CENG 499 HOMEWORK 2

İsmail Karabaş 2375186

December 7, 2023

Questions

1. **PART 1**
   I tested all the possible k values for each distance measure and below are the statistics of the outcome of each configuration.

   Considering the overall performance across iterations and the consistently high accuracy, the combination of Cosine Distance Measure and k value of 3 seems to be the most robust choice for this dataset. It provides a good balance between accuracy and generalization, as evidenced by the highest or competitive accuracy in most iterations.

```
                Testing KNN with DISTANCE MEASURE: COSINE
Iteration 1 , k value with max accuracy: 3, accuracy: 0.9533
Accuracies per k:
  k=1, average=0.9200, std_dev=0.0757, confidence_interval=(0.865861143138941, 0.9741388568610588)
  k=2, average=0.9200, std_dev=0.0757, confidence_interval=(0.865861143138941, 0.9741388568610588)
  k=3, average=0.9533, std_dev=0.0549, confidence_interval=(0.9140738116291093, 0.9925928550375575)
  k=4, average=0.9533, std_dev=0.0549, confidence_interval=(0.9140738116291093, 0.9925928550375575)
  k=5, average=0.9400, std_dev=0.0798, confidence_interval=(0.8829080414302213, 0.9970919585697788)
  k=7, average=0.9533, std_dev=0.0549, confidence_interval=(0.9140738116291093, 0.9925928550375575)

Iteration 2 , k value with max accuracy: 3, accuracy: 0.9533
Accuracies per k:
  k=1, average=0.9133, std_dev=0.0773, confidence_interval=(0.8580400000000001, 0.9686266666666667)
  k=2, average=0.9133, std_dev=0.0773, confidence_interval=(0.8580400000000001, 0.9686266666666667)
  k=3, average=0.9533, std_dev=0.0549, confidence_interval=(0.9140738116291093, 0.9925928550375575)
  k=4, average=0.9467, std_dev=0.0613, confidence_interval=(0.9028452026209377, 0.9904881307123956)
  k=5, average=0.9400, std_dev=0.0734, confidence_interval=(0.8875200592818764, 0.9924799407181237)
  k=7, average=0.9333, std_dev=0.0703, confidence_interval=(0.8830666666666667, 0.9836)

Iteration 3 , k value with max accuracy: 3, accuracy: 0.9467
Accuracies per k:
  k=1, average=0.9133, std_dev=0.0450, confidence_interval=(0.8811469621665043, 0.9455197045001625)
  k=2, average=0.9133, std_dev=0.0450, confidence_interval=(0.8811469621665043, 0.9455197045001625)
  k=3, average=0.9467, std_dev=0.0422, confidence_interval=(0.9165066666666667, 0.9768266666666666)
  k=4, average=0.9333, std_dev=0.0314, confidence_interval=(0.9108533965995355, 0.9558132700671312)
  k=5, average=0.9400, std_dev=0.0378, confidence_interval=(0.9129305715694707, 0.9670694284305295)
  k=7, average=0.9467, std_dev=0.0422, confidence_interval=(0.9165066666666667, 0.9768266666666666)

Iteration 4 , k value with max accuracy: 3, accuracy: 0.9600
Accuracies per k:
  k=1, average=0.9200, std_dev=0.0526, confidence_interval=(0.8823838710716329, 0.9576161289283674)
  k=2, average=0.9200, std_dev=0.0526, confidence_interval=(0.8823838710716329, 0.9576161289283674)
```

```
  k=3, average=0.9600, std_dev=0.0344, confidence_interval=(0.9353744631192198, 0.9846255368807801)
  k=4, average=0.9467, std_dev=0.0422, confidence_interval=(0.9165066666666667, 0.9768266666666666)
  k=5, average=0.9333, std_dev=0.0314, confidence_interval=(0.9108533965995355, 0.9558132700671312)
  k=7, average=0.9400, std_dev=0.0378, confidence_interval=(0.9129305715694707, 0.9670694284305295)

Iteration 5 , k value with max accuracy: 3, accuracy: 0.9533
Accuracies per k:
  k=1, average=0.9133, std_dev=0.1045, confidence_interval=(0.8386065610968123, 0.9880601055698544)
  k=2, average=0.9133, std_dev=0.1045, confidence_interval=(0.8386065610968123, 0.9880601055698544)
  k=3, average=0.9533, std_dev=0.0834, confidence_interval=(0.8936449737758227, 1.013021692890844)
  k=4, average=0.9533, std_dev=0.0706, confidence_interval=(0.902815958544499, 1.0038507081221677)
  k=5, average=0.9333, std_dev=0.0943, confidence_interval=(0.8658935231319397, 1.000773143534727)
  k=7, average=0.9467, std_dev=0.0689, confidence_interval=(0.8974155929051062, 0.9959177404282271)

########################################################################################################################
                    Testing KNN with DISTANCE MEASURE: MINKOWSKI
Iteration 1 , k value with max accuracy: 1, accuracy: 0.9400
Accuracies per k:
  k=1, average=0.9400, std_dev=0.0584, confidence_interval=(0.8982453707157319, 0.9817546292842683)
  k=2, average=0.9400, std_dev=0.0584, confidence_interval=(0.8982453707157319, 0.9817546292842683)
  k=3, average=0.9333, std_dev=0.0831, confidence_interval=(0.8738570112472386, 0.9928096554194281)
  k=4, average=0.9400, std_dev=0.0584, confidence_interval=(0.8982453707157319, 0.9817546292842683)
  k=5, average=0.9400, std_dev=0.0734, confidence_interval=(0.8875200592818764, 0.9924799407181237)
  k=7, average=0.9400, std_dev=0.0663, confidence_interval=(0.8925785215095289, 0.9874214784904712)

Iteration 2 , k value with max accuracy: 7, accuracy: 0.9600
Accuracies per k:
  k=1, average=0.9333, std_dev=0.0544, confidence_interval=(0.8943969407594614, 0.9722697259072053)
  k=2, average=0.9333, std_dev=0.0544, confidence_interval=(0.8943969407594614, 0.9722697259072053)
  k=3, average=0.9533, std_dev=0.0450, confidence_interval=(0.9211469621665043, 0.9855197045001625)
  k=4, average=0.9467, std_dev=0.0526, confidence_interval=(0.9090505377382994, 0.984282795595034)
  k=5, average=0.9467, std_dev=0.0613, confidence_interval=(0.9028452026209377, 0.9904881307123956)
  k=7, average=0.9600, std_dev=0.0644, confidence_interval=(0.9139298390133772, 1.0060701609866227)

Iteration 3 , k value with max accuracy: 5, accuracy: 0.9533
Accuracies per k:
  k=1, average=0.9400, std_dev=0.0492, confidence_interval=(0.9048133333333334, 0.9751866666666668)
  k=2, average=0.9400, std_dev=0.0492, confidence_interval=(0.9048133333333334, 0.9751866666666668)
  k=3, average=0.9467, std_dev=0.0422, confidence_interval=(0.9165066666666667, 0.9768266666666666)
  k=4, average=0.9400, std_dev=0.0378, confidence_interval=(0.9129305715694707, 0.9670694284305295)
  k=5, average=0.9533, std_dev=0.0450, confidence_interval=(0.9211469621665043, 0.9855197045001625)
  k=7, average=0.9333, std_dev=0.0544, confidence_interval=(0.8943969407594614, 0.9722697259072053)

Iteration 4 , k value with max accuracy: 7, accuracy: 0.9533
Accuracies per k:
  k=1, average=0.9267, std_dev=0.0734, confidence_interval=(0.8741867259485431, 0.9791466073847904)
  k=2, average=0.9267, std_dev=0.0734, confidence_interval=(0.8741867259485431, 0.9791466073847904)
  k=3, average=0.9400, std_dev=0.0584, confidence_interval=(0.8982453707157319, 0.9817546292842683)
  k=4, average=0.9333, std_dev=0.0770, confidence_interval=(0.8782689588854806, 0.9883977077811861)
  k=5, average=0.9467, std_dev=0.0689, confidence_interval=(0.8974155929051062, 0.9959177404282271)
  k=7, average=0.9533, std_dev=0.0450, confidence_interval=(0.9211469621665043, 0.9855197045001625)

Iteration 5 , k value with max accuracy: 3, accuracy: 0.9467
Accuracies per k:
```

```
  k=1, average=0.9267, std_dev=0.0734, confidence_interval=(0.8741867259485431, 0.9791466073847904)
  k=2, average=0.9267, std_dev=0.0734, confidence_interval=(0.8741867259485431, 0.9791466073847904)
  k=3, average=0.9467, std_dev=0.0613, confidence_interval=(0.9028452026209377, 0.9904881307123956)
  k=4, average=0.9333, std_dev=0.0629, confidence_interval=(0.8883734598657376, 0.9782932068009291)
  k=5, average=0.9400, std_dev=0.0492, confidence_interval=(0.9048133333333334, 0.9751866666666668)
  k=7, average=0.9400, std_dev=0.0584, confidence_interval=(0.8982453707157319, 0.9817546292842683)

##########################################################################################################################
                    Testing KNN with DISTANCE MEASURE:MAHALANOBIS
Iteration 1 , k value with max accuracy: 3, accuracy: 0.9133
Accuracies per k:
  k=1, average=0.8733, std_dev=0.0663, confidence_interval=(0.8259118548428622, 0.9207548118238046)
  k=2, average=0.8733, std_dev=0.0663, confidence_interval=(0.8259118548428622, 0.9207548118238046)
  k=3, average=0.9133, std_dev=0.0632, confidence_interval=(0.8680933333333335, 0.9585733333333334)
  k=4, average=0.9133, std_dev=0.0450, confidence_interval=(0.8811469621665043, 0.9455197045001625)
  k=5, average=0.9000, std_dev=0.0567, confidence_interval=(0.8594737177185527, 0.9405262822814475)
  k=7, average=0.8733, std_dev=0.0663, confidence_interval=(0.8259118548428622, 0.9207548118238046)

Iteration 2 , k value with max accuracy: 5, accuracy: 0.9200
Accuracies per k:
  k=1, average=0.8867, std_dev=0.0892, confidence_interval=(0.8228854168966738, 0.9504479164366596)
  k=2, average=0.8867, std_dev=0.0892, confidence_interval=(0.8228854168966738, 0.9504479164366596)
  k=3, average=0.9067, std_dev=0.0717, confidence_interval=(0.8554045238233472, 0.957928809509986)
  k=4, average=0.9133, std_dev=0.0834, confidence_interval=(0.8536449737758228, 0.9730216928908441)
  k=5, average=0.9200, std_dev=0.0526, confidence_interval=(0.8823838710716329, 0.9576161289283674)
  k=7, average=0.8800, std_dev=0.0878, confidence_interval=(0.8172169534561016, 0.9427830465438987)

Iteration 3 , k value with max accuracy: 3, accuracy: 0.8933
Accuracies per k:
  k=1, average=0.8800, std_dev=0.0422, confidence_interval=(0.8498400000000002, 0.9101600000000001)
  k=2, average=0.8800, std_dev=0.0422, confidence_interval=(0.8498400000000002, 0.9101600000000001)
  k=3, average=0.8933, std_dev=0.0783, confidence_interval=(0.8373587422723489, 0.9493079243943178)
  k=4, average=0.8933, std_dev=0.0466, confidence_interval=(0.8599901987742937, 0.9266764678923729)
  k=5, average=0.8867, std_dev=0.0706, confidence_interval=(0.8361492918778324, 0.937184041455501)
  k=7, average=0.8867, std_dev=0.0773, confidence_interval=(0.8313733333333334, 0.94196)

Iteration 4 , k value with max accuracy: 5, accuracy: 0.9267
Accuracies per k:
  k=1, average=0.8867, std_dev=0.0996, confidence_interval=(0.8154013670227773, 0.9579319663105561)
  k=2, average=0.8867, std_dev=0.0996, confidence_interval=(0.8154013670227773, 0.9579319663105561)
  k=3, average=0.9067, std_dev=0.0783, confidence_interval=(0.8506920756056822, 0.9626412577276511)
  k=4, average=0.9067, std_dev=0.0953, confidence_interval=(0.8384816425696473, 0.9748516907636859)
  k=5, average=0.9267, std_dev=0.0798, confidence_interval=(0.869574708096888, 0.9837586252364455)
  k=7, average=0.8933, std_dev=0.0843, confidence_interval=(0.8330133333333333, 0.9536533333333334)

Iteration 5 , k value with max accuracy: 4, accuracy: 0.9067
Accuracies per k:
  k=1, average=0.8733, std_dev=0.1063, confidence_interval=(0.7972660636892144, 0.9494006029774524)
  k=2, average=0.8733, std_dev=0.1063, confidence_interval=(0.7972660636892144, 0.9494006029774524)
  k=3, average=0.9000, std_dev=0.0786, confidence_interval=(0.8438001581655054, 0.9561998418344947)
  k=4, average=0.9067, std_dev=0.0953, confidence_interval=(0.8384816425696473, 0.9748516907636859)
  k=5, average=0.8867, std_dev=0.1091, confidence_interval=(0.8086318151934664, 0.964701518139867)
  k=7, average=0.8867, std_dev=0.0892, confidence_interval=(0.8228854168966738, 0.9504479164366596)
```

