Part 1

Hyperparameters:

K was given 3 different values: 3, 4, 5
Distance metrics were: Cosine similarity, Minkowski, Mahalanobis
This makes 3*3 = 9 configurations in total.

Accuracies and confidence intervals for each configuration:

K: 3
distance_metric: cos
mean_accuracy: 0.95
confidence_interval: ('0.91', '0.99')
K: 3
distance_metric: minkowski
mean_accuracy: 0.94
confidence_interval: ('0.88', '1.00')
K: 3
distance_metric: mahalanobis
mean_accuracy: 0.90
confidence_interval: ('0.84', '0.96')
K: 4
distance_metric: cos
mean_accuracy: 0.94
confidence_interval: ('0.89', '0.99')
K: 4
distance_metric: minkowski
mean_accuracy: 0.93
confidence_interval: ('0.88', '0.99')
K: 4
distance_metric: mahalanobis
mean_accuracy: 0.90
confidence_interval: ('0.83', '0.97')
K: 5
distance_metric: cos
mean_accuracy: 0.94
confidence_interval: ('0.88', '1.00')

K: 5
distance_metric: minkowski
mean_accuracy: 0.94
confidence_interval: ('0.88', '0.99')
K: 5
distance_metric: mahalanobis
mean_accuracy: 0.88
confidence_interval: ('0.81', '0.96')

Best hyperparameter values:

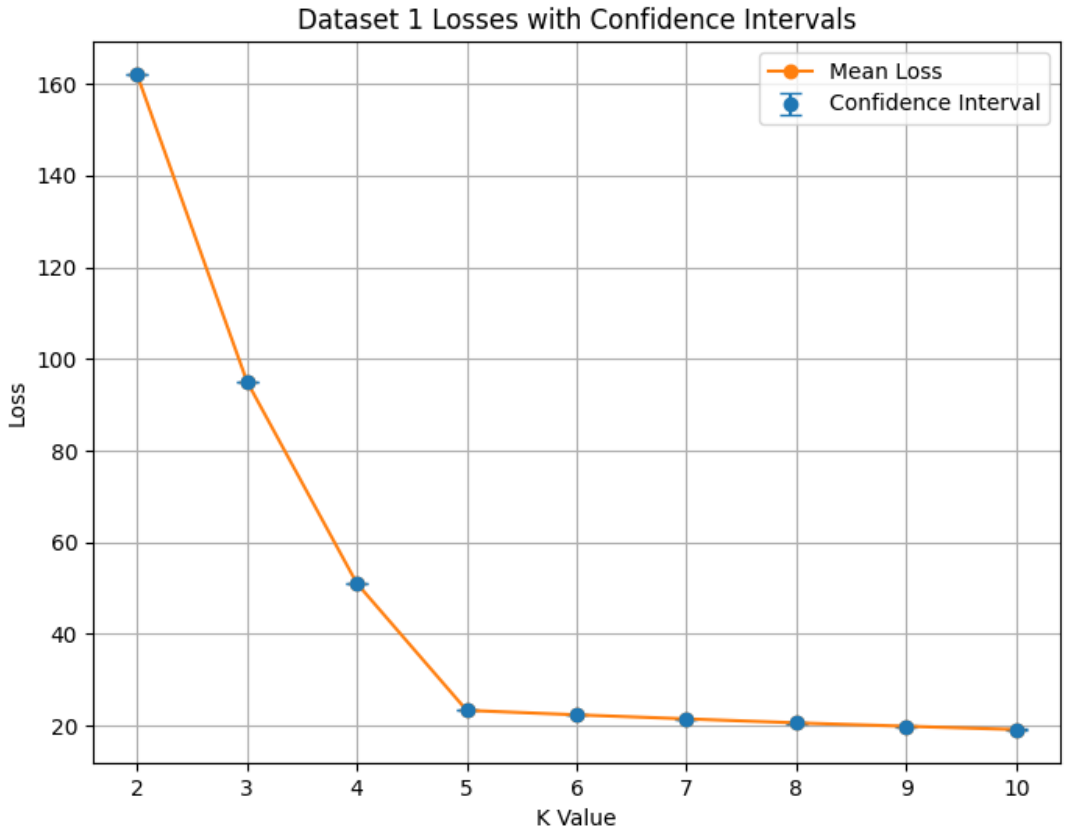
K = 3
Distance metric = Cosine similarity
I chose these as the best hyperparameter values because they resulted in the best mean accuracy(0.95) with a 95% confidence interval of (0.91, 0.99).

Part 2

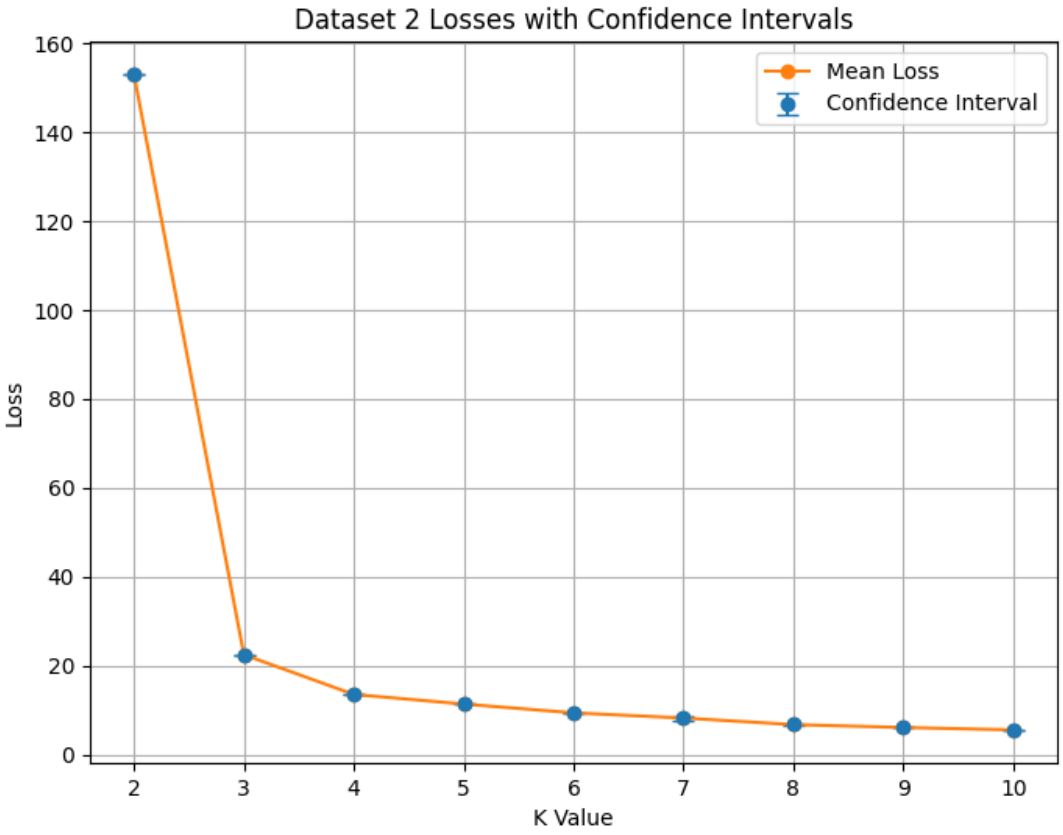
K versus Loss Graphs

KMeans

Dataset1

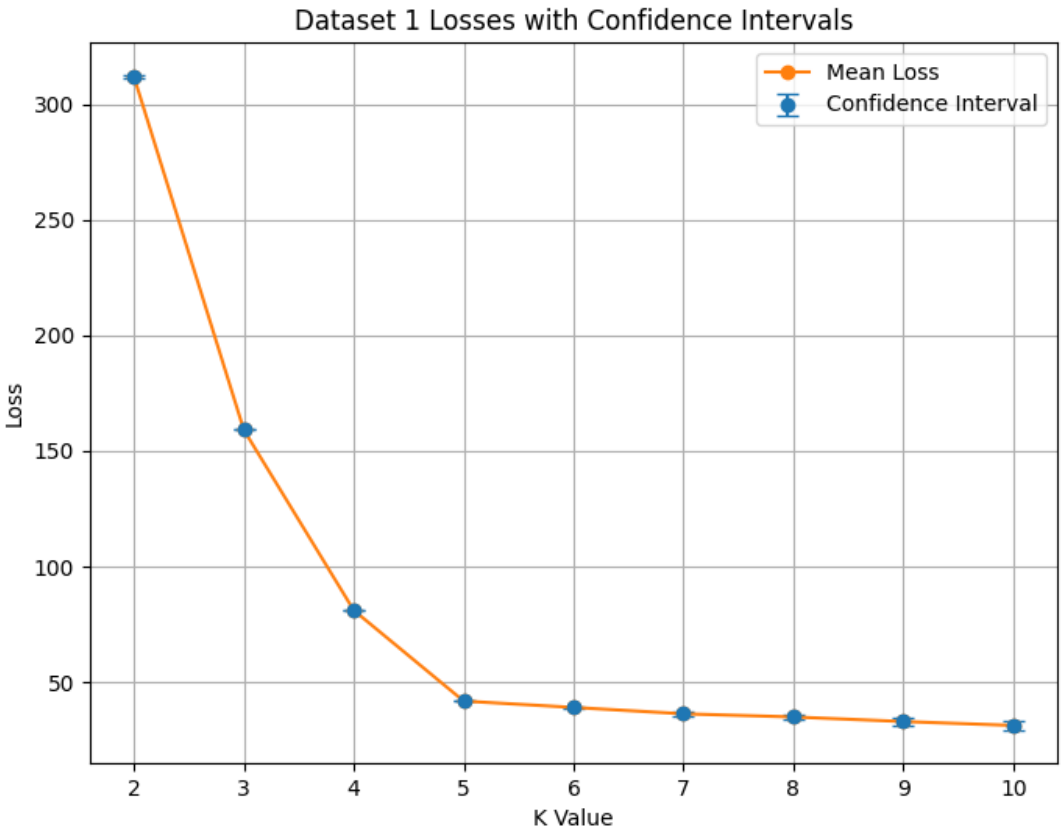


Dataset2

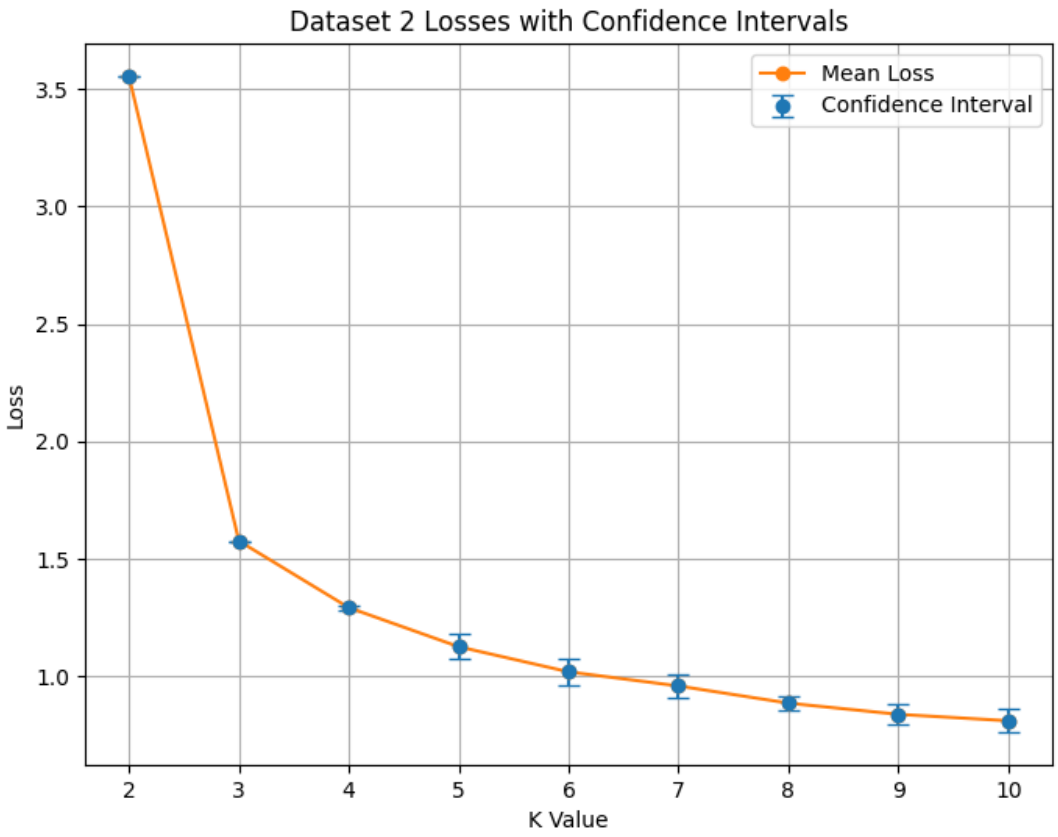


KMedoids

Dataset 1


Dataset 1 Losses with Confidence Intervals
Mean Loss
Confidence Interval
Loss
K Value
300
250
200
150
100
50
2
3
4
5
6
7
8
9
10

Dataset 2



Comments

For the KMeans algorithm, the K-Loss graph for dataset 1 indicates a **K value of 5** is best suited when the Elbow Method is used(at K = 5, we no longer see substantial loss decrease, and the graph takes the shape of an elbow).
For dataset 2, following a similar logic, **K equal to 3** is the best choice.

For the KMedoids algorithm, following the same logic as KMeans, the best choice for **K for dataset 1 is 5**.
For dataset 2, **it is 3**.

