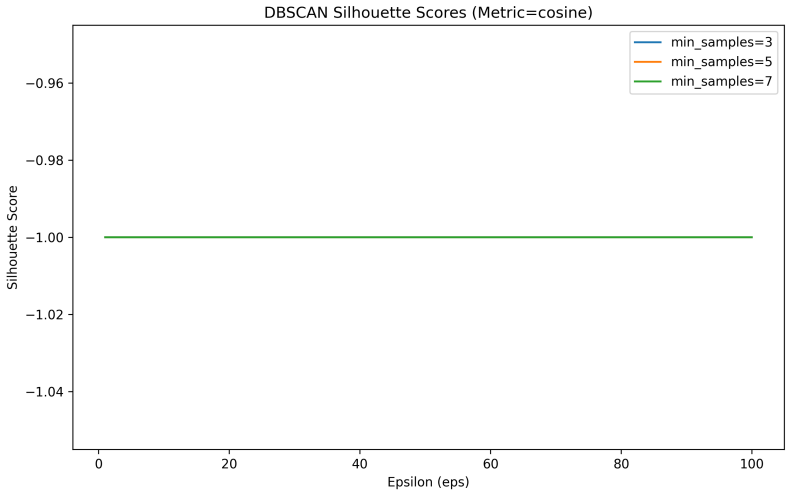

Figure 1: DBSCAN: Silhouette (Cosine Distance)

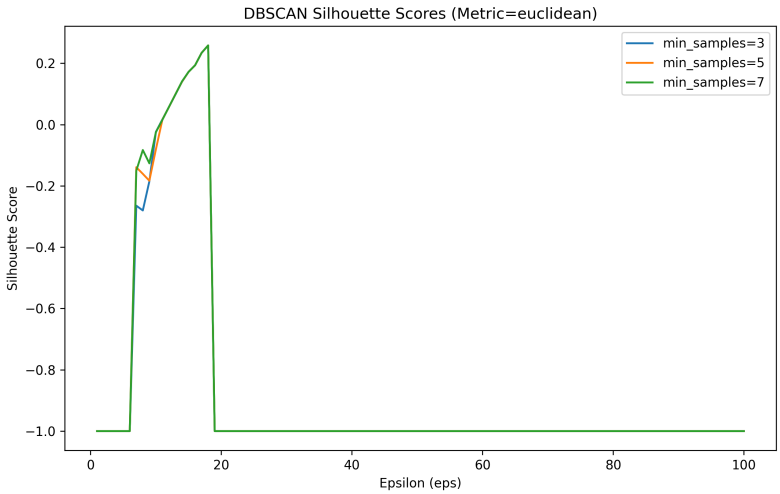

Figure 2: DBSCAN: Silhouette (Euclidean Distance)