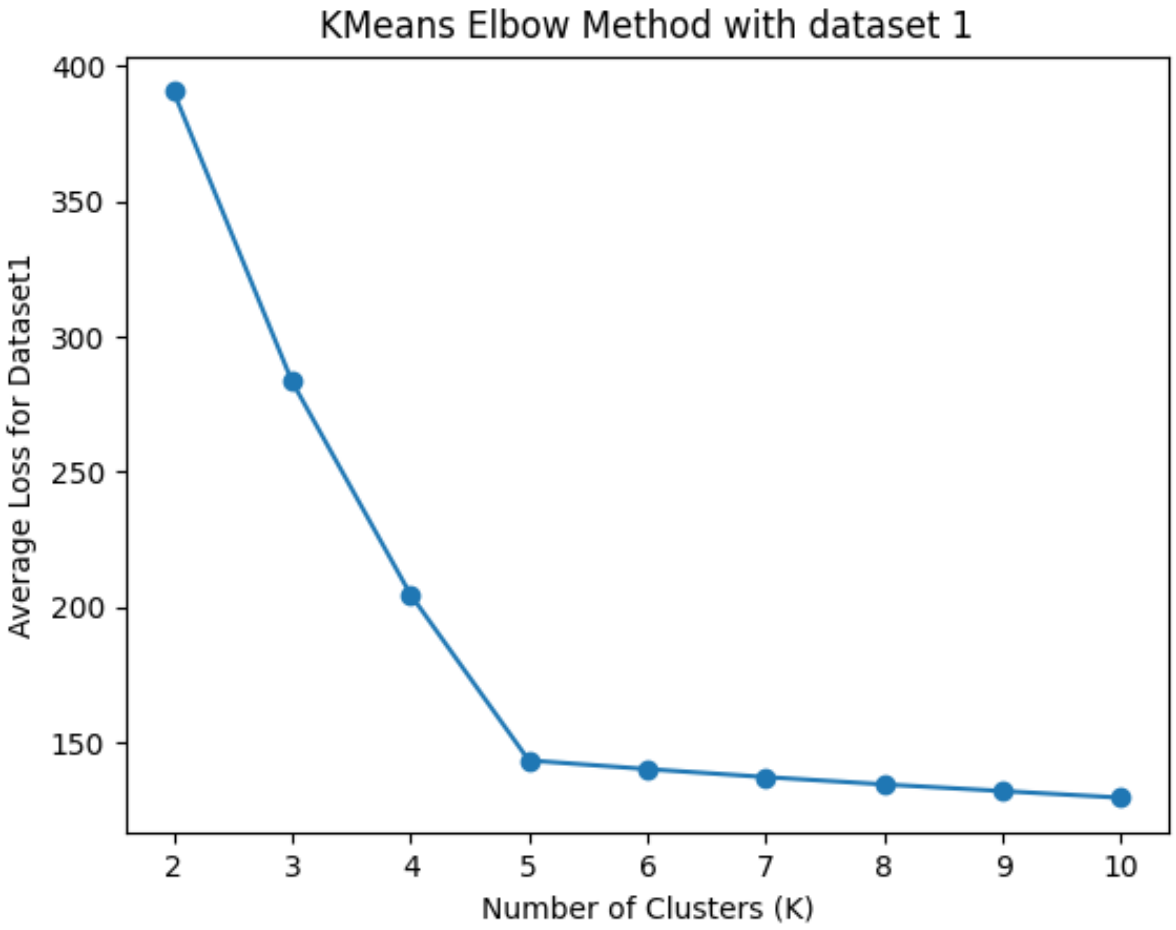
2. **PART 2**

(a) **Kmeans**

Elbow Method:

The elbow method looks for the "elbow" or bend in the plot of the metric (such as loss or inertia) against the number of clusters. This bend represents the point where the marginal gain in clustering quality decreases significantly, indicating that adding more clusters doesn't contribute much to improving the model's fit to the data.