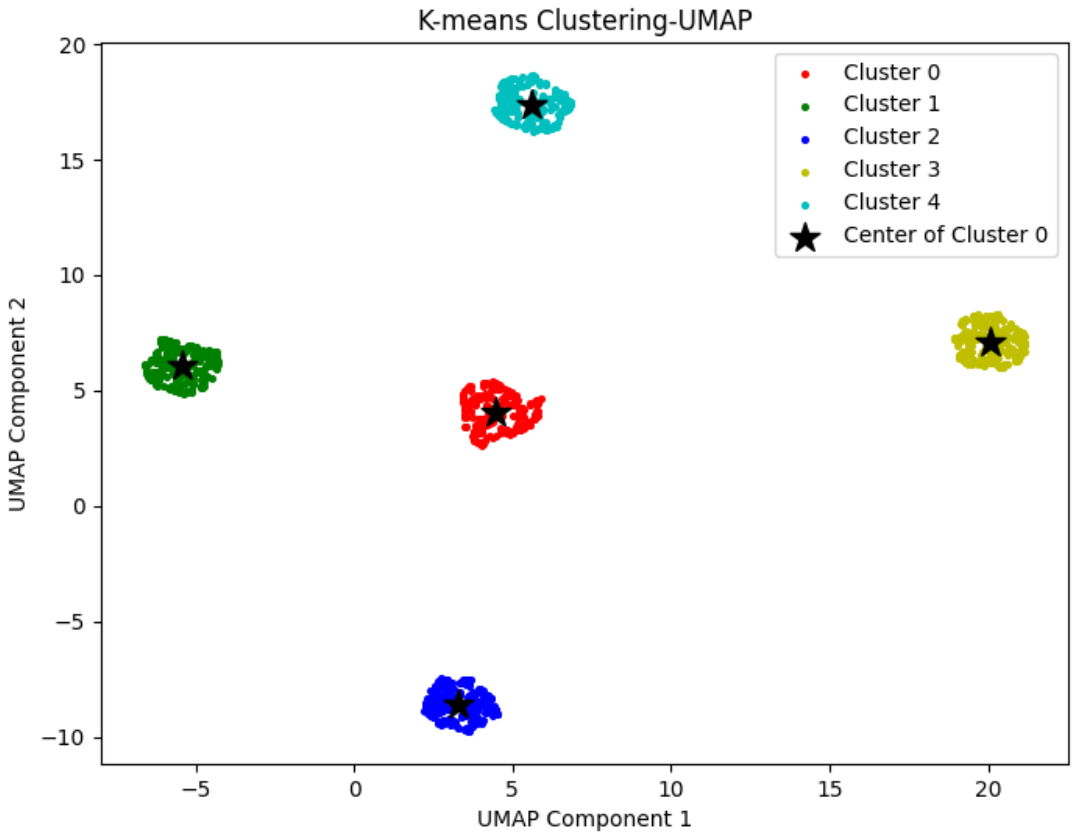
Dimensionality Reduction Graphs

KMeans

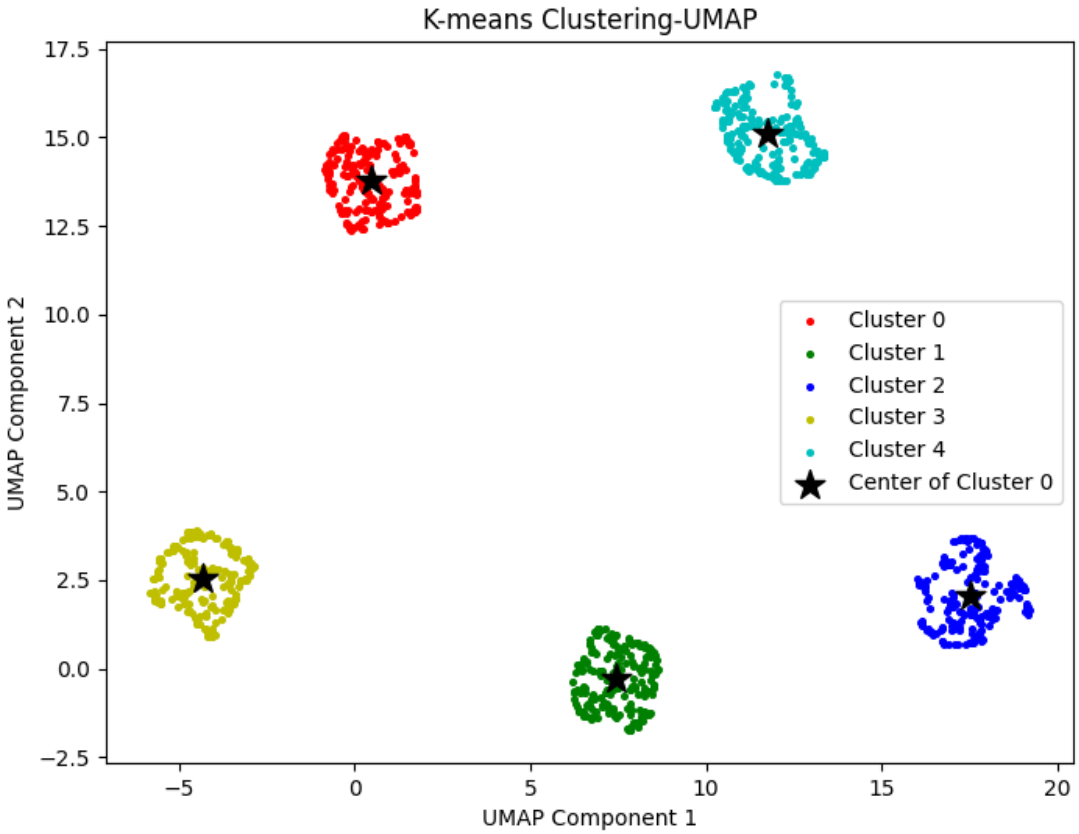
Dataset 1

UMAP

Metric: Euclidean



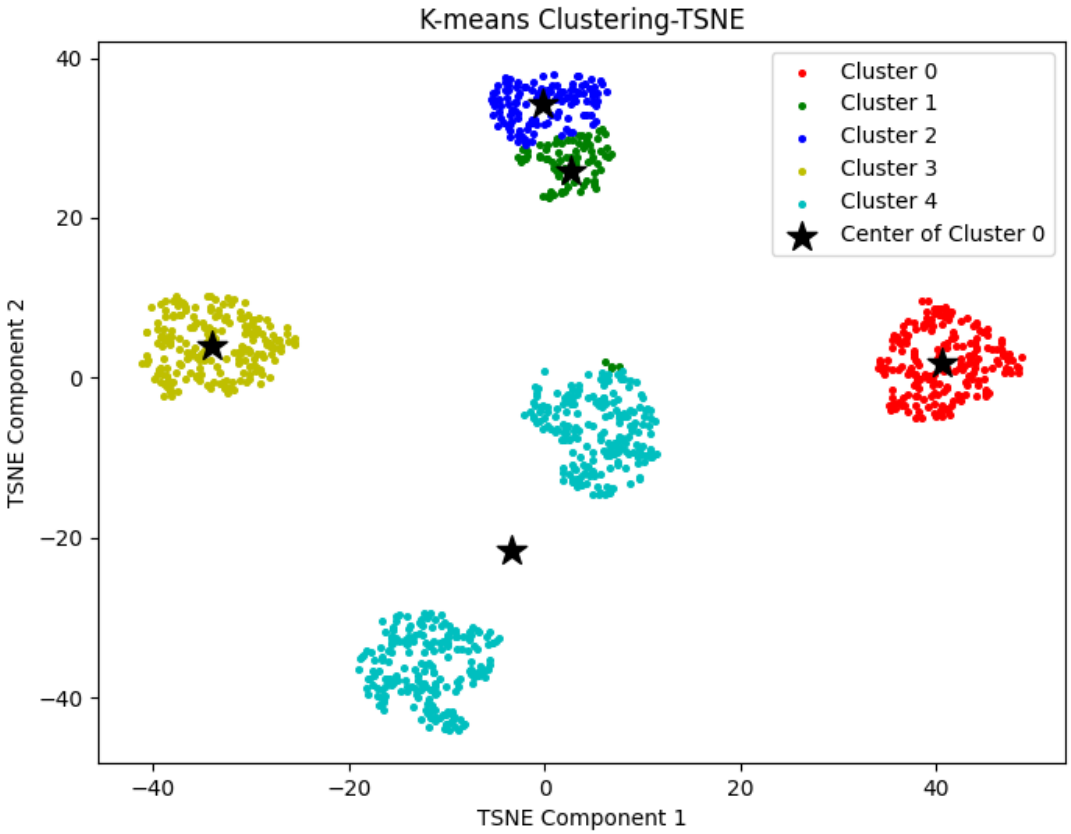
Metric: Cosine



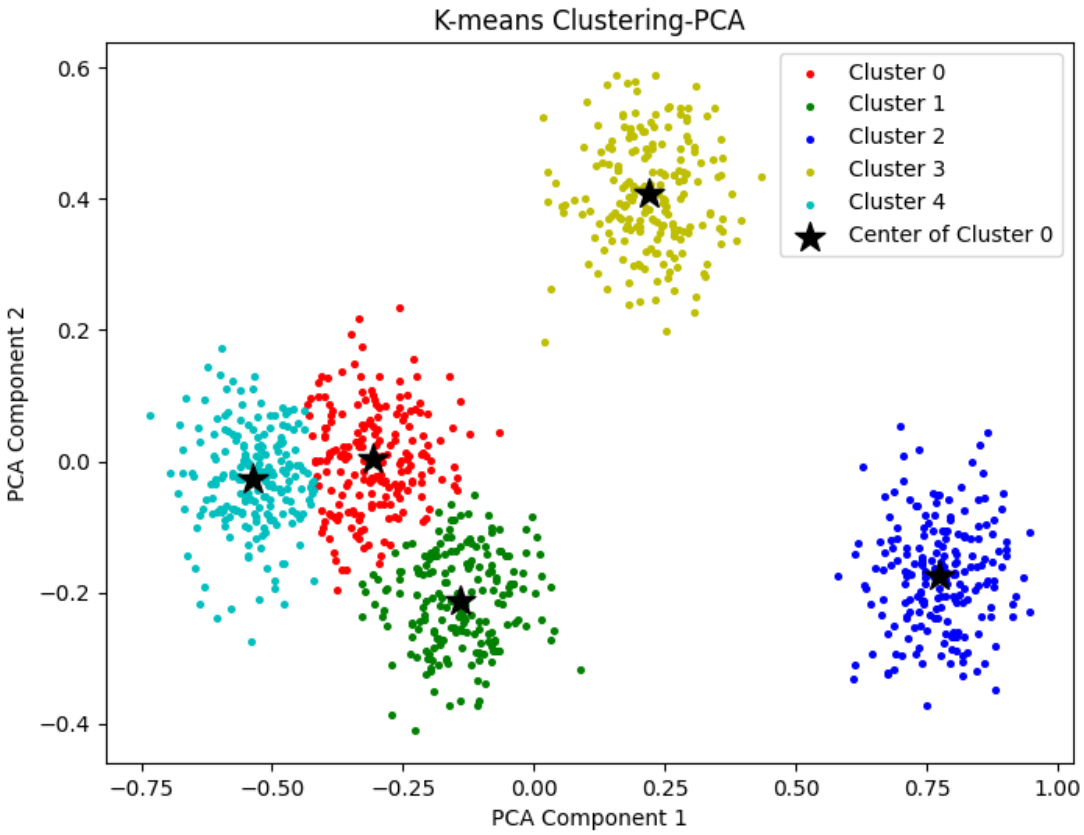
t-SNE

Metric: Euclidean

Metric: Cosine



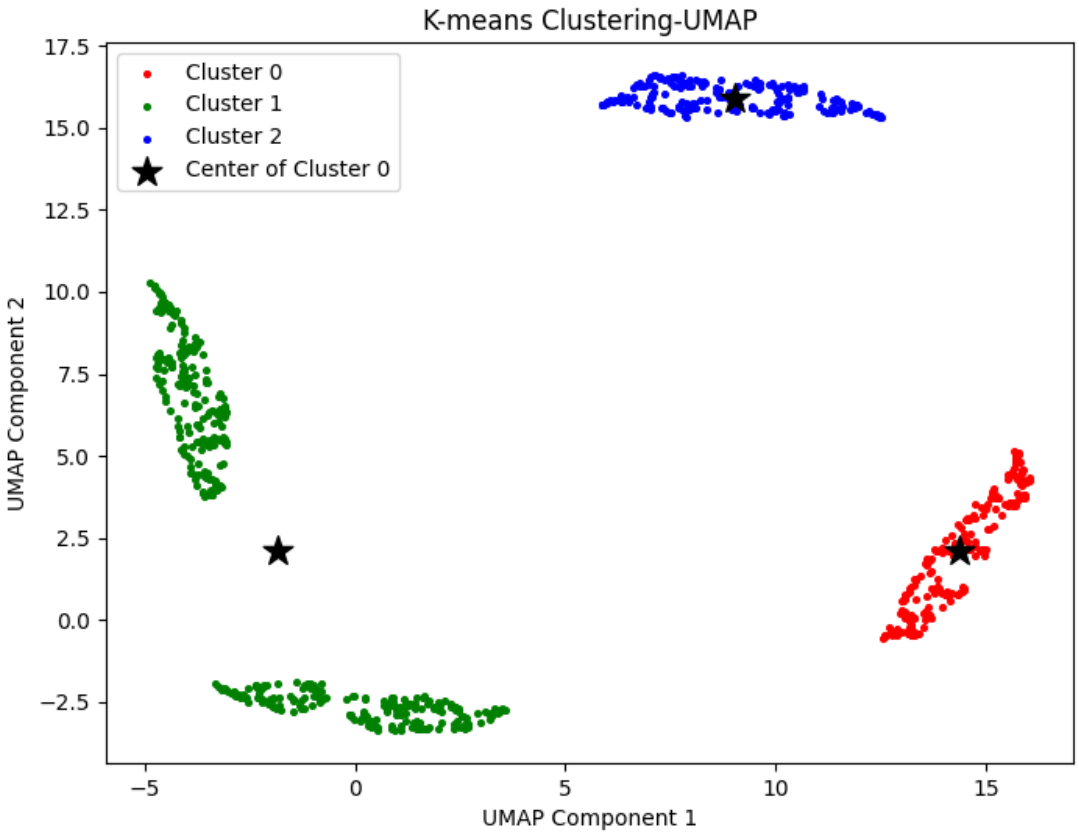
PCA



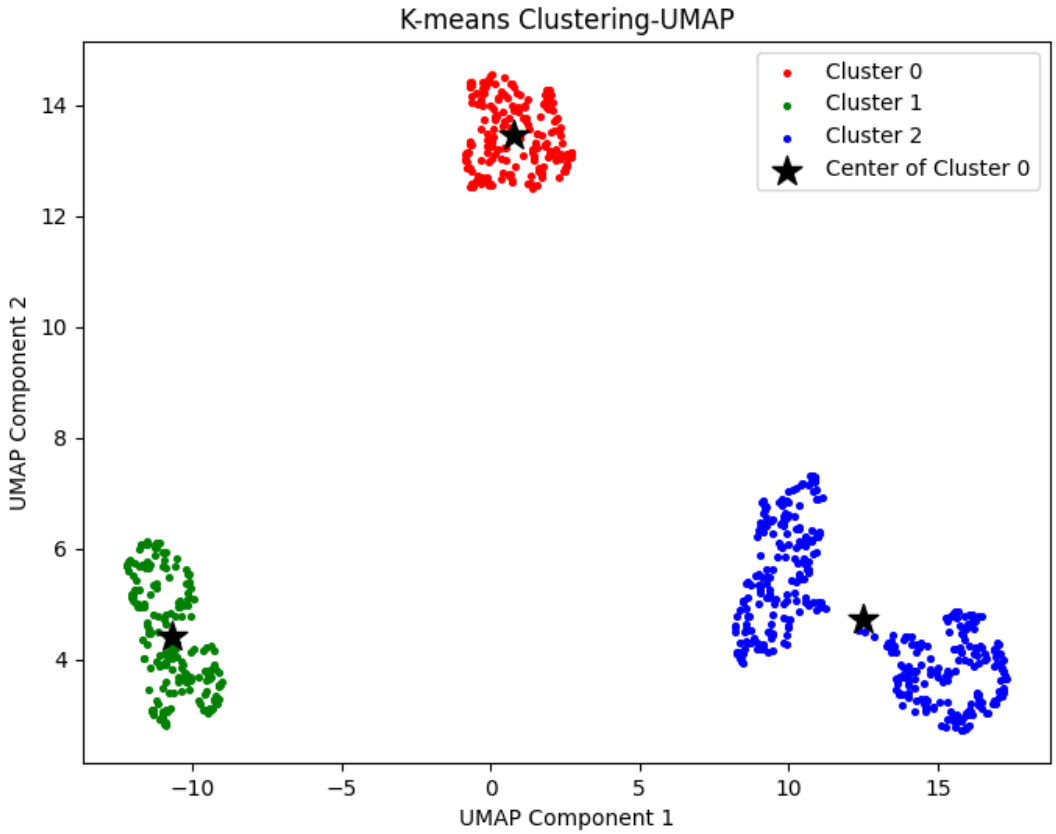
Dataset 2

UMAP

Metric: Euclidean

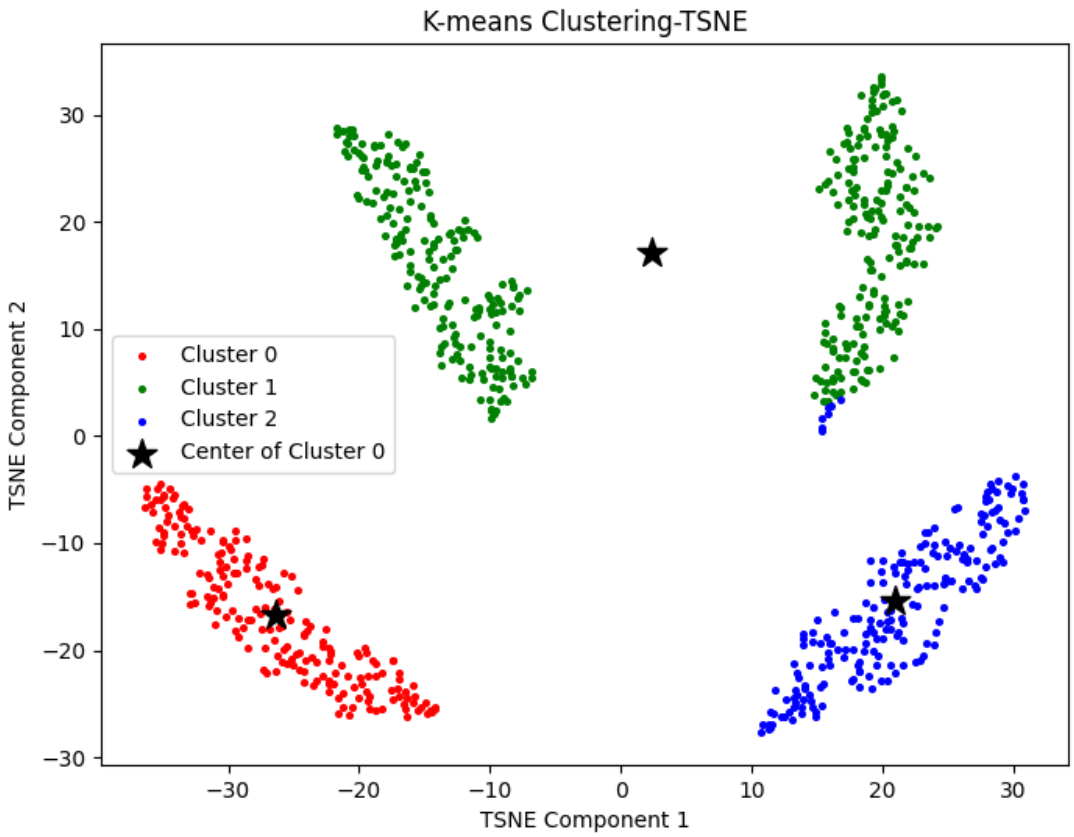

K-means Clustering-UMAP
Cluster 0
Cluster 1