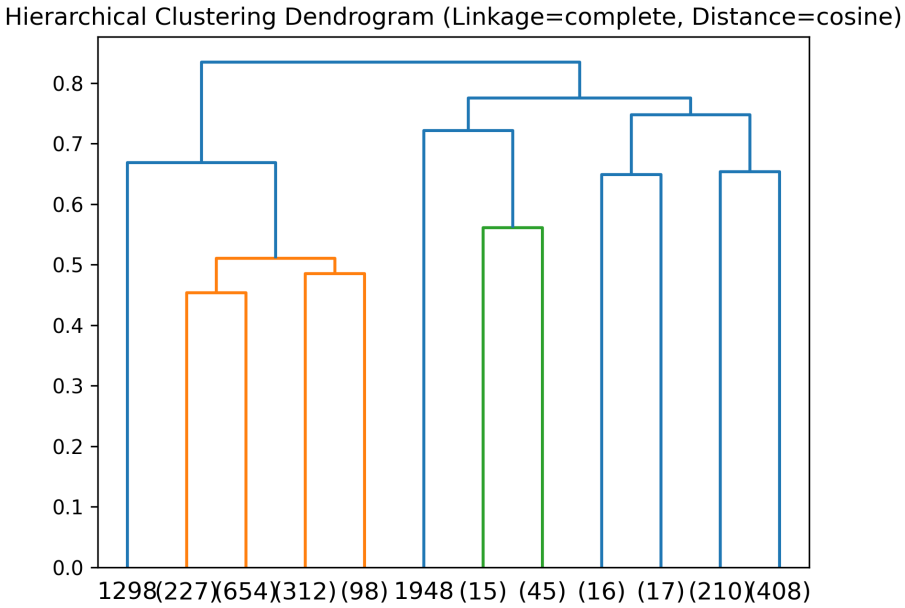

Figure 3: Dendrogram: Complete Linkage (Cosine Distance)

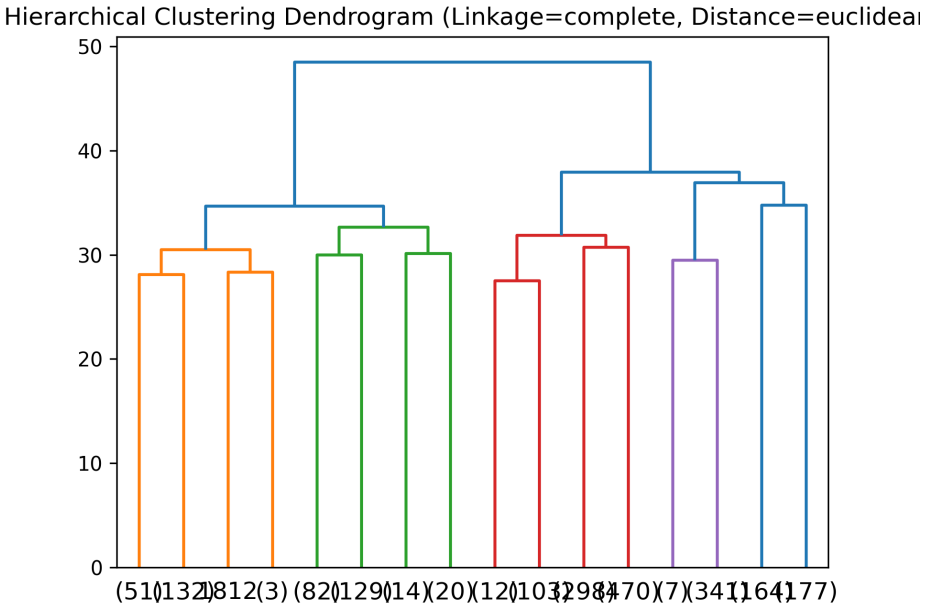

Figure 4: Dendrogram: Complete Linkage (Euclidean Distance)

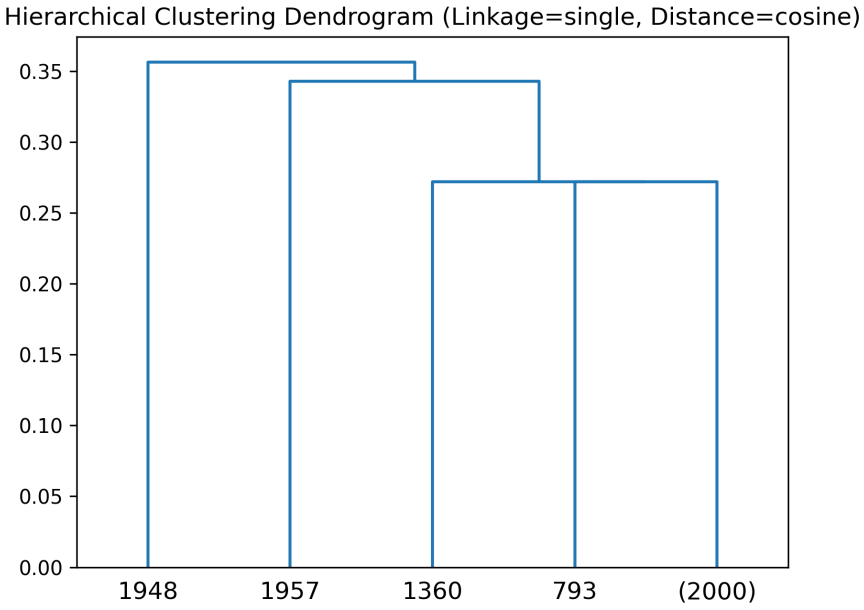

Figure 5: Dendrogram: Single Linkage (Cosine Distance)