The idea is to find the point where the increase in the number of clusters starts providing diminishing returns, forming an elbow shape in the plot. This optimal number of clusters is a balance between fitting the data well and avoiding overfitting, helping to identify a suitable level of granularity in the clustering solution.

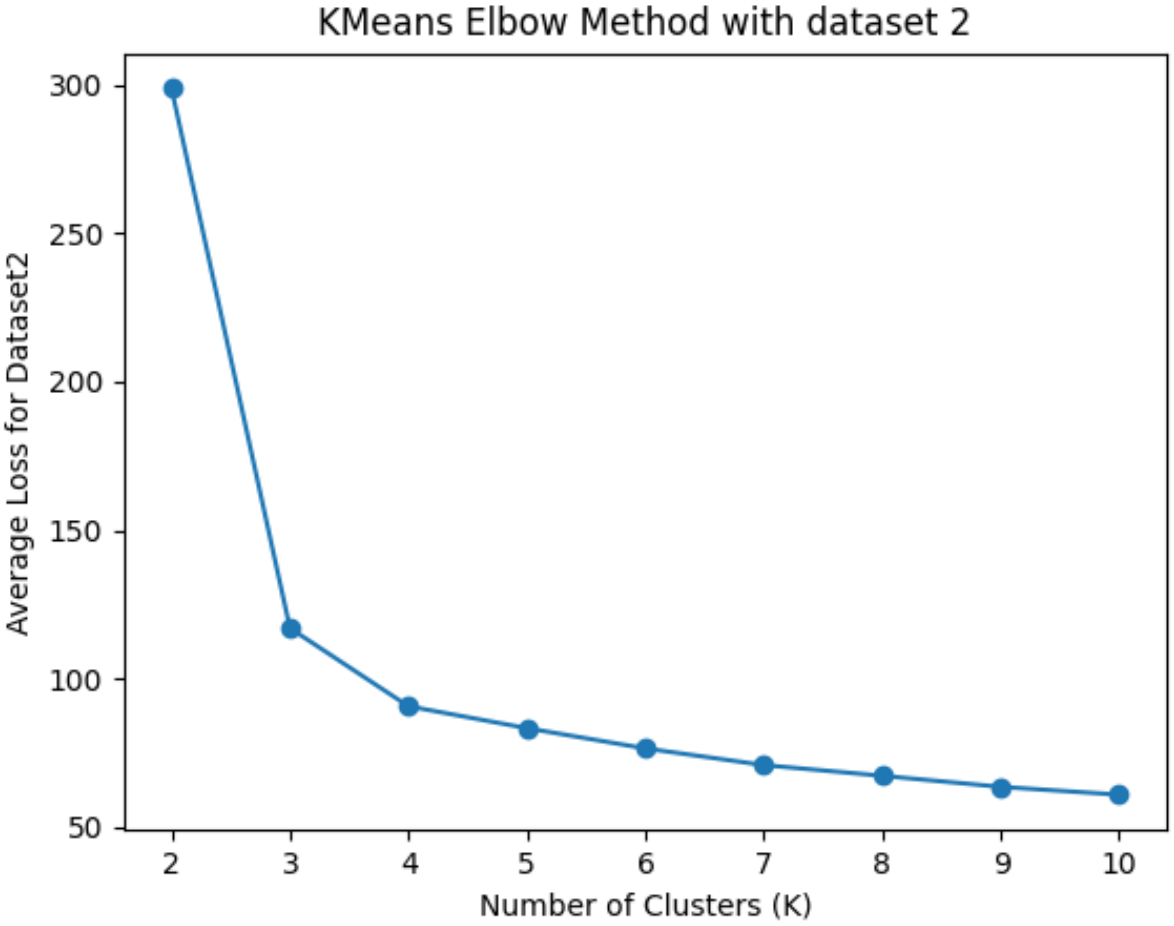


In this dataset the best number of cluster is 5, using the elbow method we can see that after **5**, increasing the number of clusters does not give a huge gain. And obviously increasing number of k will result in a decrease in the loss.

Confidence Intervals for Accuracy Values:

```
Dataset1, k = 2, average loss: 390.4839, for 95% confidence interval: ±0.6839
```

```
Dataset1, k = 3, average loss: 283.4793, for 95% confidence interval: ±7.2346
Dataset1, k = 4, average loss: 204.8770, for 95% confidence interval: ±0.0000
Dataset1, k = 5, average loss: 143.5868, for 95% confidence interval: ±0.0000
Dataset1, k = 6, average loss: 140.3836, for 95% confidence interval: ±0.0500
Dataset1, k = 7, average loss: 137.4462, for 95% confidence interval: ±0.0686
Dataset1, k = 8, average loss: 134.7478, for 95% confidence interval: ±0.1782
Dataset1, k = 9, average loss: 132.2826, for 95% confidence interval: ±0.2327
Dataset1, k = 10, average loss: 129.8314, for 95% confidence interval: ±0.2585
```



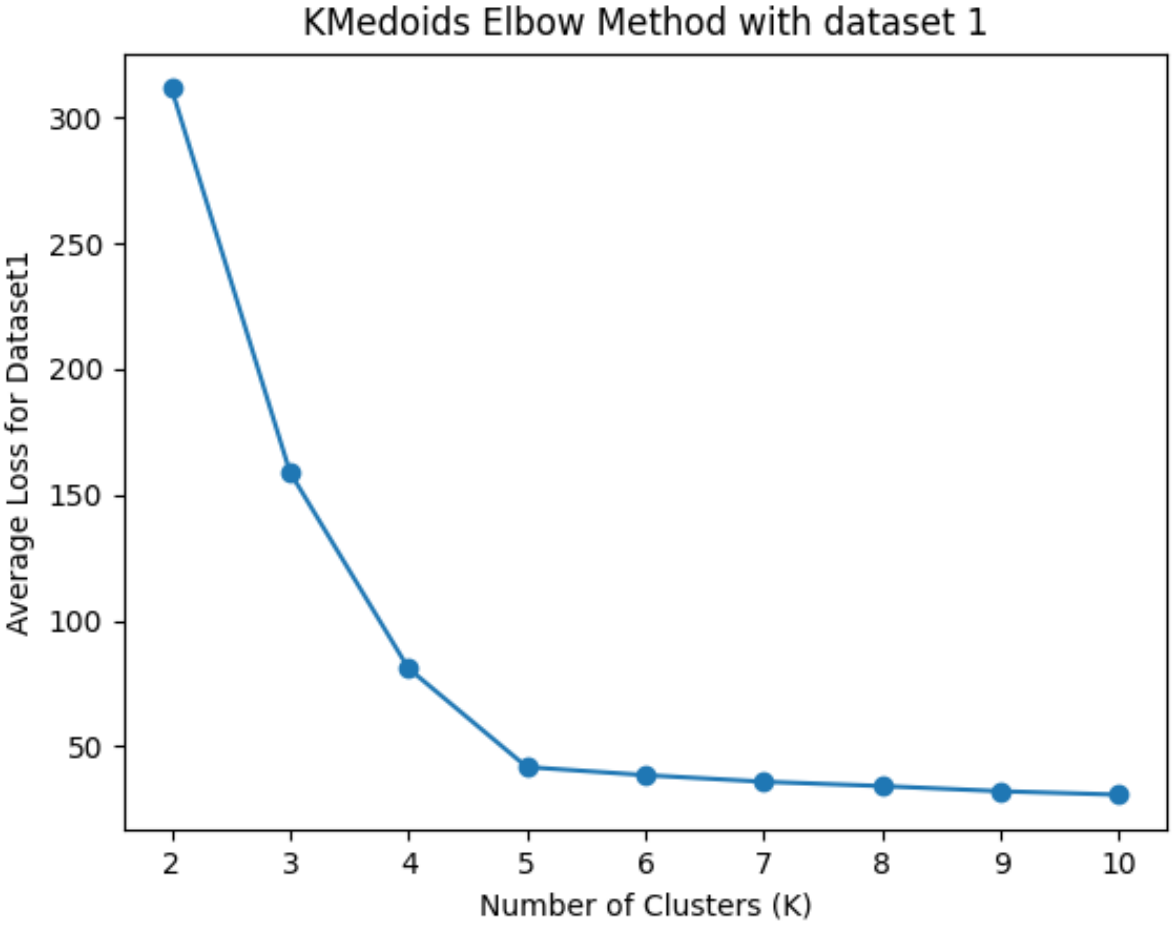
This data set is a little bit tricky. In lessons, the professor said that the elbow point will be clear to point. However in this graph we see two elbow points one at the k=3 and one at k = 4. When we look at the decrease in loss values, we can see that the decrease at point 4 is not as much as the previous points but still it is 4 times bigger than the following k values. Therefore the best K value is **4**.