Cluster 2
Center of Cluster 0
UMAP Component 2
UMAP Component 1
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5
0
5
10
15

Metric: Cosine

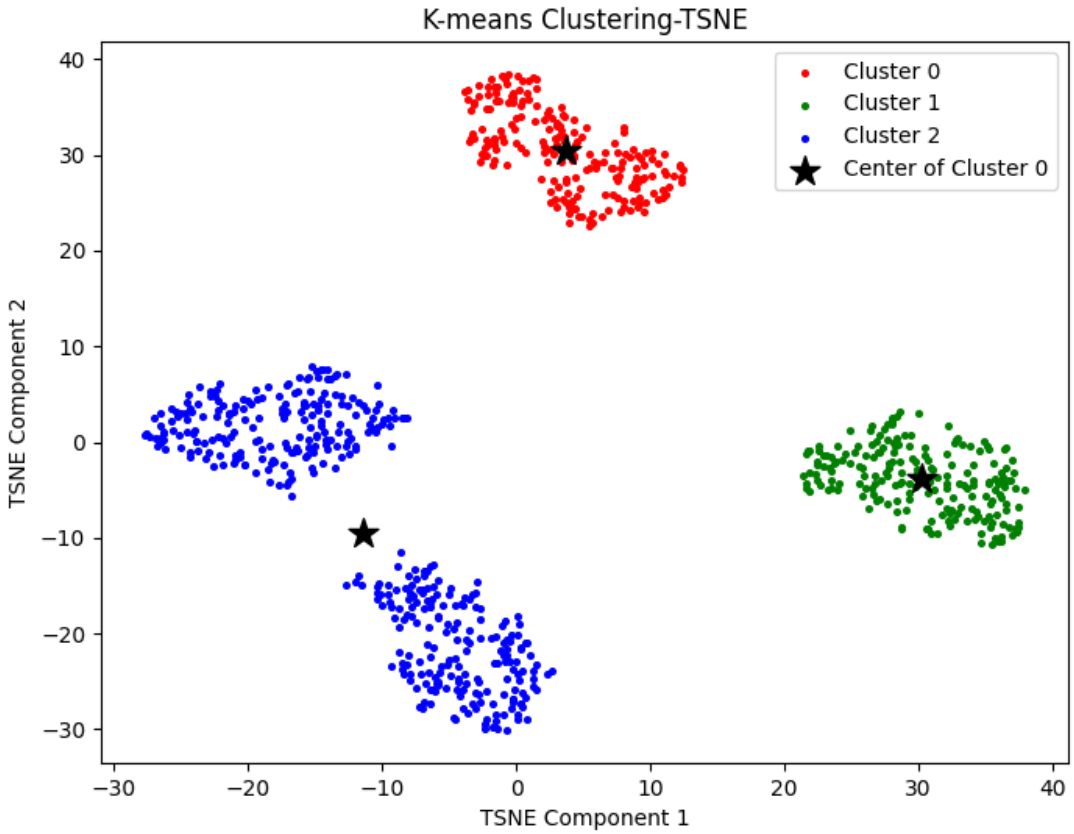


t-SNE

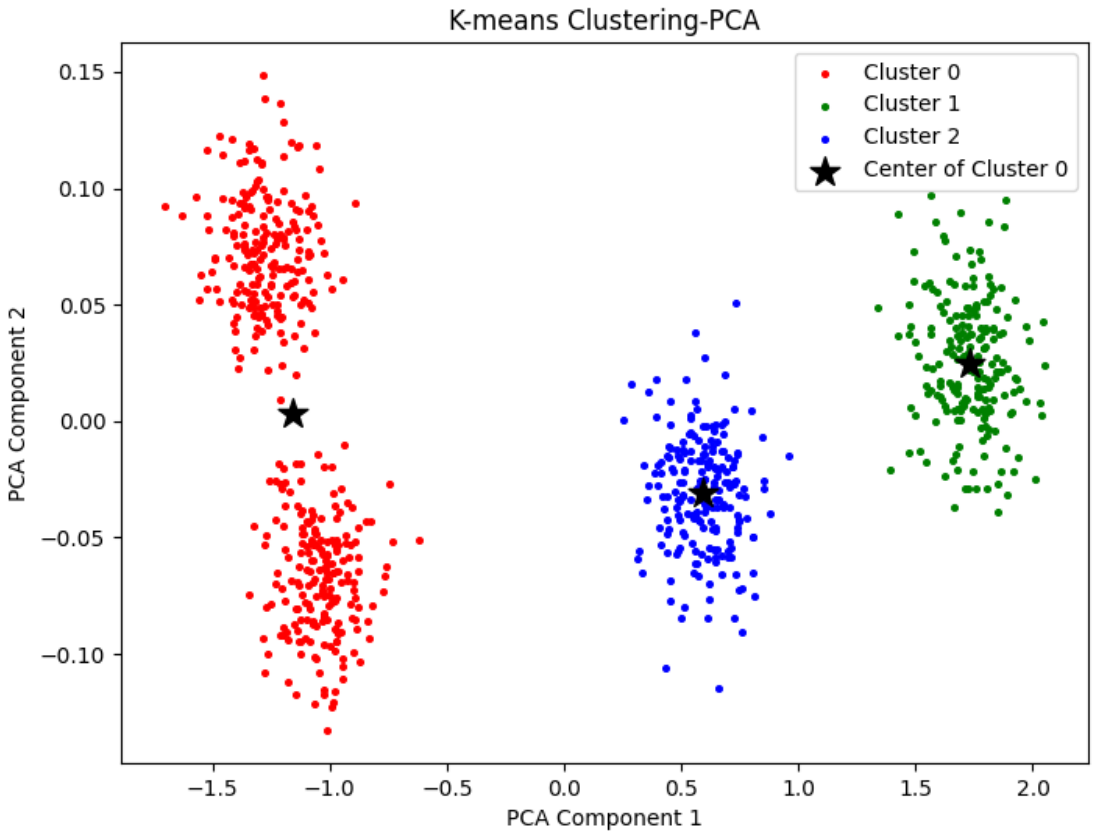
Metric: Euclidean



Metric: Cosine



PCA

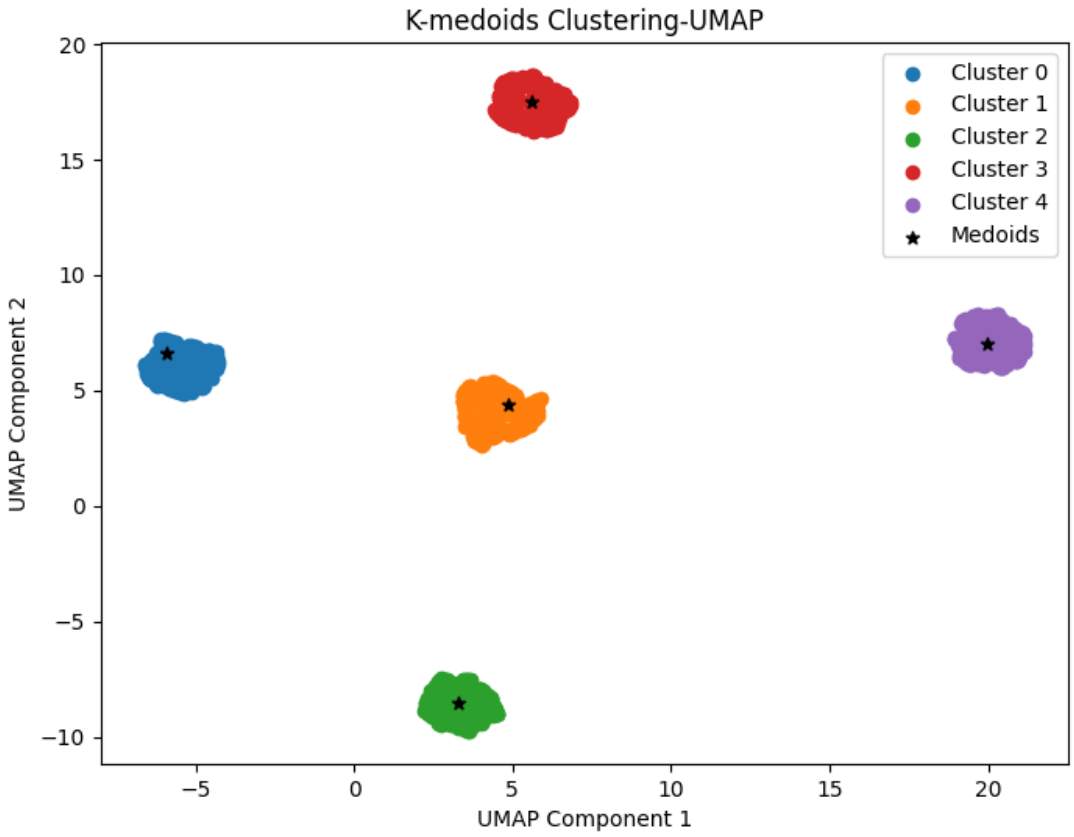

K-means Clustering-PCA
Cluster 0
Cluster 1
Cluster 2
Center of Cluster 0
PCA Component 2
PCA Component 1
0.15
0.10
0.05
0.00
−0.05
−0.10
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