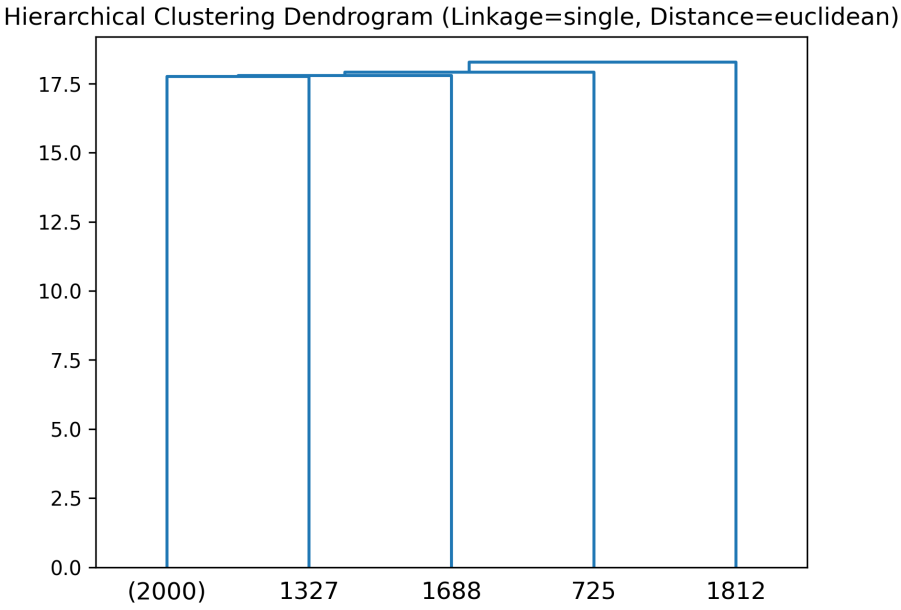

Figure 6: Dendrogram: Single Linkage (Euclidean Distance)

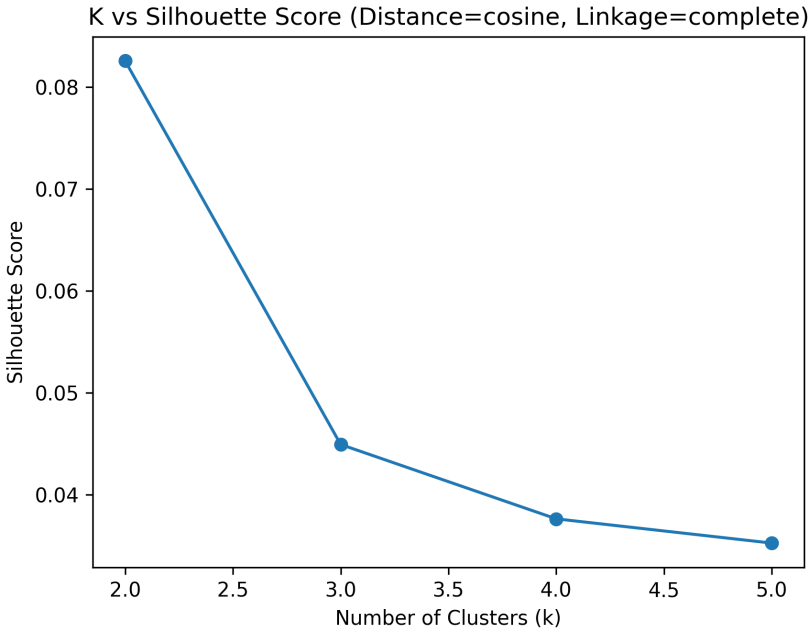

Figure 7: K vs Silhouette: Cosine Distance (Complete Linkage)