Confidence Intervals for Accuracy Values:

```
Dataset2, k = 2, average loss: 298.7745, for 95% confidence interval: ±0.0000
Dataset2, k = 3, average loss: 117.0799, for 95% confidence interval: ±0.0000
Dataset2, k = 4, average loss: 90.7792, for 95% confidence interval: ±0.0019
Dataset2, k = 5, average loss: 83.3987, for 95% confidence interval: ±0.0983
Dataset2, k = 6, average loss: 76.5814, for 95% confidence interval: ±0.1462
Dataset2, k = 7, average loss: 70.9696, for 95% confidence interval: ±0.7354
```

```
Dataset2, k = 8, average loss: 67.3678, for 95% confidence interval: ±0.7623
Dataset2, k = 9, average loss: 63.6430, for 95% confidence interval: ±0.5342
Dataset2, k = 10, average loss: 61.0045, for 95% confidence interval: ±0.3885
```

(b) **Kmedoids**

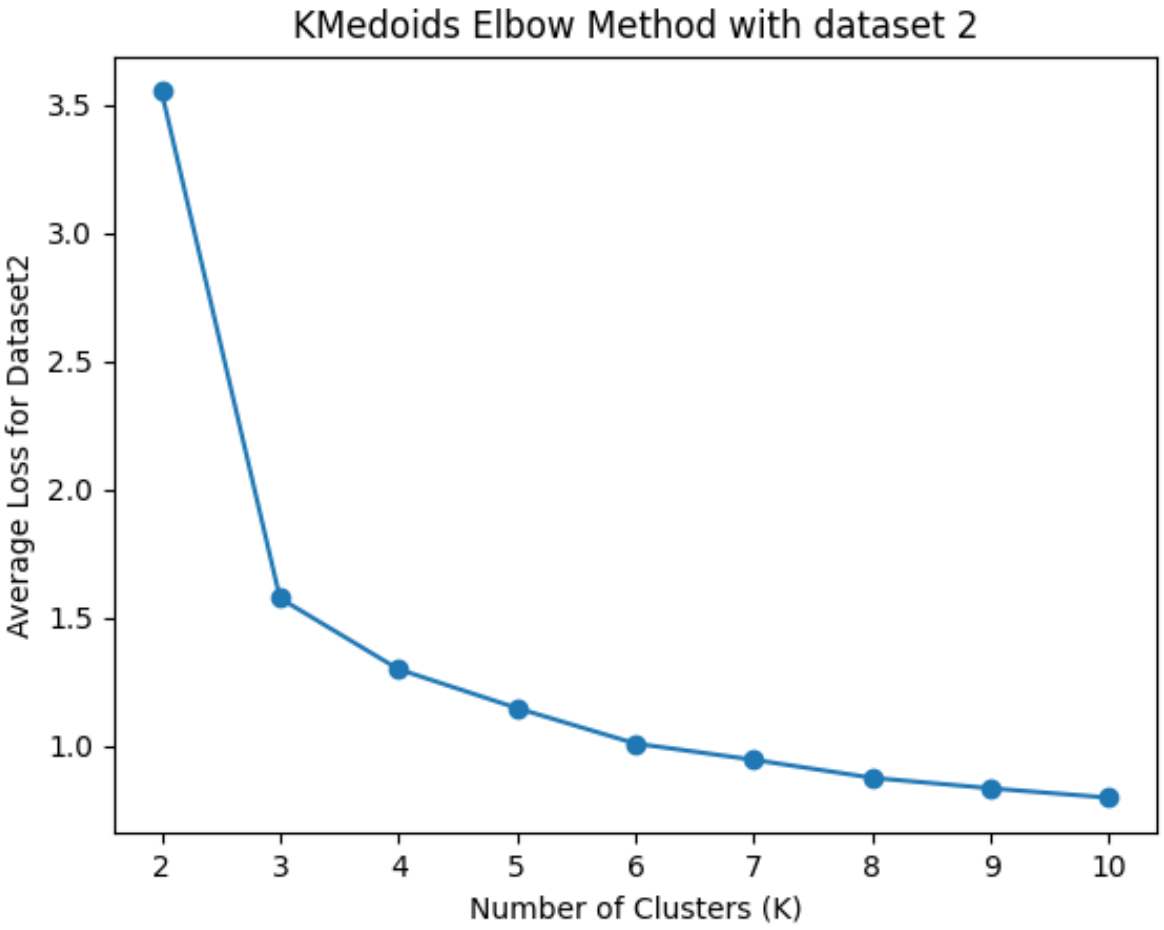


Similar to the Kmeans, using Kmedoids for this dataset the best number of cluster is 5, using the elbow method we can see that after **5**, increasing the number of clusters does not give a huge gain. And obviously increasing number of k will result in a decrease in the loss.

Confidence Intervals for Accuracy Values:

```
Dataset1, k = 2, average loss: 311.6445, for 95% confidence interval: ±0.0000
Dataset1, k = 3, average loss: 159.0805, for 95% confidence interval: ±0.0000
Dataset1, k = 4, average loss: 81.0789, for 95% confidence interval: ±0.0000
Dataset1, k = 5, average loss: 41.8291, for 95% confidence interval: ±0.0006
Dataset1, k = 6, average loss: 38.6117, for 95% confidence interval: ±0.1185
Dataset1, k = 7, average loss: 35.9978, for 95% confidence interval: ±0.2147
Dataset1, k = 8, average loss: 34.2732, for 95% confidence interval: ±0.4842
```

```
Dataset1, k = 9, average loss: 32.2091, for 95% confidence interval: ±0.5827
Dataset1, k = 10, average loss: 30.8355, for 95% confidence interval: ±0.5948
```



Similar to However in this graph we see two elbow points one at the k=3 and one at k = 4. When we look at the decrease in loss values, However here there is not much decrease at k = 4 . So the best number of clusters is **3**

Confidence Intervals for Accuracy Values:

```
Dataset2, k = 2, average loss: 3.5535, for 95% confidence interval: ±0.0000
Dataset2, k = 3, average loss: 1.5771, for 95% confidence interval: ±0.0000
Dataset2, k = 4, average loss: 1.3002, for 95% confidence interval: ±0.0185
Dataset2, k = 5, average loss: 1.1486, for 95% confidence interval: ±0.0221
Dataset2, k = 6, average loss: 1.0103, for 95% confidence interval: ±0.0144
Dataset2, k = 7, average loss: 0.9468, for 95% confidence interval: ±0.0131
Dataset2, k = 8, average loss: 0.8767, for 95% confidence interval: ±0.0162
Dataset2, k = 9, average loss: 0.8357, for 95% confidence interval: ±0.0072
Dataset2, k = 10, average loss: 0.7991, for 95% confidence interval: ±0.0143
```

Run time analysis

Kmeans:
Consider a scenario with k cluster points. In each iteration, the task involves computing the distance from each data point to all clusters, selecting the minimum distance for each point. This operation amounts to n calculations, with each computation's cost scaling with both the dimensionality (d) and the number of clusters (k). Consequently, the computational effort for this step is proportional to n * k * d.