KMedoids

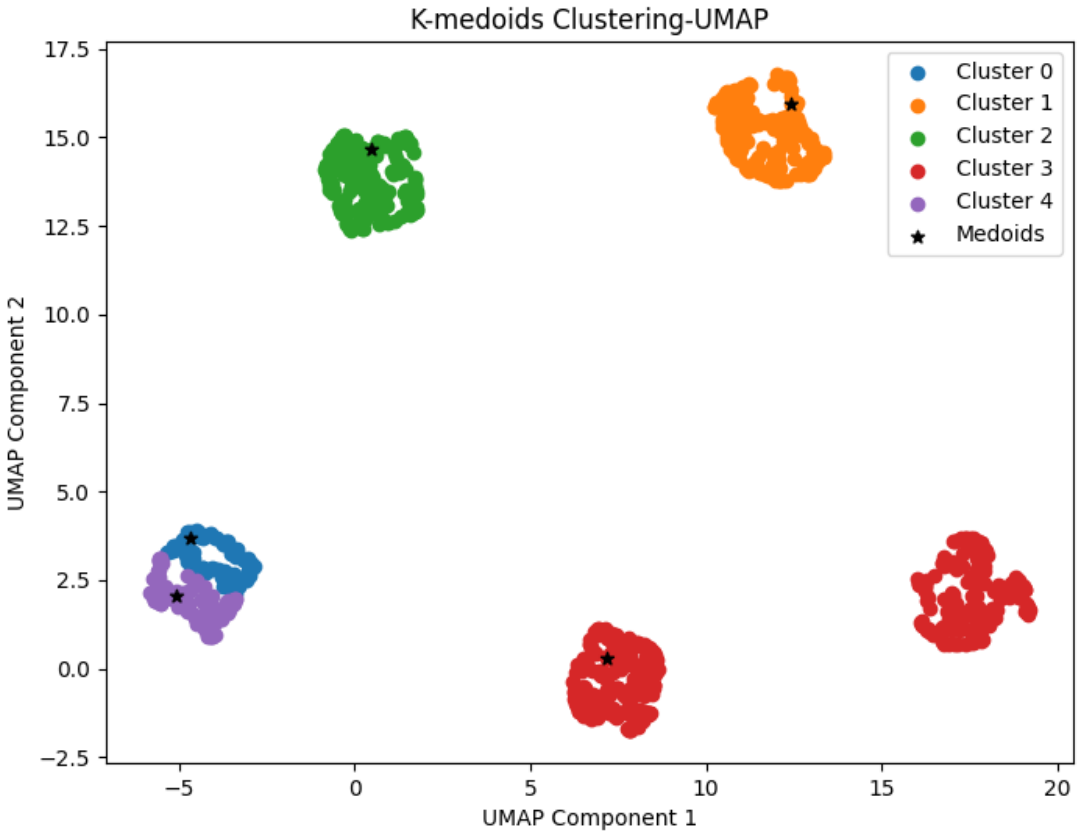
Dataset 1

UMAP

Metric: Euclidean

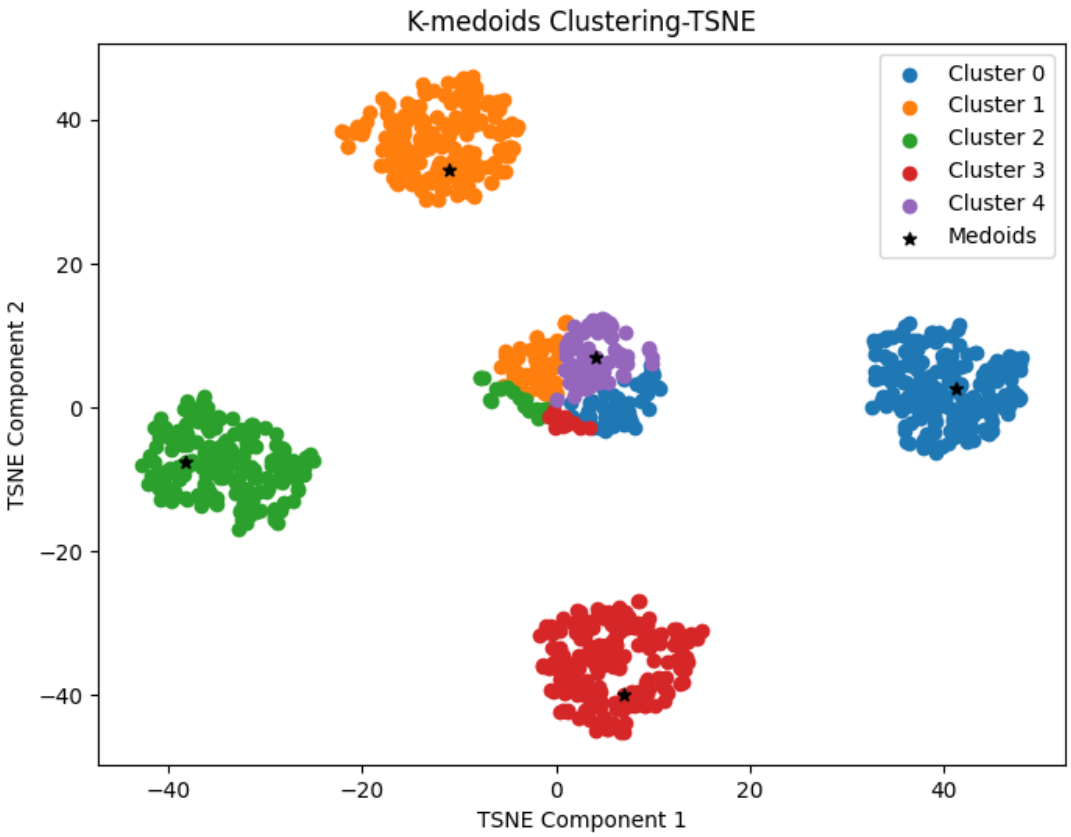


Metric: Cosine

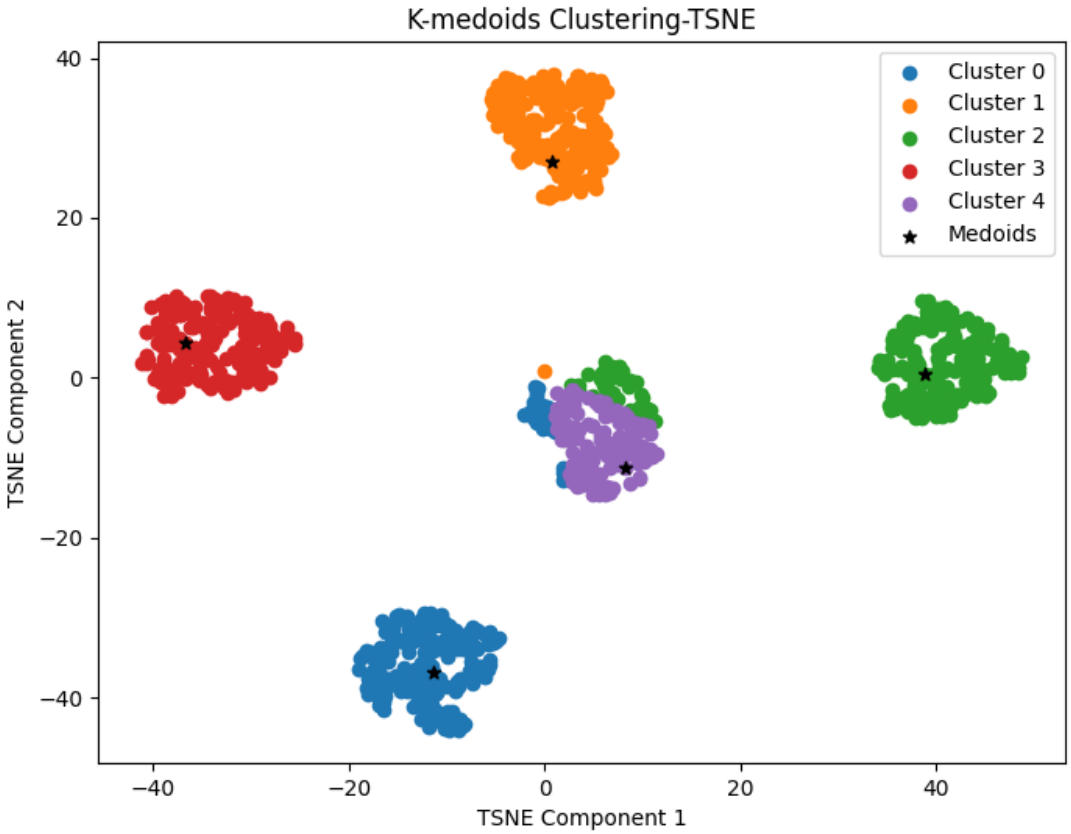


t-SNE

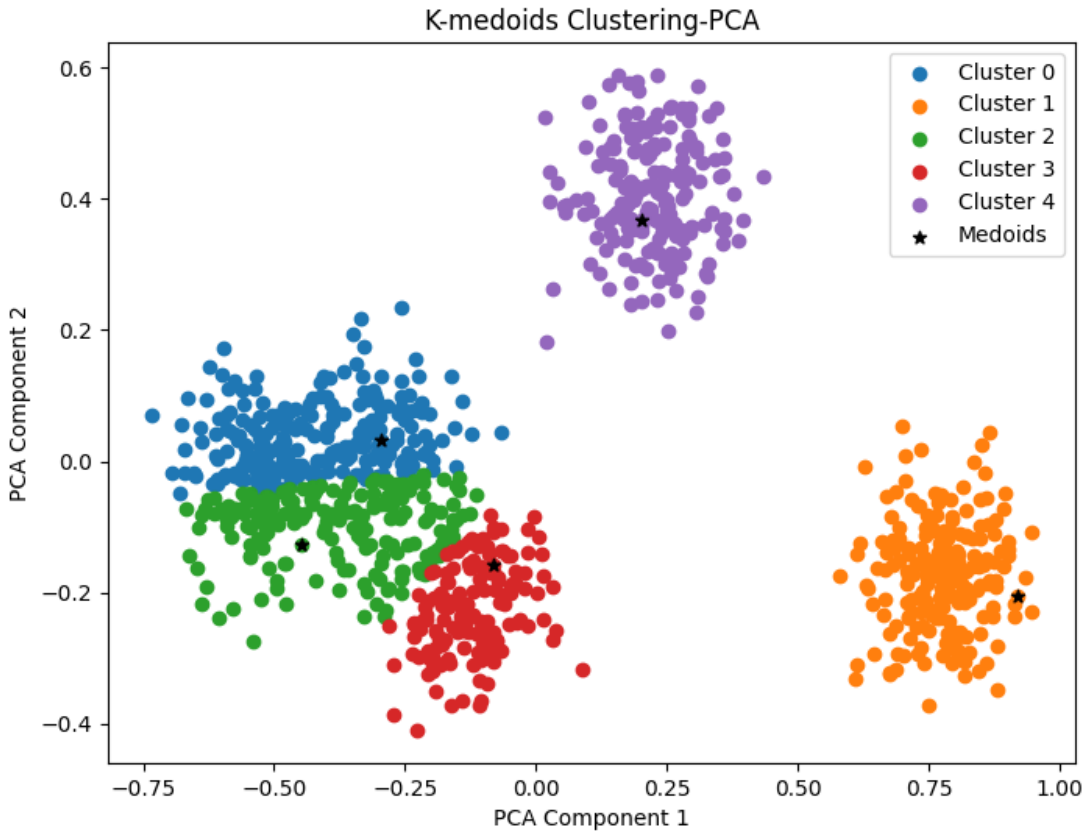
Metric: Euclidean



Metric: Cosine



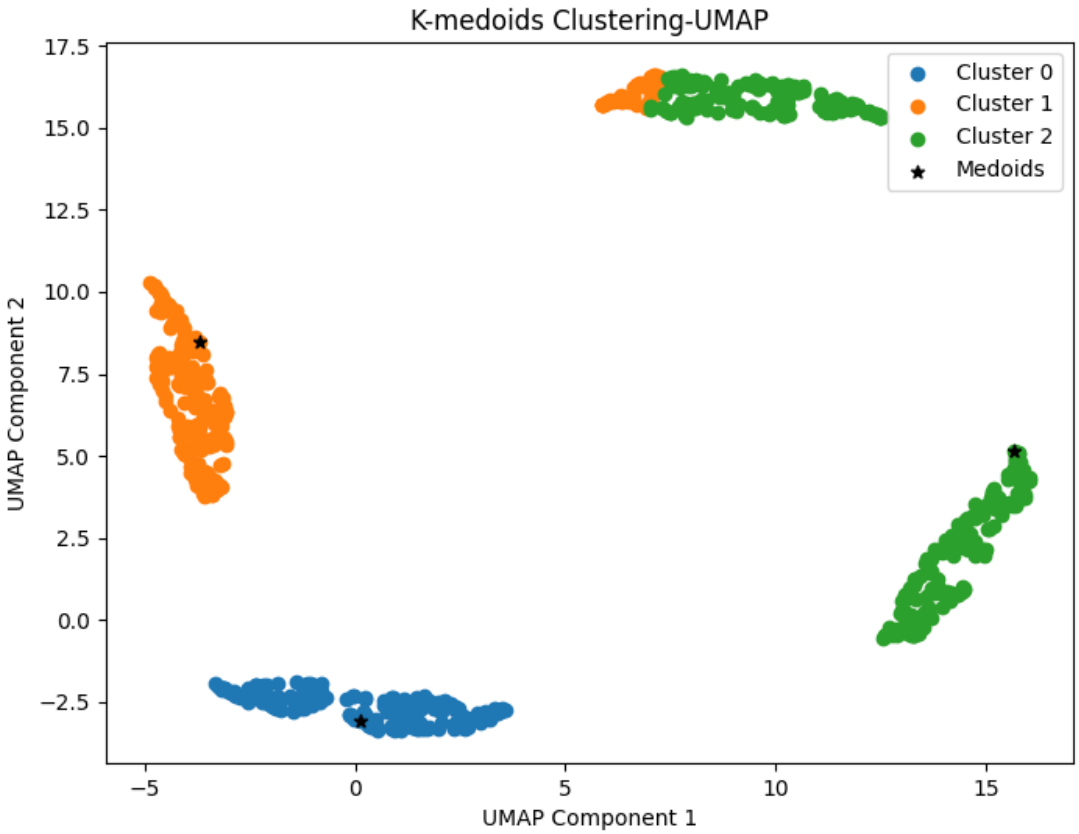
PCA



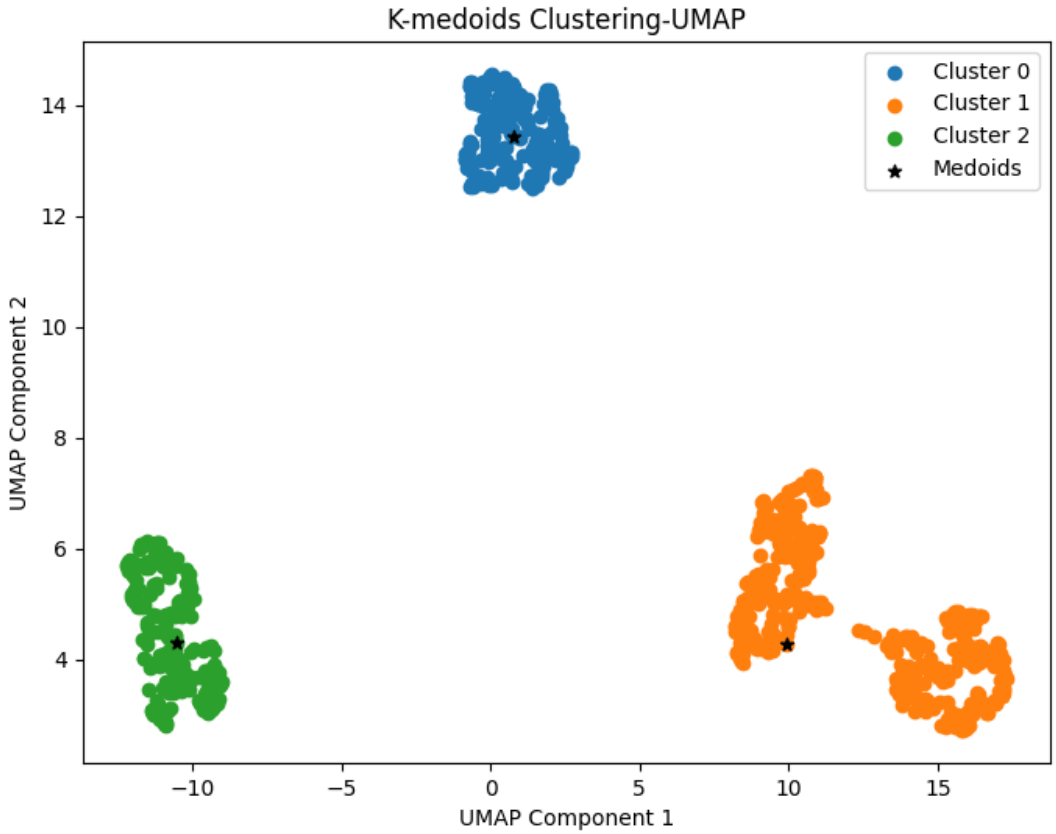
Dataset 2

UMAP

Metric: Euclidean

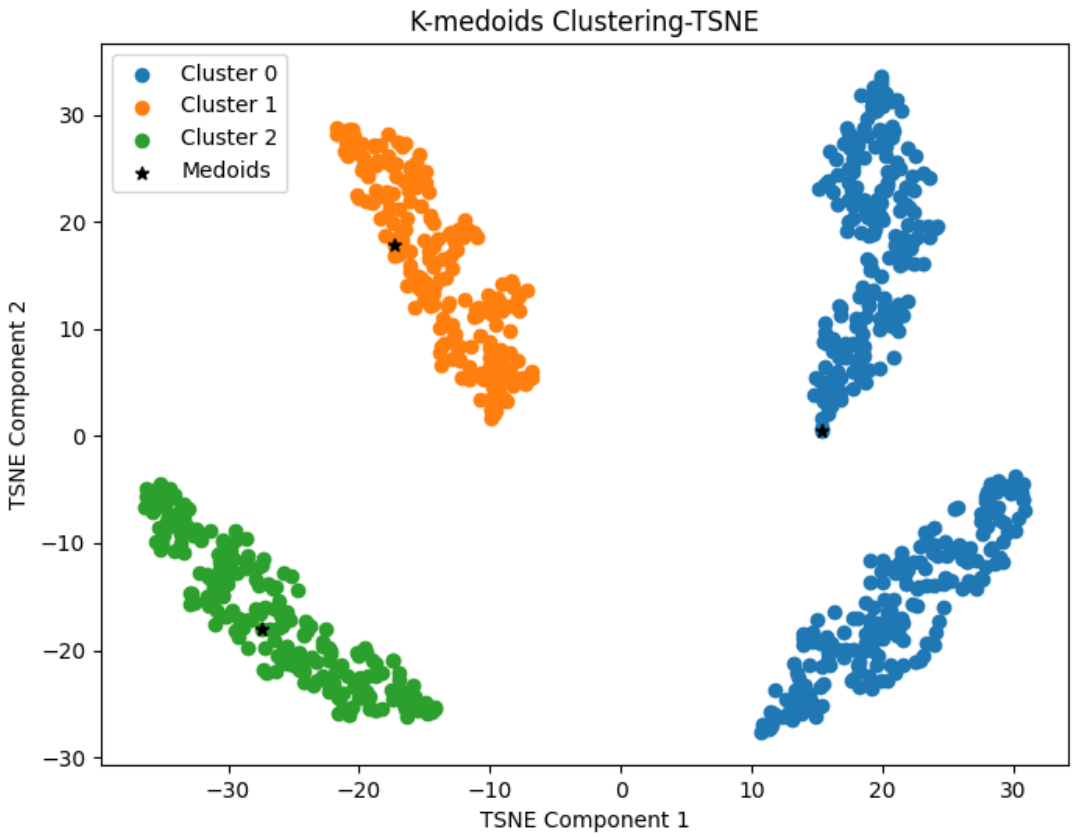

K-medoids Clustering-UMAP
Cluster 0
Cluster 1
Cluster 2
Medoids
UMAP Component 2
UMAP Component 1
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5
0
5
10
15

Metric: Cosine

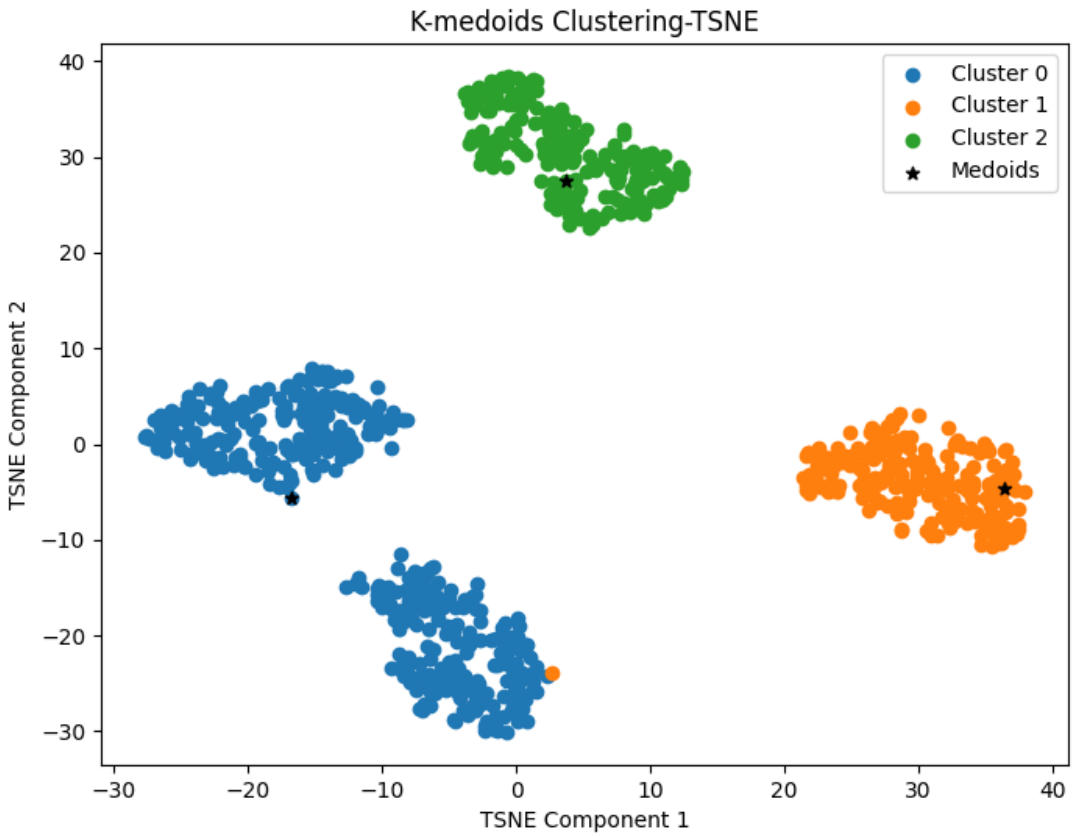


t-SNE

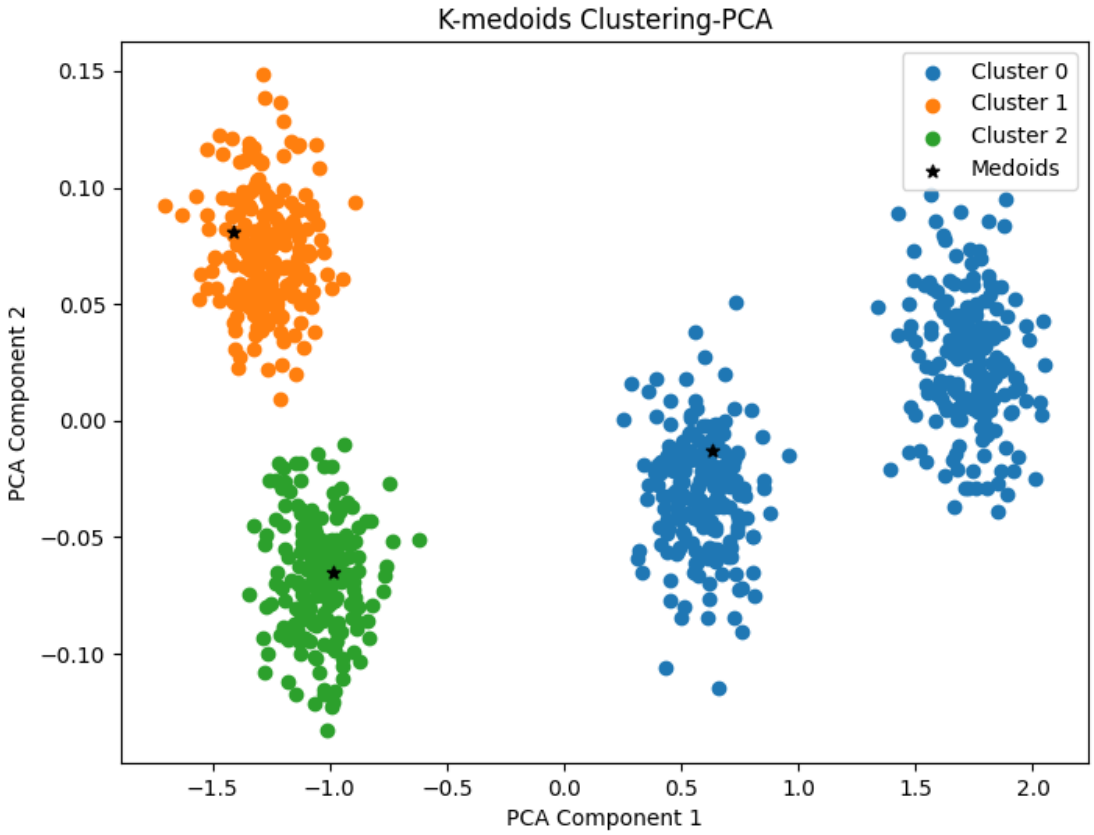
Metric: Euclidean



Metric: Cosine



PCA



Comments

Best Results

I didn't notice too much difference between the methods I employed.

t-SNE and UMAP dimensionality reduction methods gave good results.

The same is true for which metrics I used as well(Cosine and Euclidean).