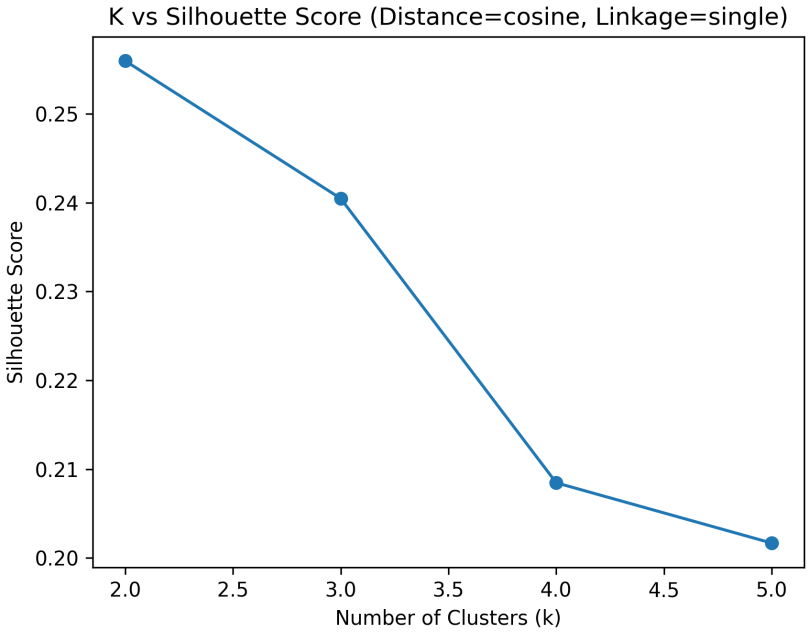

Figure 8: K vs Silhouette: Cosine Distance (Single Linkage)

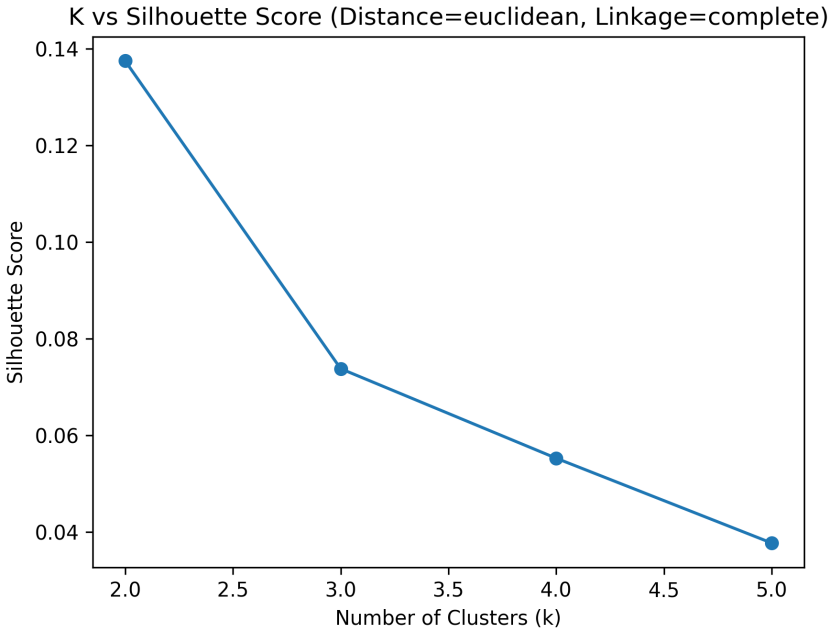

Figure 9: K vs Silhouette: Euclidean Distance (Complete Linkage)