Furthermore, at the conclusion of each iteration, the algorithm computes the average of clusters, which is proportional to N * d, considering N data points. When factoring in the number of iterations (I), the overall time complexity becomes I * (N * k * d + N * d).

which is $\boxed{\text{O(I * (N * k * d + N * d))}}$ assuming the number N is the largest number. If the other parameters are small , we could say it is O(N) but, even for medium sized parameter the coefficient of N will be around 100 and it is safe to use the cumbersome one.

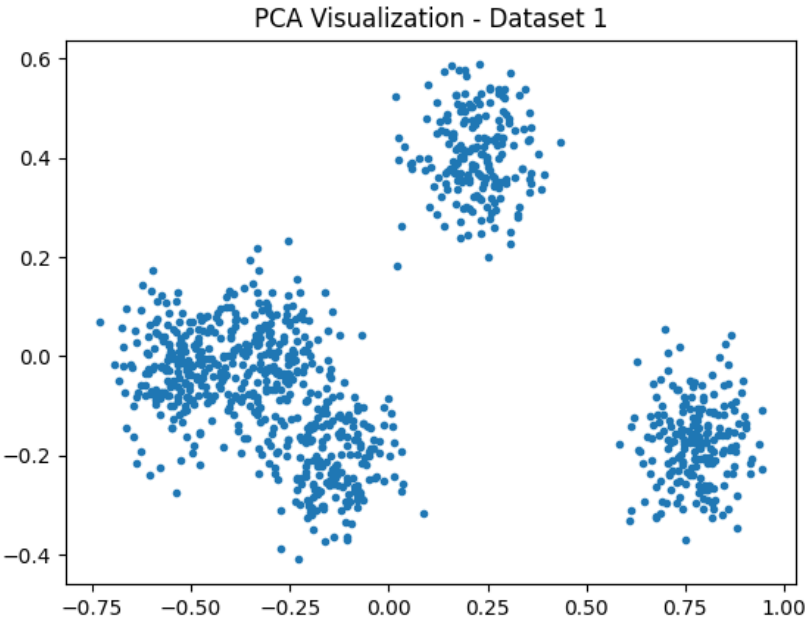
Kmedoids:
Initializing medoids: This step has a time complexity of O(K), where K is the number of clusters.

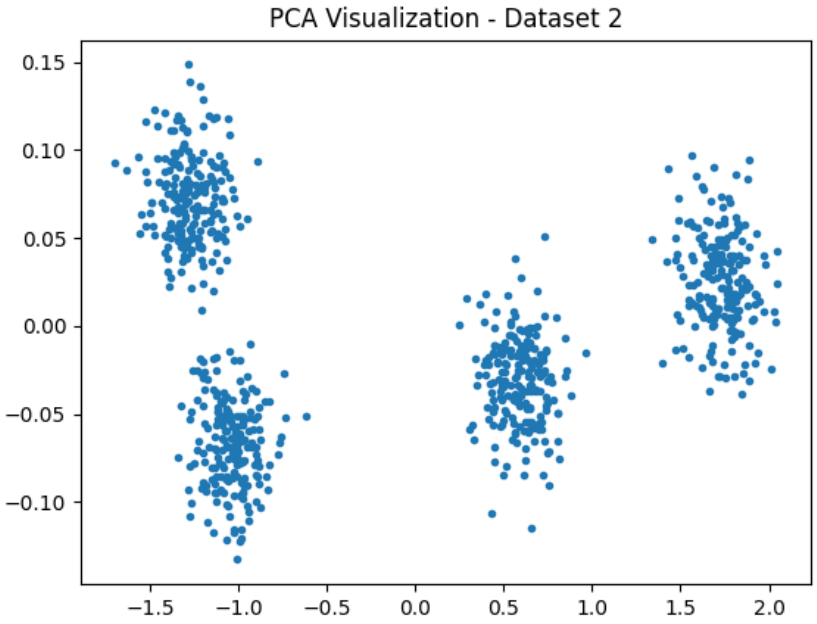
Assigning each point to the nearest cluster: This involves iterating through each data point (N) and each cluster (K) to find the minimum distance. Thus, the complexity for this step is O(N * K).

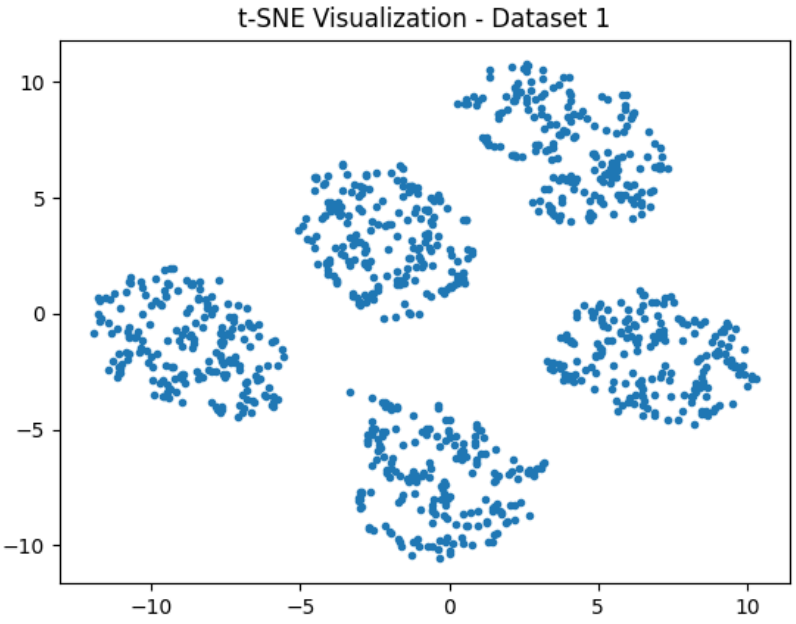
Updating medoids: For each cluster, it calculates the sum of distances in a rough calculation (N/t * N/t) between the points within the cluster, t could be any number. The overall complexity for this step is $O(K * N^2)$.

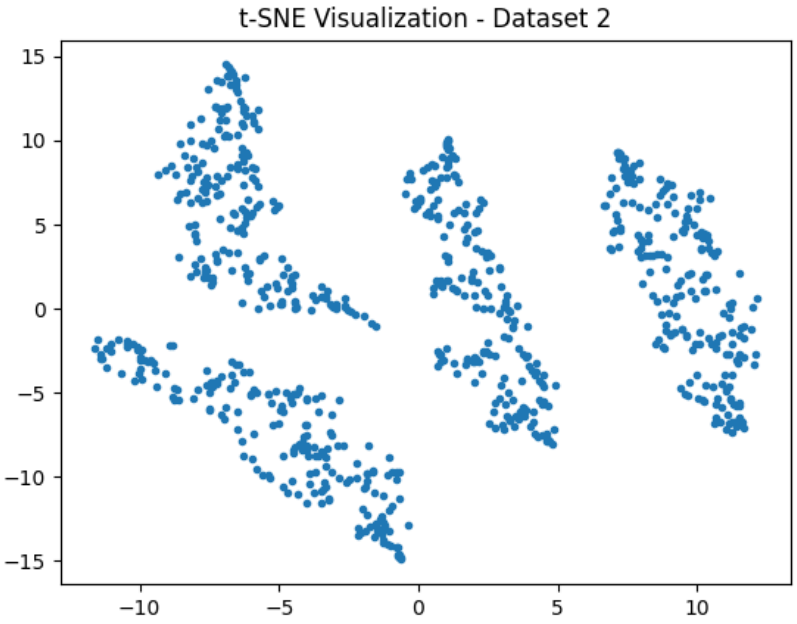
Hence, the runtime complexity is quadratic, $\boxed{O(I * K * N^2)}$, which highlights a dependency on the number of clusters (K), the square of number of data points (N), and the number of iterations (I).