All dimensionality reduction methods and hyperparameters resulted in good clusters. However, I noticed that the PCA method would sometimes produce not-so-distinct clusters.

So, in conclusion, I would employ t-SNE and UMAP instead of PCA for dimensionality reduction.

Number of Clusters

I identified K as 5 for dataset 1, and the visualized points obtained by using the dimensionality reduction methods confirmed K = 5 as the best choice.

However, for dataset 2, the visualization showed that K = 4 should've been the correct choice(I chose 3 instead).

Worst-case Running Time Analysis

KMeans:

Computing distance between a data point and a cluster center across d dimensions, which is O(d).

Repeating this for N data points is O(N * d).

For K centers, it is O(N * K * d).

For I iterations, it is

O(I * N * K * d)

KMedoids:

Computing distance for all dimensions d from a medoid to N other data points is O(N * d).

There can be at most N many potential medoids. Then, complexity becomes O(N * N * d).

Repeating this for K clusters: O(K * N^2 * d)

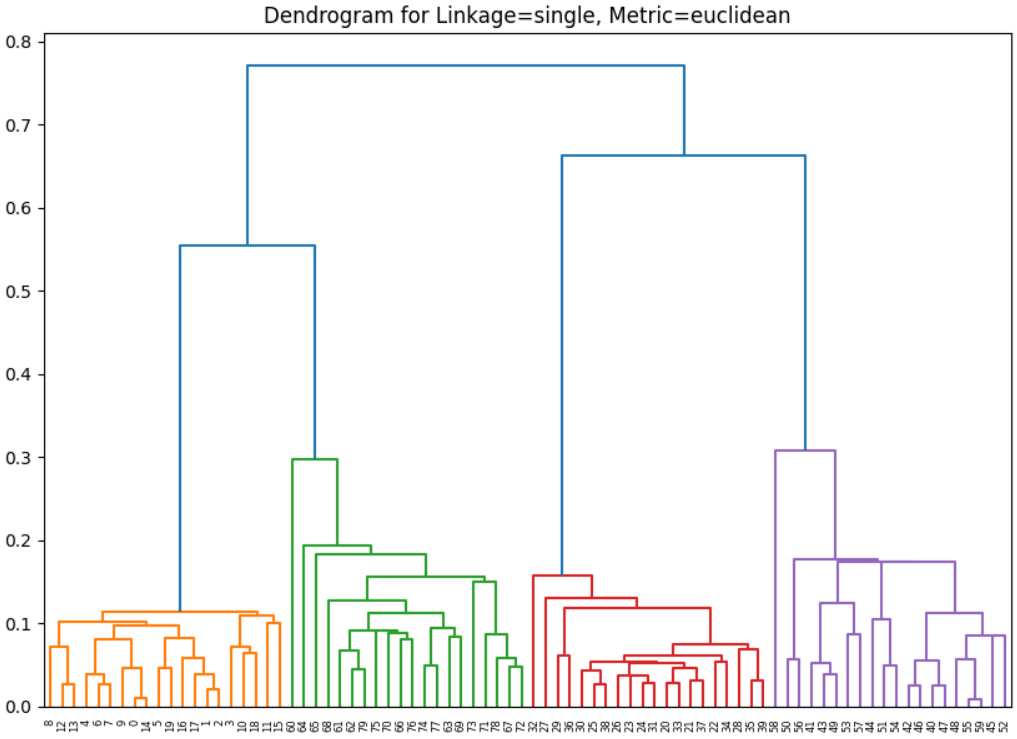
For I iterations:

O(I * K * N^2 * d)