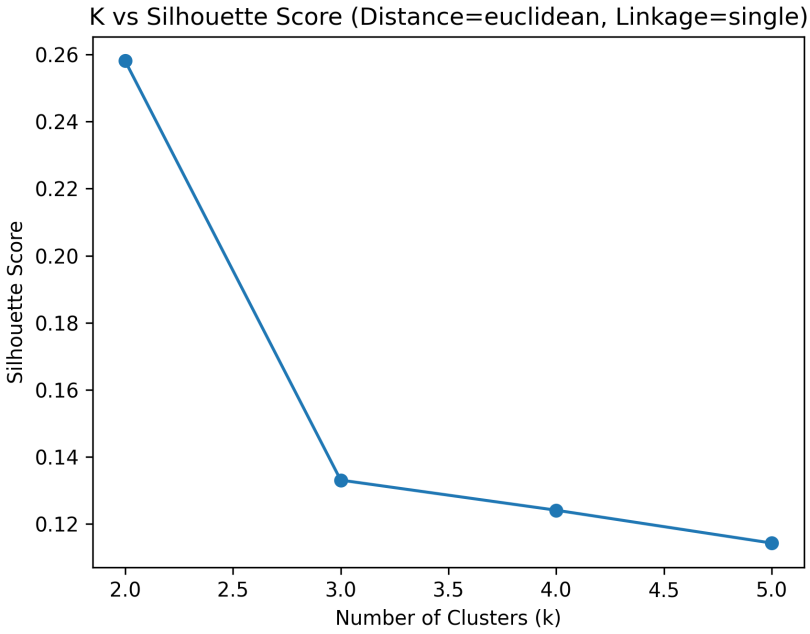

Figure 10: K vs Silhouette: Euclidean Distance (Single Linkage)

For this part, to implement both DBSCAN and Hierarchical Agglomerative Clustering methods, I used scikitlearn implementation. I have tried HAC with distance functions=["cosine", "euclidean"], linkage functions=["single", "complete"] and k values=[2,3,4,5] as it is stated in the assignment pdf. I have tried DBSCAN with hyperparameters,epsvalues=1 to 100, min sample values=[3, 5, 7], distance metrics=["euclidean", "cosine", "manhattan"].

For HAC, the silhouette scores decreased as K increased, showing that k=2 gave the most compact clustering. Since the dimensions were very large, to plot the dendrogram I used truncate mode="level" and p=3.

For DBSCAN, I have tried different hyperparameter configurations as I have stated and except euclidean(didnt include manhattan since it is same as cosine which has all -1 values), I didn't get positive values. For euclidean, the best 4 hyperparameters according to the highest silhouette score,
1)eps=18, min samples=3
2)eps=18, min samples=5
3)eps=17, min samples=3
4)eps=17, min samples=5

3.2 Part3 Dimensionality Reduction

For part3 dimensionality reduction., I used my own implementation of PCA where I implemented it on part2. For tSNE I used the scikitlearn implementation, and for UMAP I used the umap library.
I have tried 2 different settings. Firstly I have used the datasets directly with the dimensionality reduction methods. Then used HAC and DBSCAN with the best hyperparameter settings to provide labels, which I added their plots as well. As it can be seen from the plots, neither DBSCAN and HAC can separate this data perfectly when it is in these settings for dimensionality reduction.