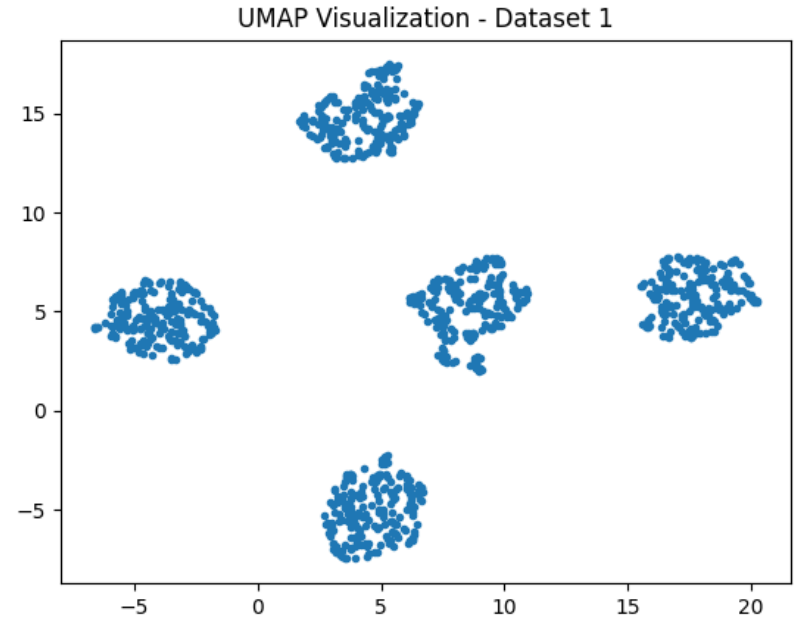
(c) **Dimensionality Reductions**

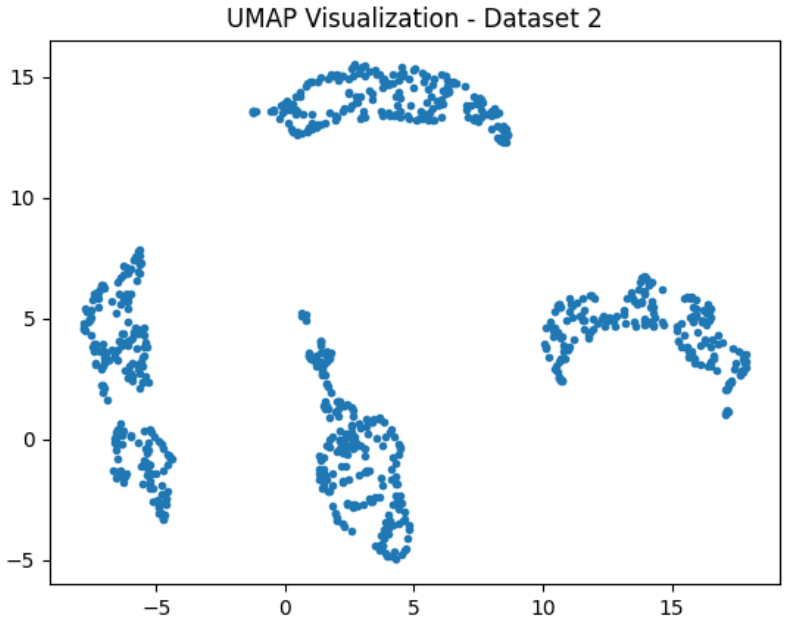



PCA Visualization - Dataset 2
0.15
0.10
0.05
0.00
−0.05
−0.10
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


t-SNE Visualization - Dataset 1
10
5
0
−5
−10
−10
−5
0
5
10


t-SNE Visualization - Dataset 2
15
10
5
0
−5
−10
−15
−10
−5
0
5
10


UMAP Visualization - Dataset 1
15
10
5
0
−5
−5
0
5
10
15
20

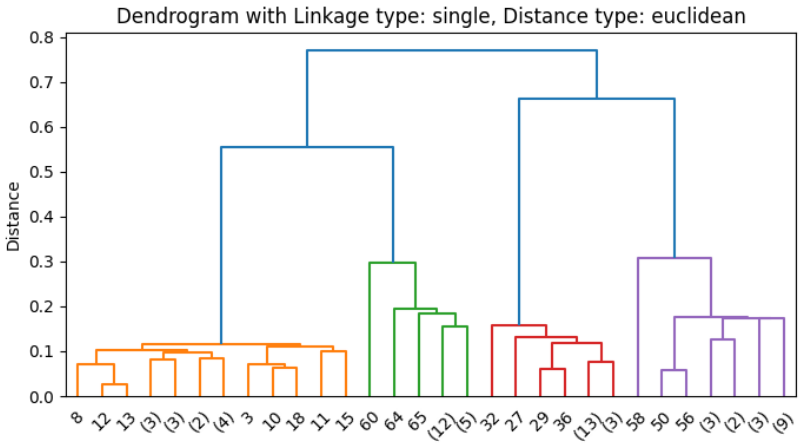


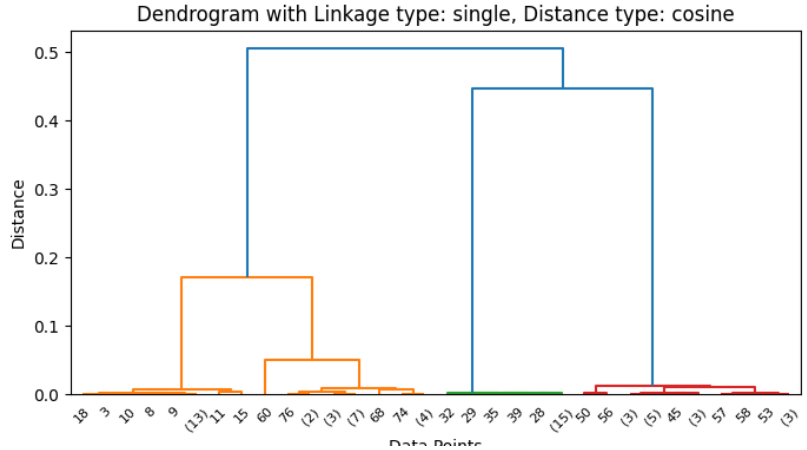
I used these three dimension reduction techniques, In my opinion UMAP and t-SNE are better than PAC. As you can see PCA can't correctly divide the dataset with 5 clusters even in the wrong graph we can see there is 5 clusters.

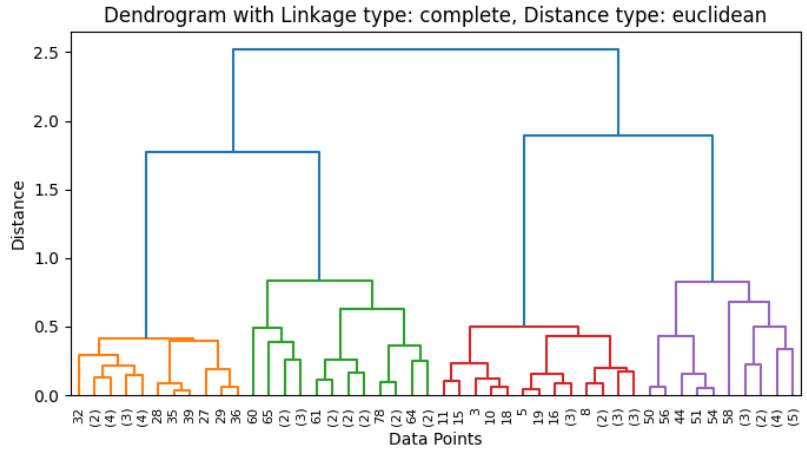
Overall the findings of dimension reduction methods and clustering algorithms are same. The only exception is according to elbow method in the kmedoid for dataset 2 we found the best cluster number as 3. I think this mistake is due to location of cluster. on the visualizations two clusters are very close to each other this could be the cause of the and relatively small change while passing from k=3 to k = 4.