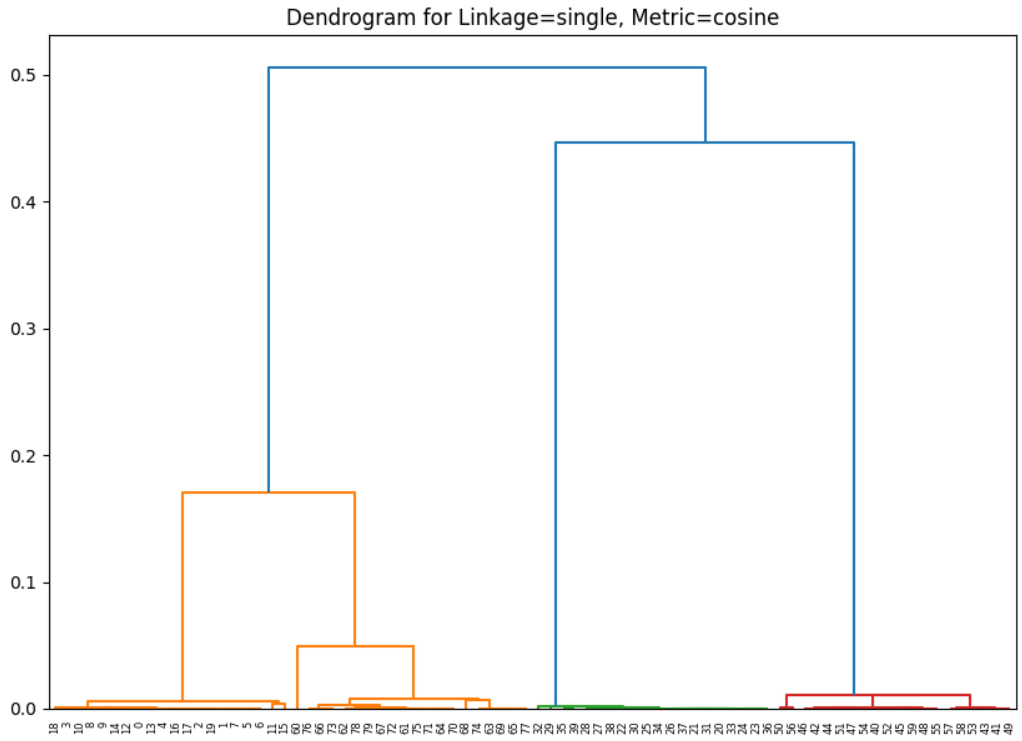
Part 3

Dendrograms

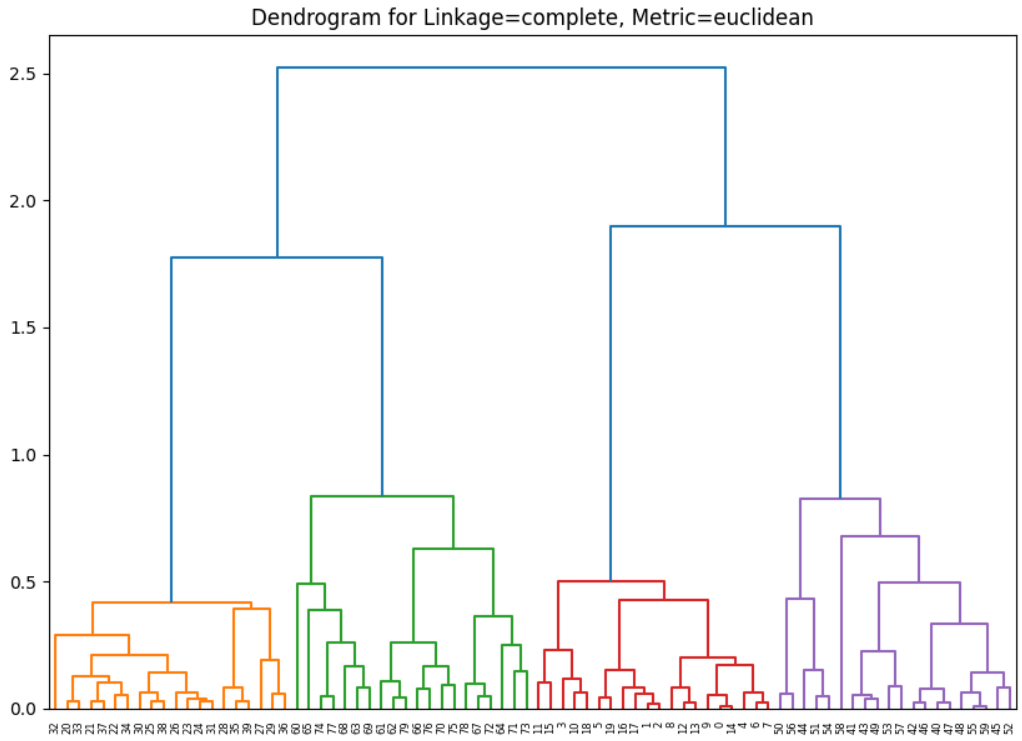
Linkage: Single & Metric: Euclidean



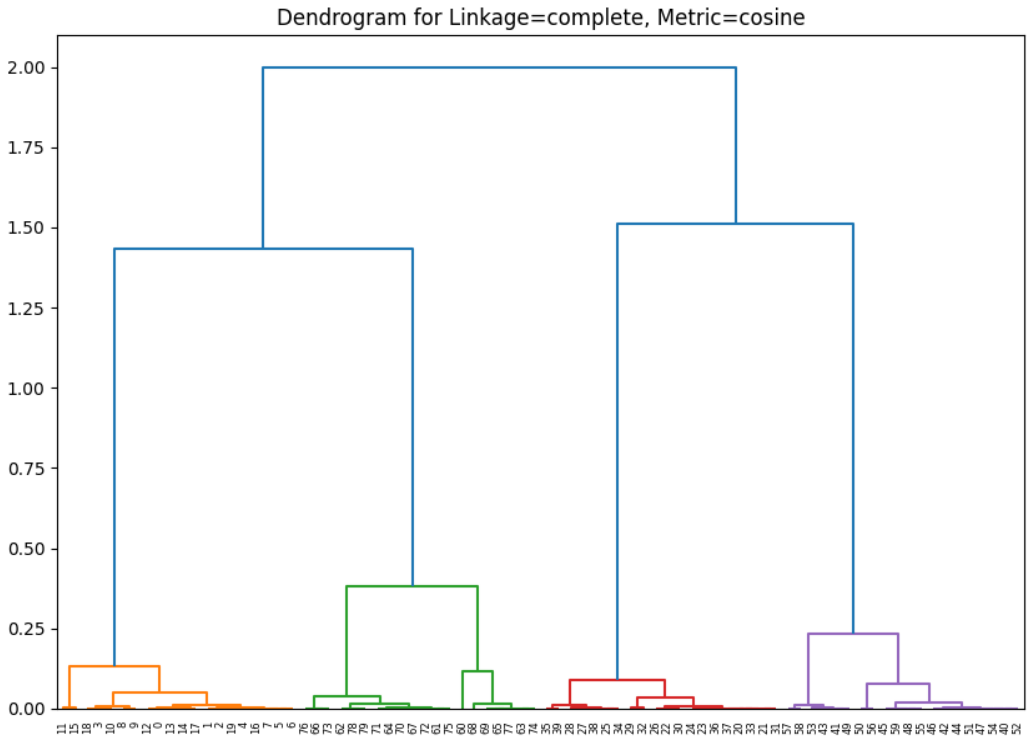
Linkage: Single & Metric: Cosine



Linkage: Complete & Metric: Euclidean



Linkage: Complete & Metric: Cosine



Silhouette Analysis

The best configurations are highlighted in bold text below:

For linkage: single, metric: euclidean, K: 2 the average silhouette_score is: 0.49462610483169556
For linkage: single, metric: euclidean, K: 3 the average silhouette_score is: 0.5929003953933716
For linkage: single, metric: euclidean, K: 4 the average silhouette_score is: 0.7720839977264404
For linkage: single, metric: euclidean, K: 5 the average silhouette_score is: 0.670854389667511

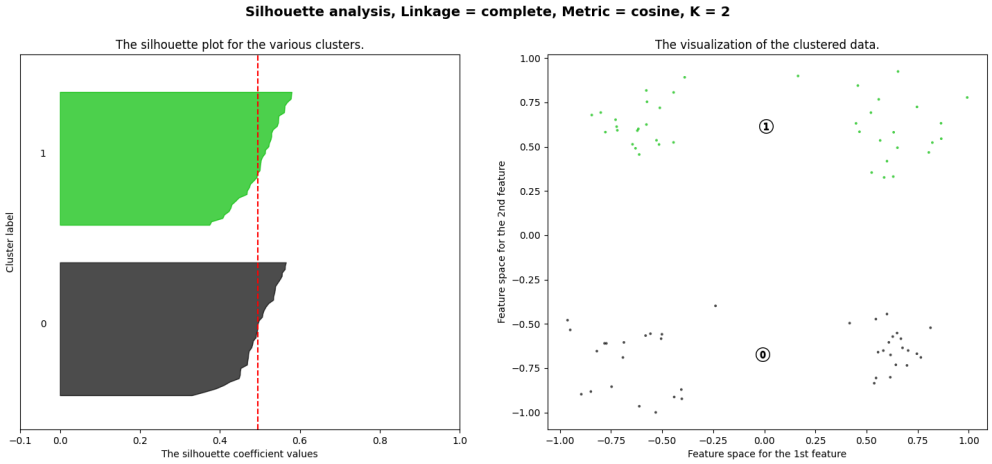
For linkage: single, metric: cosine, K: 2 the average silhouette_score is: 0.49462610483169556
For linkage: single, metric: cosine, K: 3 the average silhouette_score is: 0.5929003953933716
For linkage: single, metric: cosine, K: 4 the average silhouette_score is: 0.7720839977264404
For linkage: single, metric: cosine, K: 5 the average silhouette_score is: 0.6814747452735901

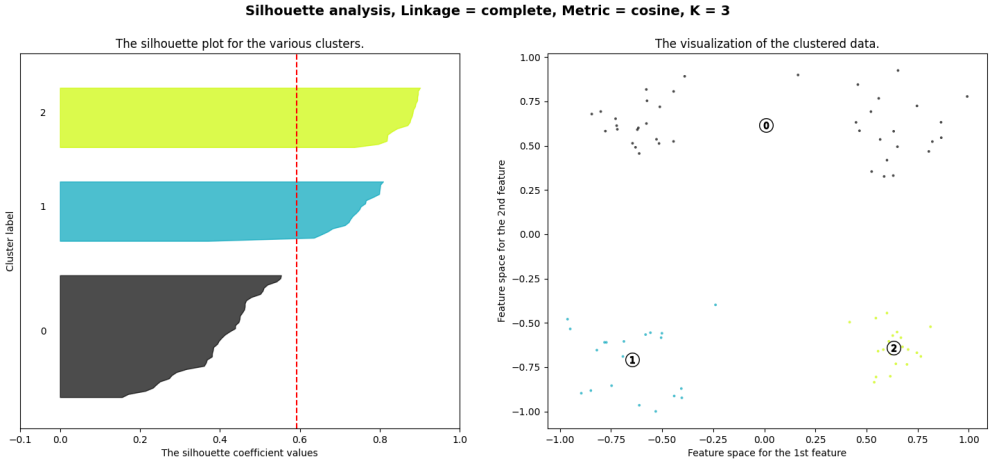
For linkage: complete, metric: euclidean, K: 2 the average silhouette_score is: 0.47934556007385254
For linkage: complete, metric: euclidean, K: 3 the average silhouette_score is: 0.5894443392753601
For linkage: complete, metric: euclidean, K: 4 the average silhouette_score is: 0.7720839977264404
For linkage: complete, metric: euclidean, K: 5 the average silhouette_score is: 0.6800521016120911

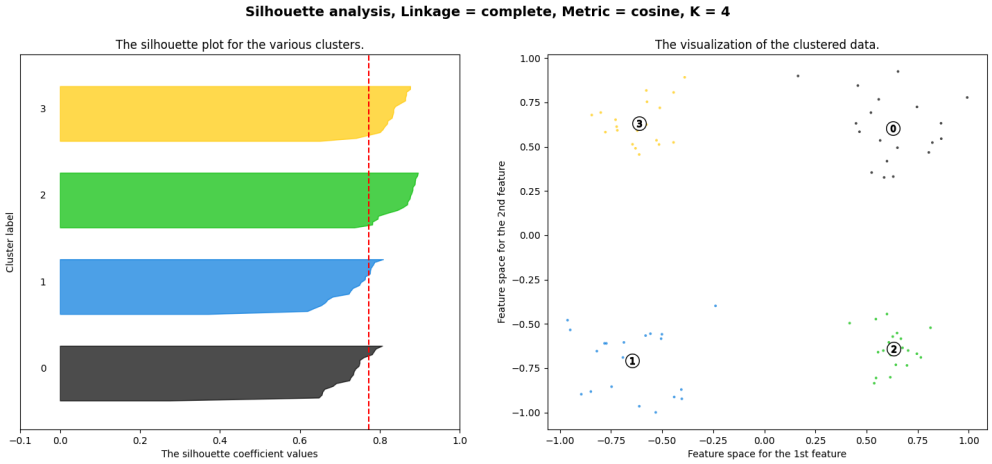
For linkage: complete, metric: cosine, K: 2 the average silhouette_score is: 0.49462610483169556
For linkage: complete, metric: cosine, K: 3 the average silhouette_score is: 0.5929003953933716
For linkage: complete, metric: cosine, K: 4 the average silhouette_score is: 0.7720839977264404
For linkage: complete, metric: cosine, K: 5 the average silhouette_score is: 0.6800521016120911

For all the configurations, **K = 4** attains the highest average silhouette score.

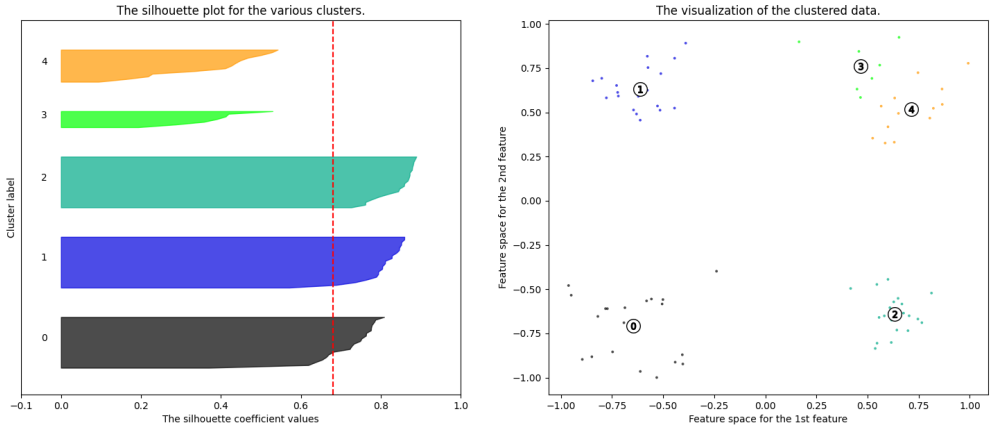
Silhouette Graphs



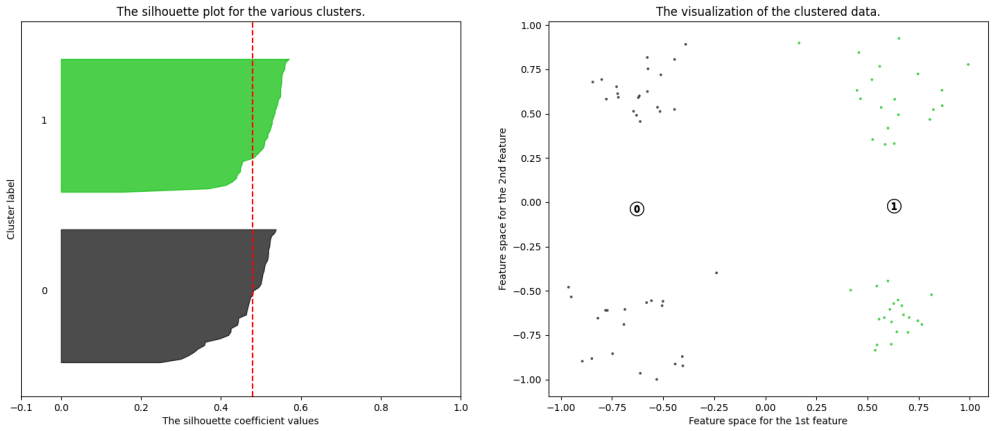




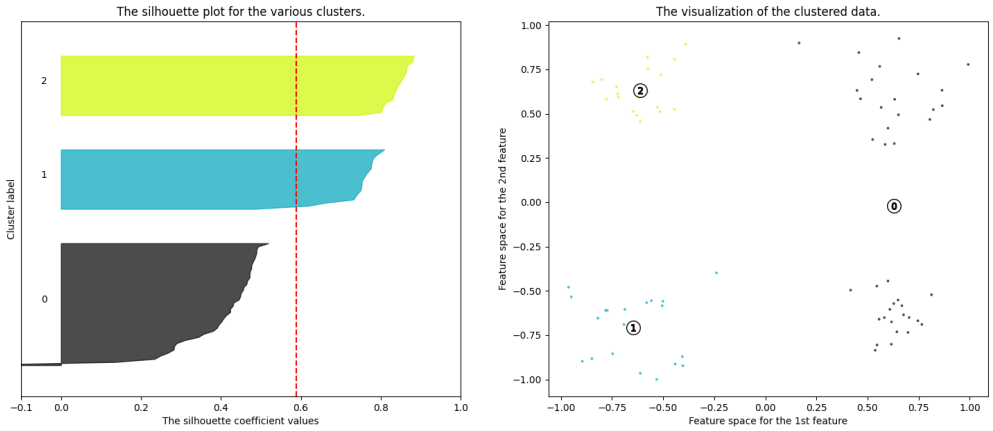
Silhouette analysis, Linkage = complete, Metric = cosine, K = 5