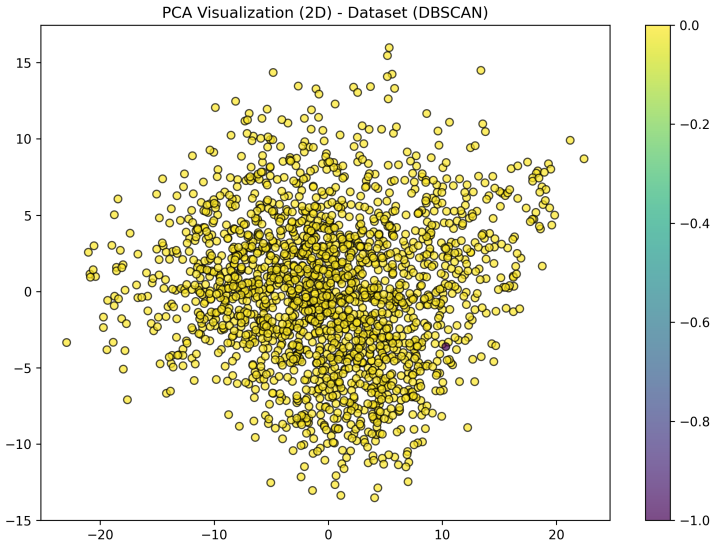

Figure 11: DBSCAN: PCA 2D Visualization

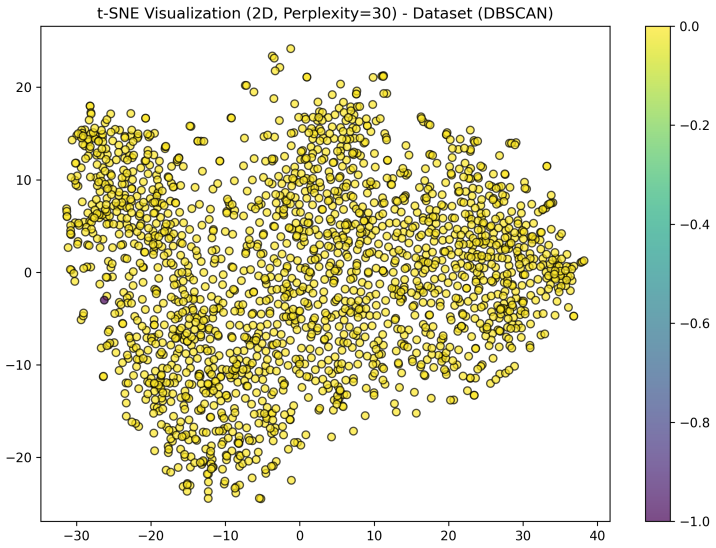

Figure 12: DBSCAN: t-SNE 2D Visualization (Perplexity = 30)

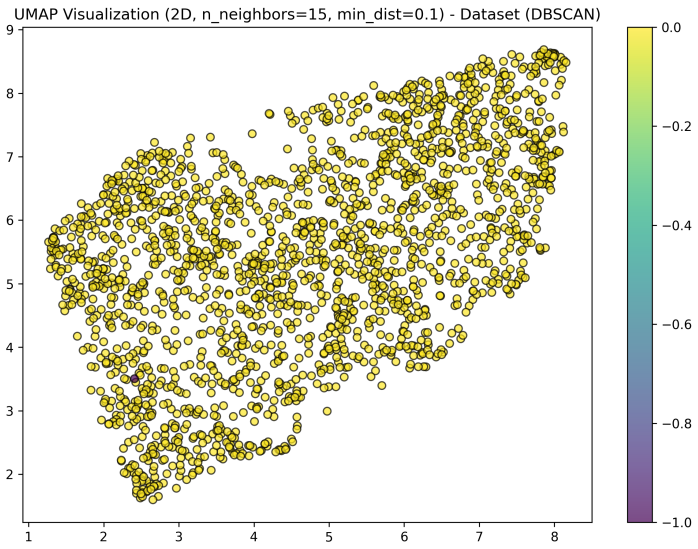

Figure 13: DBSCAN: UMAP 2D Visualization (Neighbors = 15, Min Distance = 0.1)