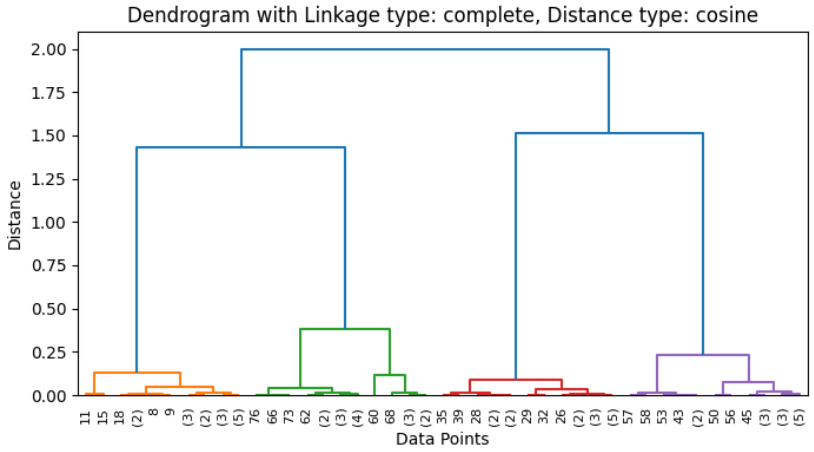
3. **PART 3**

(a) **Dendograms**

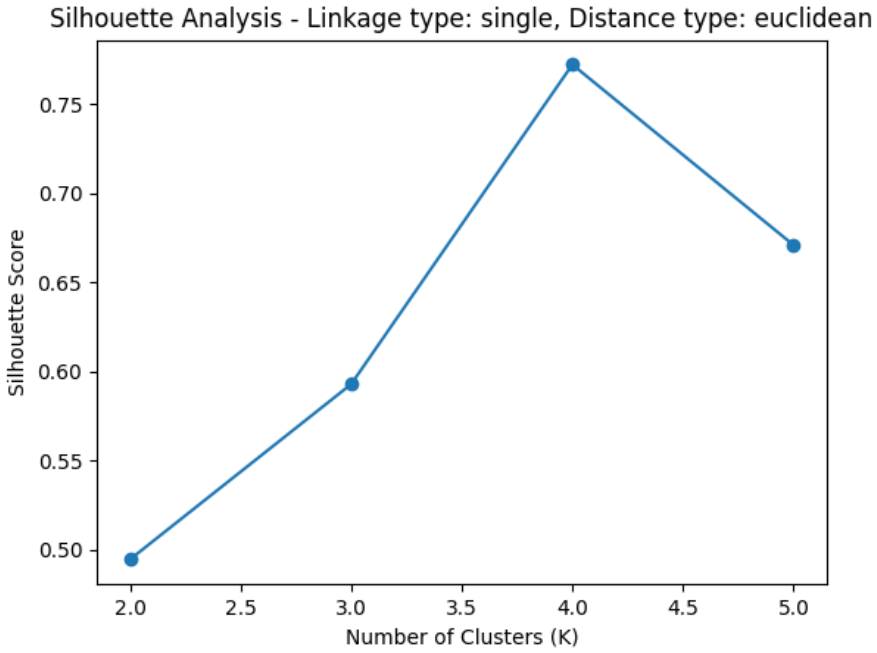


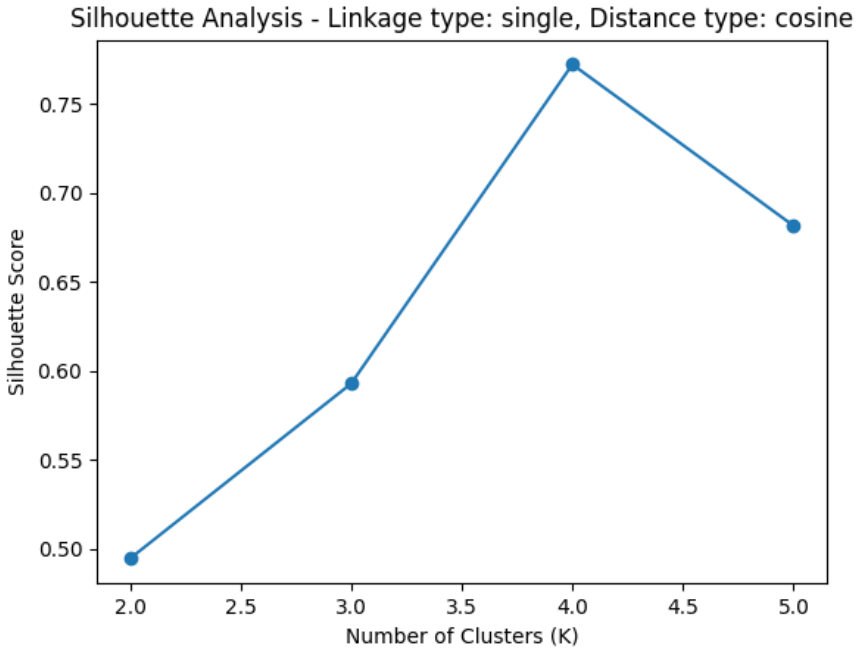


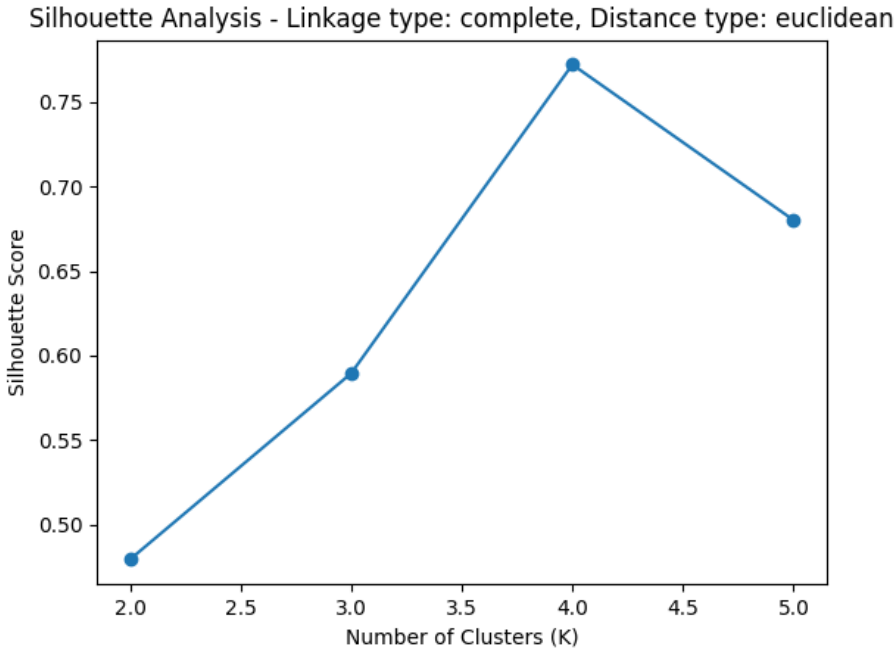


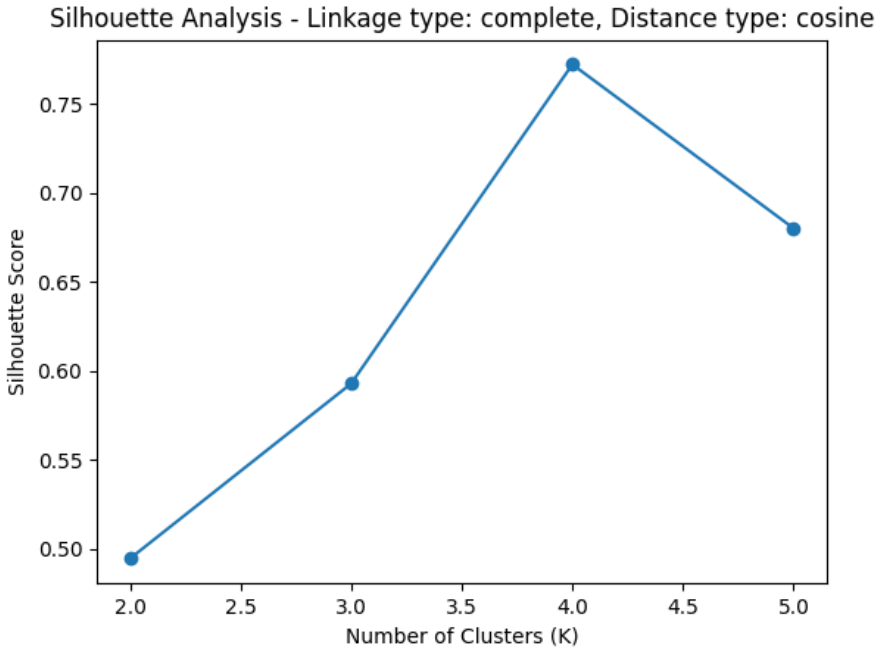


(b) **Silhouette Value Plots**


Silhouette Analysis - Linkage type: single, Distance type: euclidean
Silhouette Score
0.75
0.70
0.65
0.60
0.55
0.50
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Number of Clusters (K)


Silhouette Analysis - Linkage type: single, Distance type: cosine
Silhouette Score
0.75
0.70
0.65
0.60
0.55
0.50
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Number of Clusters (K)





The average silhouette value is computed over all instances in the dataset and provides an overall measure of the clustering quality. A higher average silhouette score indicates better-defined clusters.