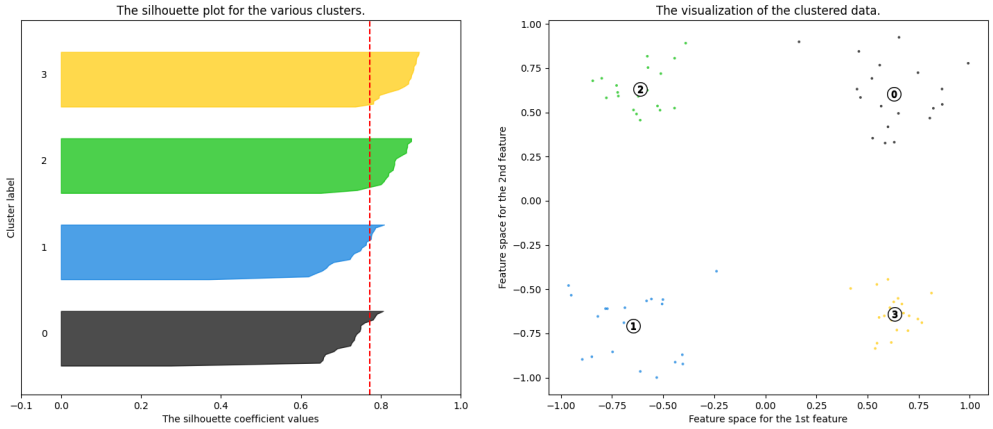
Silhouette analysis, Linkage = complete, Metric = euclidean, K = 2



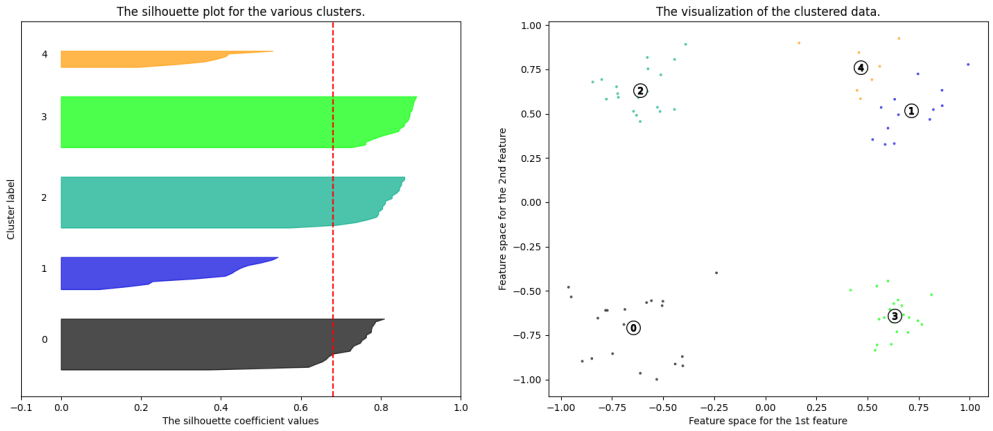
Silhouette analysis, Linkage = complete, Metric = euclidean, K = 3



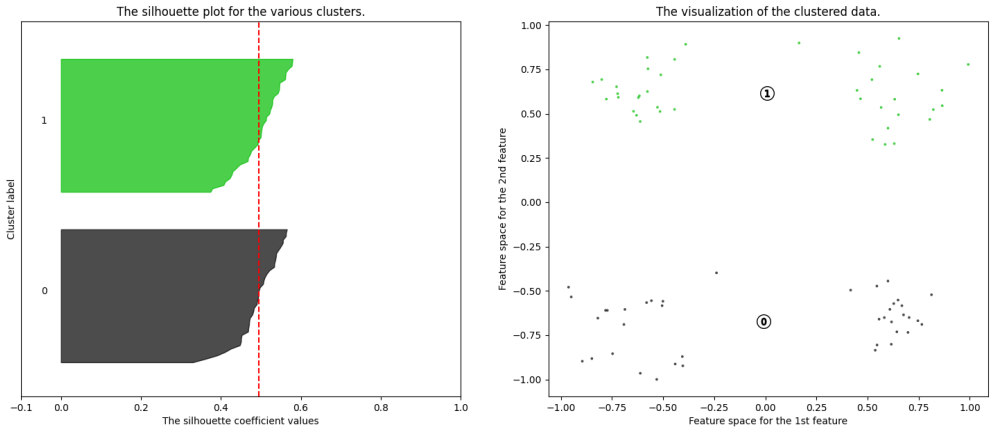
Silhouette analysis, Linkage = complete, Metric = euclidean, K = 4



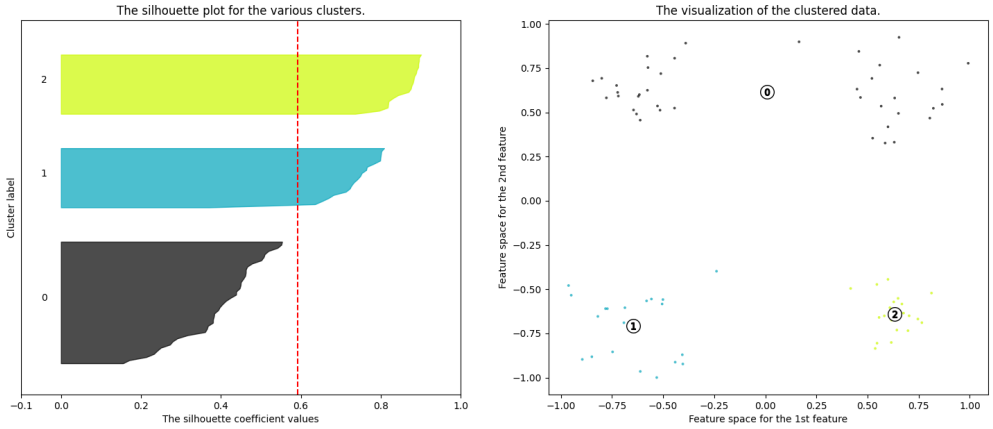
Silhouette analysis, Linkage = complete, Metric = euclidean, K = 5



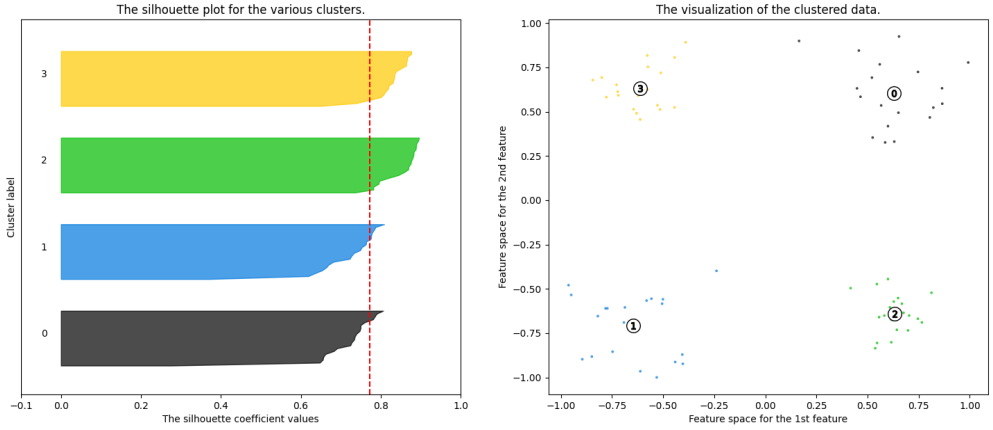
Silhouette analysis, Linkage = single, Metric = cosine, K = 2



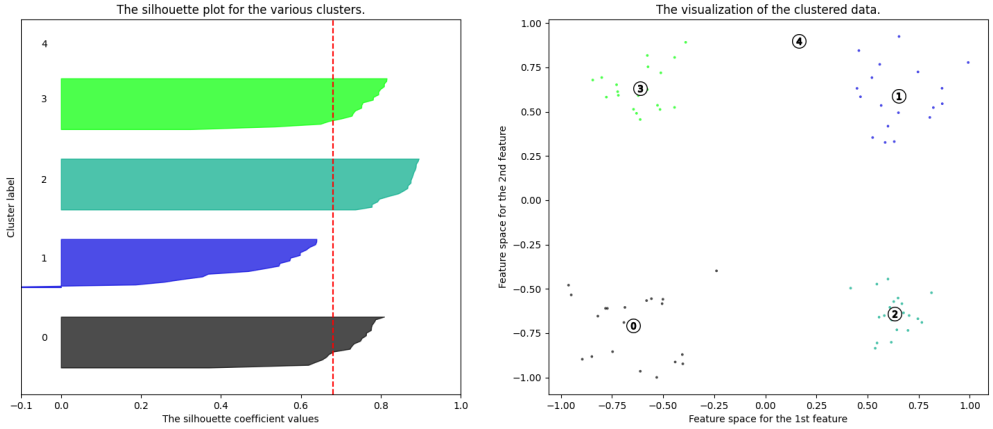
Silhouette analysis, Linkage = single, Metric = cosine, K = 3



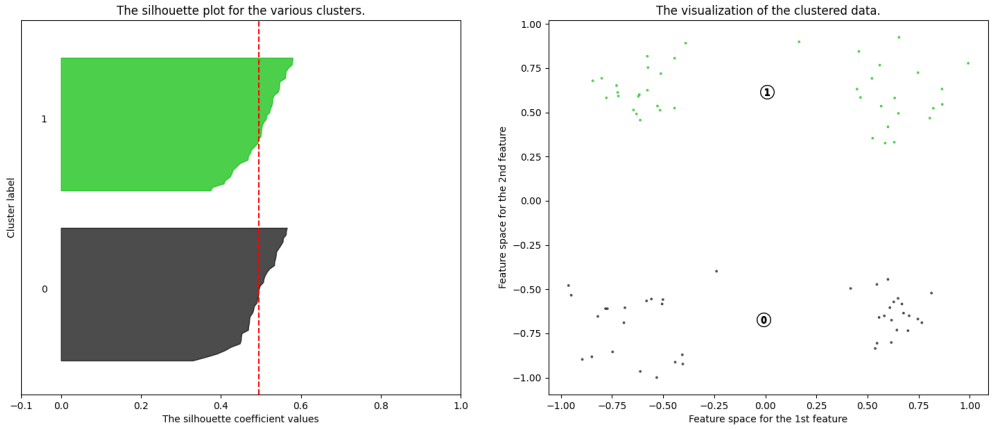
Silhouette analysis, Linkage = single, Metric = cosine, K = 4



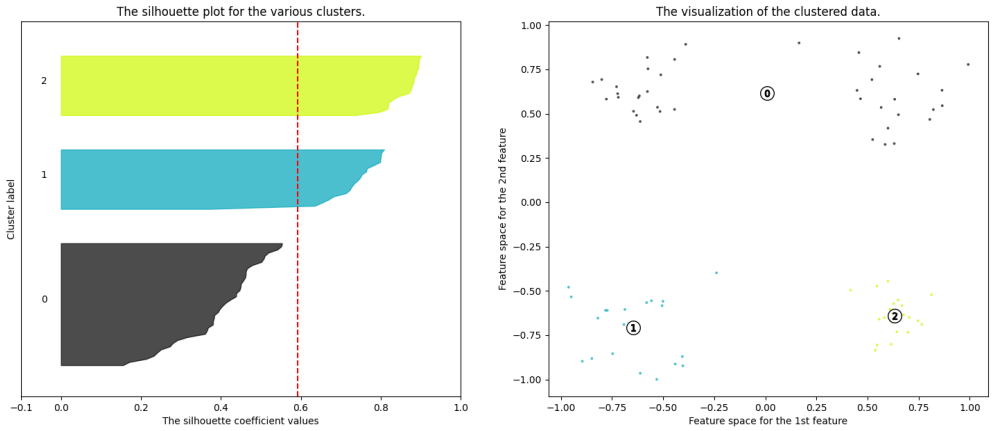
Silhouette analysis, Linkage = single, Metric = cosine, K = 5