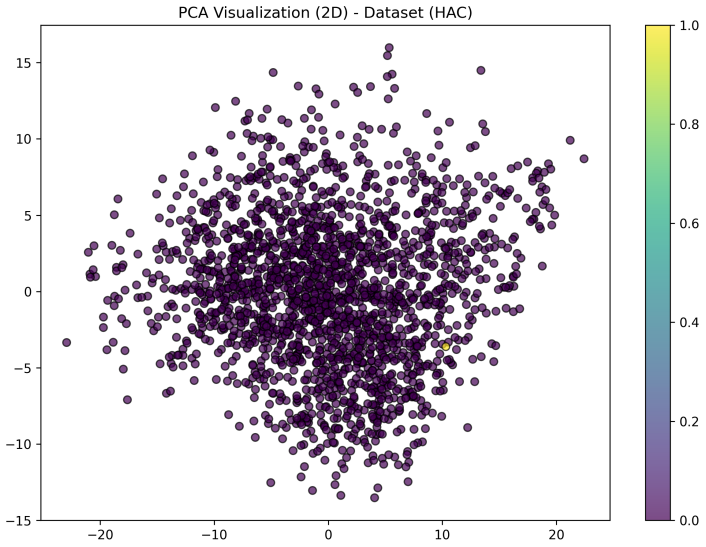

Figure 14: HAC: PCA 2D Visualization

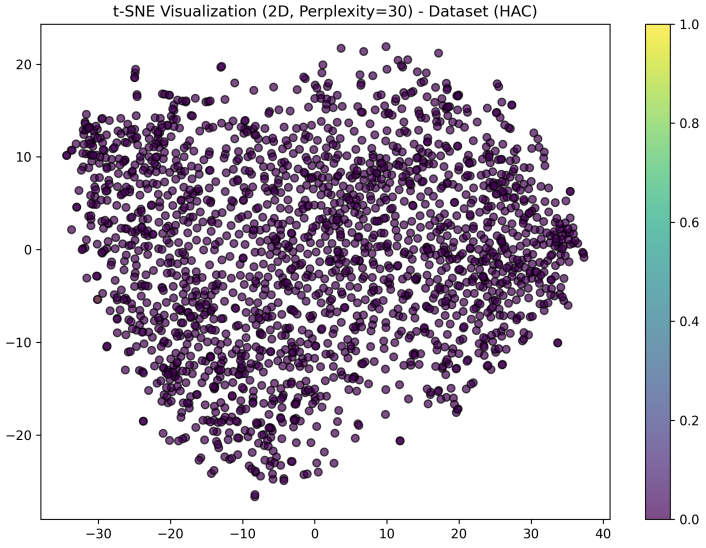

Figure 15: HAC: t-SNE 2D Visualization (Perplexity = 30)

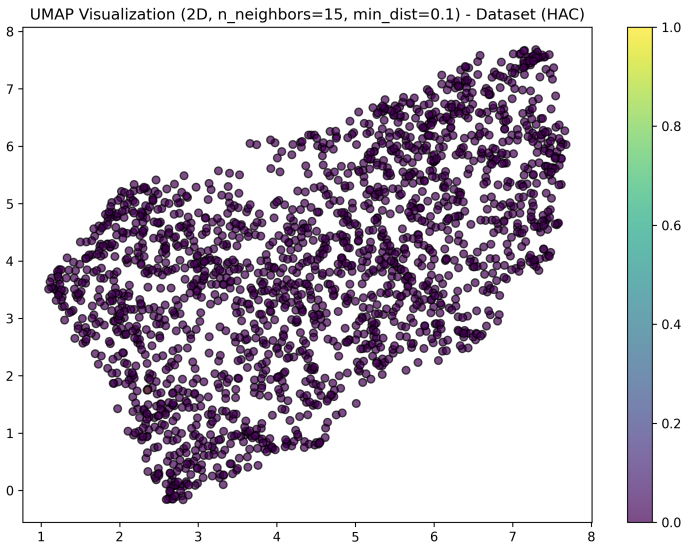

Figure 16: HAC: UMAP 2D Visualization (Neighbors = 15, Min Distance = 0.1)