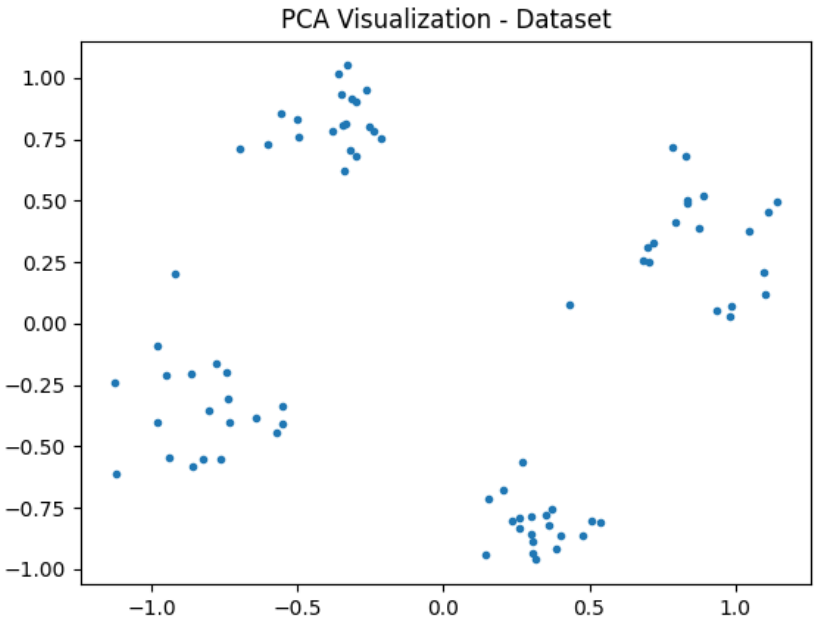
```
Linkage type: single, Distance type: euclidean, K: 2, Silhouette Score: 0.4946
Linkage type: single, Distance type: euclidean, K: 3, Silhouette Score: 0.5929
```

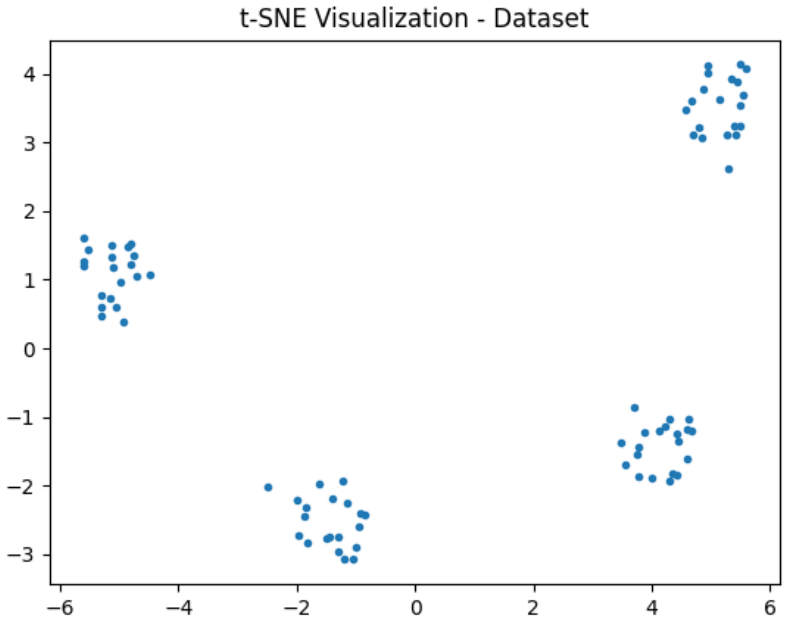
```
Linkage type: single, Distance type: euclidean, K: 4, Silhouette Score: 0.7721

Linkage type: single, Distance type: cosine, K: 2, Silhouette Score: 0.4946
Linkage type: single, Distance type: cosine, K: 3, Silhouette Score: 0.5929
Linkage type: single, Distance type: cosine, K: 4, Silhouette Score: 0.7721
Linkage type: single, Distance type: cosine, K: 5, Silhouette Score: 0.6815

Linkage type: complete, Distance type: euclidean, K: 2, Silhouette Score: 0.4793
Linkage type: complete, Distance type: euclidean, K: 3, Silhouette Score: 0.5894
Linkage type: complete, Distance type: euclidean, K: 4, Silhouette Score: 0.7721
Linkage type: complete, Distance type: euclidean, K: 5, Silhouette Score: 0.6801

Linkage type: complete, Distance type: cosine, K: 2, Silhouette Score: 0.4946
Linkage type: complete, Distance type: cosine, K: 3, Silhouette Score: 0.5929
Linkage type: complete, Distance type: cosine, K: 4, Silhouette Score: 0.7721
Linkage type: complete, Distance type: cosine, K: 5, Silhouette Score: 0.6801
```

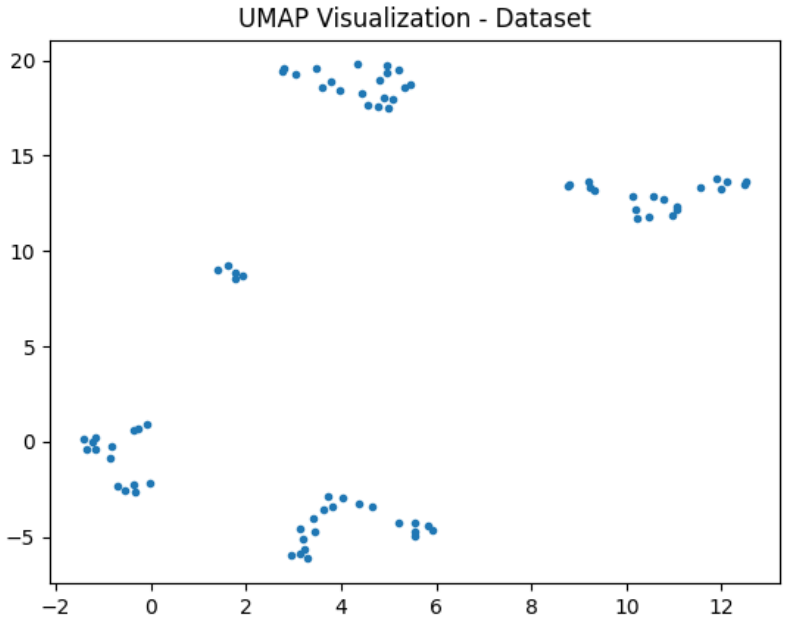
For all the configuration the highest silhouette score is attained at k = 4. At the point k = 4, all the configurations have the same score.

It's more common to observe scores between -1 and 1. Generally, a silhouette score above 0.5 is considered good, and a score above 0.7 is considered excellent. Negative scores indicate overlapping clusters.

(c) **dimension Reduction Plots**







From silhouette score groups we were expecting the reduction plots to have 4 clusters and overall all the plots has 4 clusters even if UMAP visualization has some noise.

(d) **Run Time Analysis for HAC** In each iteration, the algorithm needs to compute the distances or similarities between all pairs of clusters, which $N^2 * D$ where N is the number of data points.
The distance or similarity between all pairs of clusters needs to be recalculated after each merge, the.So, overall time complexity is $I * N^2 * D$ I is the number of iterations, N is the number of data points, and D

is the dimensionality of each data point.

Given a dataset with 1 million data points, each having a dimension of 120,000, using K-means clustering might be more practical. K-means tends to be computationally more efficient than hierarchical agglomerative clustering (HAC) for large datasets due to its linear complexity with respect to the number of data points. The almost cubic time complexity of HAC, especially with such high-dimensional data, could lead to significant computational costs. Because of the memory requirements it might not be possible to use HAC.