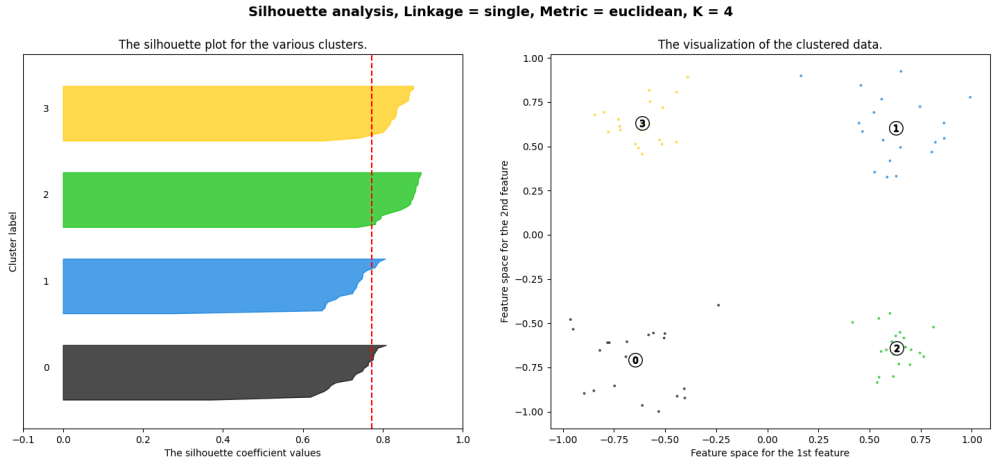


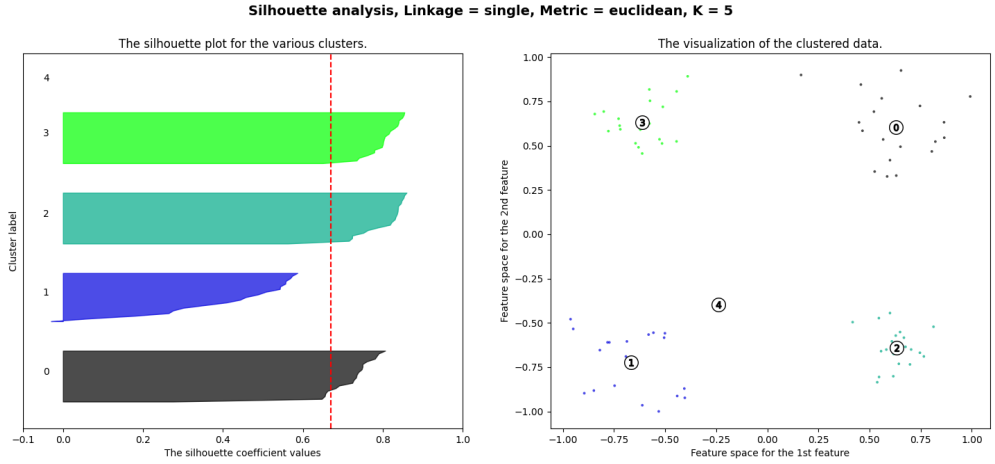
Silhouette analysis, Linkage = single, Metric = euclidean, K = 2



Silhouette analysis, Linkage = single, Metric = euclidean, K = 3







Comments

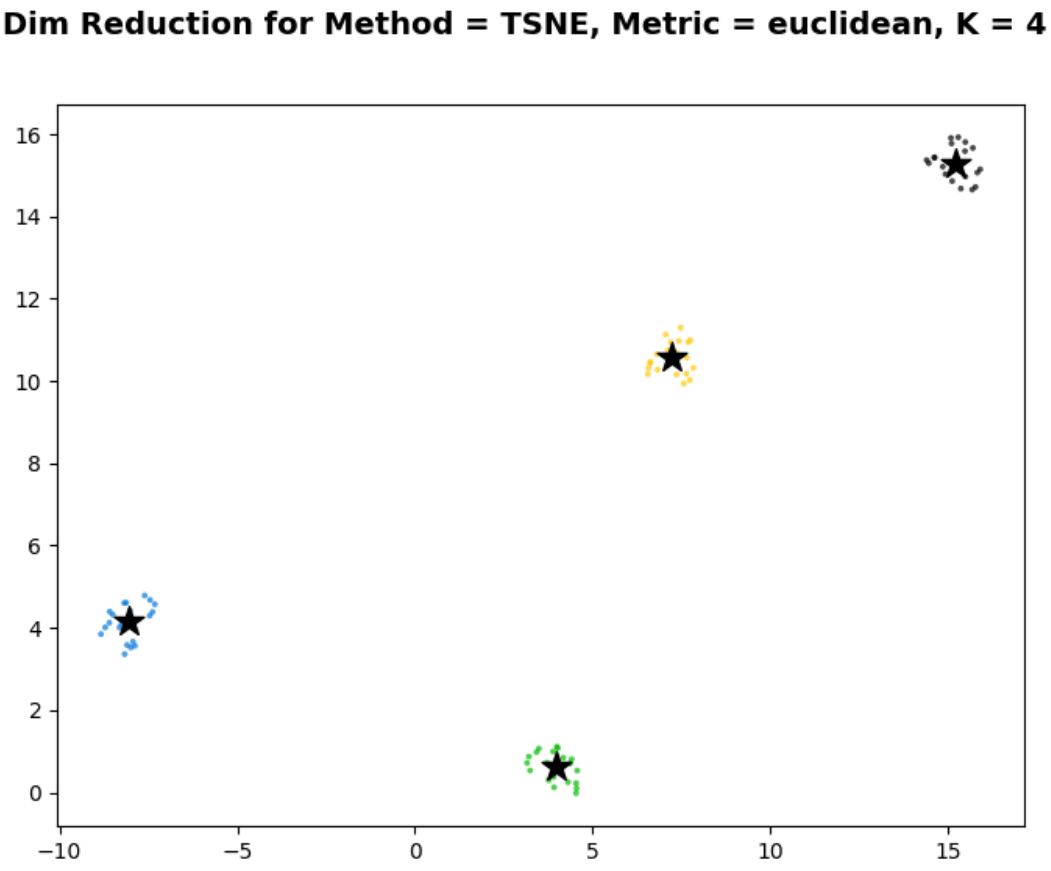
As can be seen from the graphs, the silhouette average score is highest for K = 4. Other metrics and linkages do not contribute to this score much, if at all.

From the cluster visualization to the right of the silhouette graphs, it is obvious that there are 4 clusters.

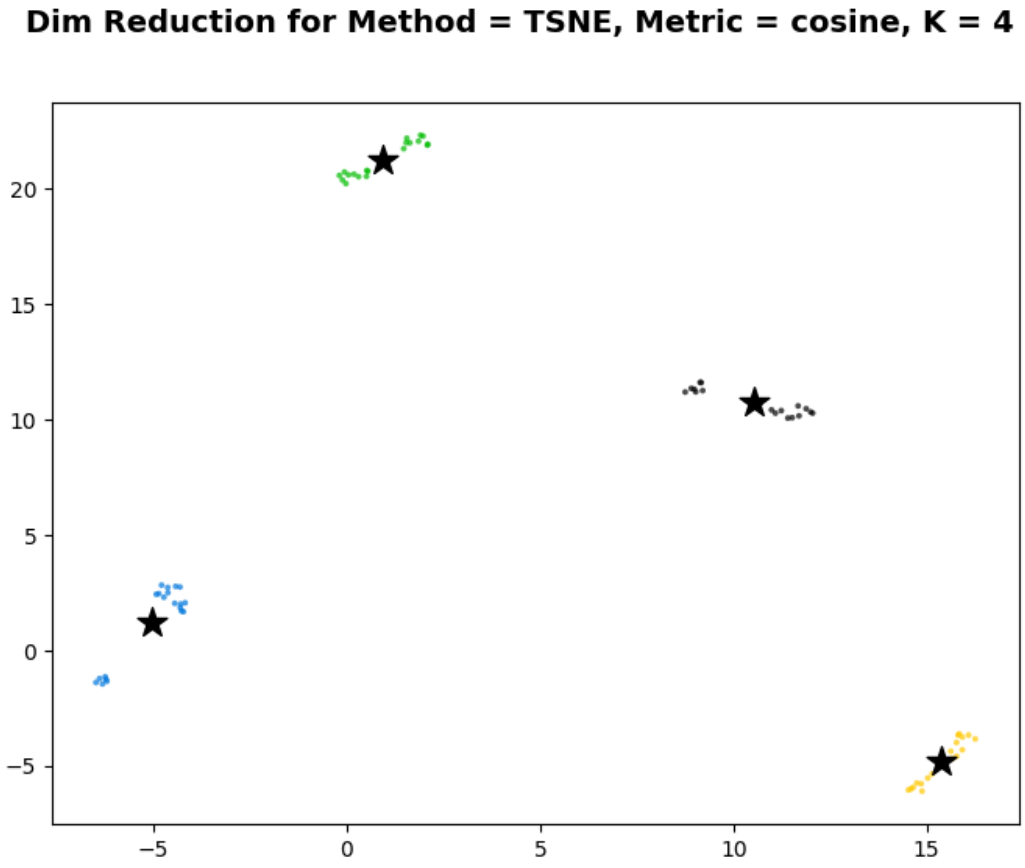
Dimensionality Reduction

t-SNE

Euclidean

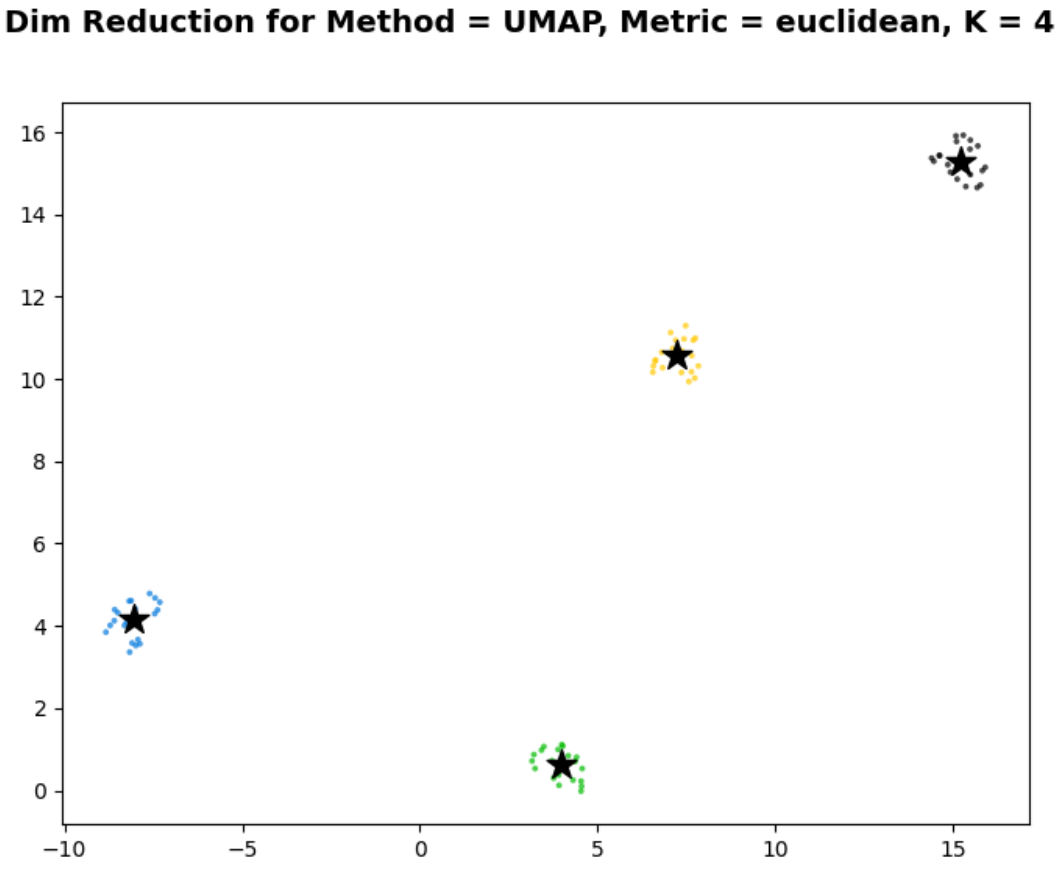


Cosine

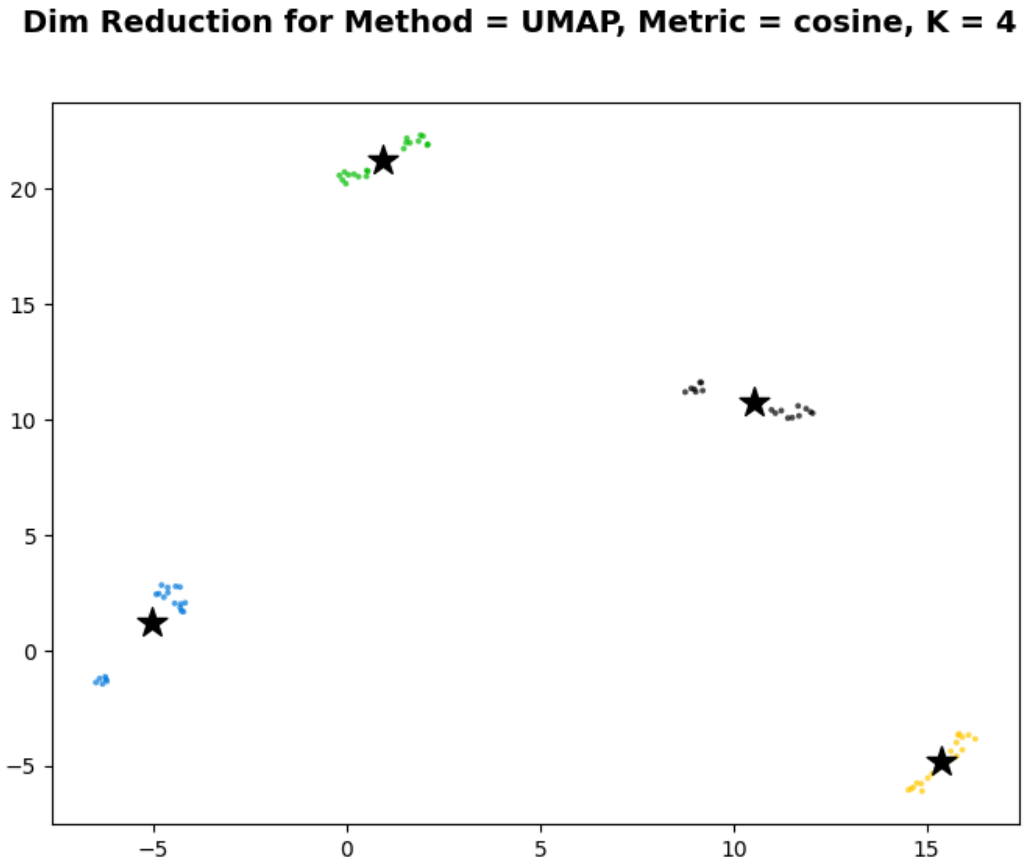

Dim Reduction for Method = TSNE, Metric = cosine, K = 4
20
15
10
5
0
−5
−5
0
5
10
15

UMAP

Euclidean



Cosine



Comments

As it is clear from the dimensionality reduction graphs, the K = 4 value seems to be the correct choice, which was also the case when we plotted their average silhouette scores.

Worst-case Running Time Analysis

HAC

At each iteration, the distance between clusters is computed, which can be at worst O(N^2).
In each iteration, the closest clusters are merged, thus, there can at most be N-1 iterations.
Distance computation between two points in D-dimensional space is O(D).

The final complexity is:

O(N^3 * D)

K-Means or HAC

The worst-case running time complexity for the K-Means algorithm would be:

O(K * N * D) where K is the number of clusters.

Therefore, I would prefer to use K-Means for such large amounts of data with high-dimensional data points since its running time complexity is better.