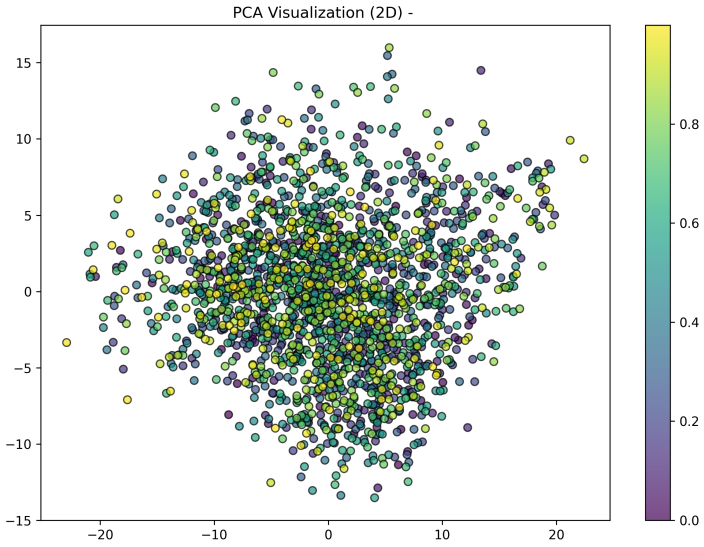

Figure 17: PCA 2D Visualization

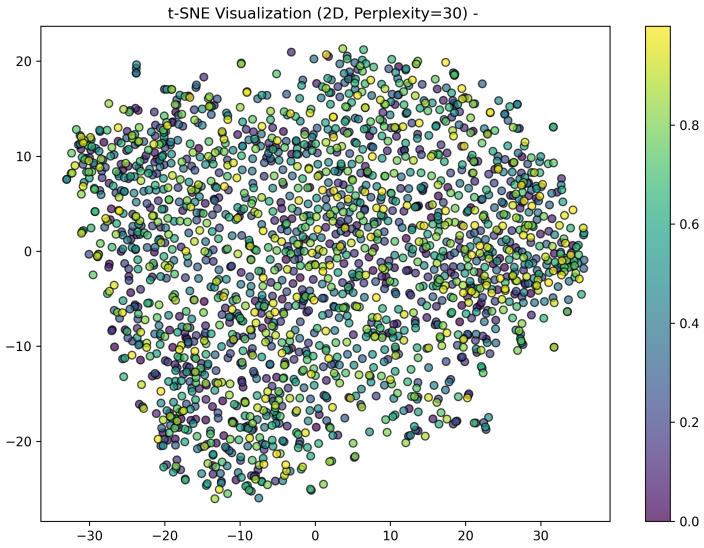

Figure 18: t-SNE 2D Visualization (Perplexity = 30,)

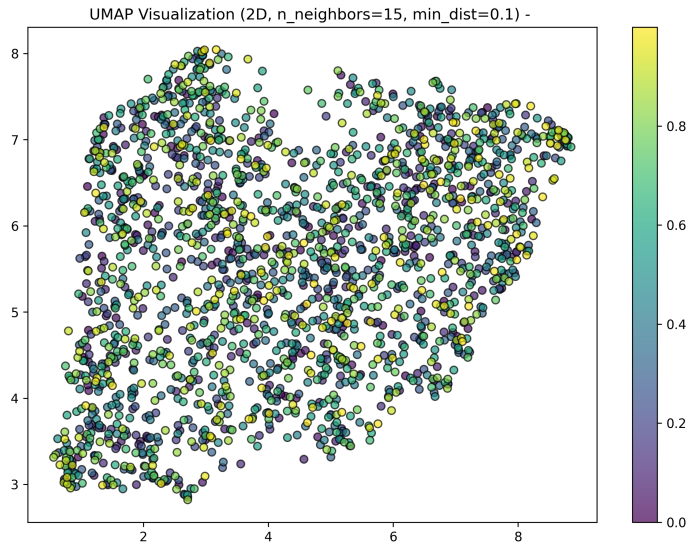

Figure 19: UMAP 2D Visualization (Neighbors = 15, Min Distance = 0.1,)

3.3 Runtime

For every iteration, the pairwise summation is calculated, so: HAC Time Complexity: $O(N^3 \cdot d)$

For the given example of " a dataset consisting of 1 million data points each of which has a dimension of 120000 (e.g., 200x200 RGB image)." It has a data size as $N = 10^6$, dimensionality $D = 120000$. For HAC, we know the time complexity, so if we calculate $O(10^118 \cdot 120000$ we get $O(1.2 \cdot 10^{23})$, which is very large. If we would use K means, from the time complexity formula $(O(I \cdot N \cdot K \cdot d))$, we need to decide K and iterations, lets assume they are 10 and 100 to for simplicity.. It gets $O(1.2 \cdot 10^{13}$. Which is a lot better compared to HAC.

In conclusion, K-means is significantly more scalable and efficient for large datasets, especially when dimensionality is high, so I would choose Kmeans for this given example.