CEng499-HW2

Selim Tarık ARI

2467611

1 Part 1

In this experiment, we evaluated the performance of a K-Nearest Neighbors (KNN) classifier across different hyperparameter configurations, including varying values of K (the number of neighbors) and different distance metrics: Cosine, Minkowski, and Mahalanobis. The performance of the classifier was assessed using several metrics including Mean Accuracy, Precision, Recall, F1-Score, and the 95% Confidence Interval (CI). The dataset used was preprocessed and split using 10-fold Stratified Cross-Validation, repeated five times.

1.1 Evaluation Metrics

1.1.1 Accuracy

Accuracy is the proportion of correctly predicted instances out of the total instances. It is a basic but commonly used metric for evaluating classification performance. The formula for accuracy is:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

In our experiment, we observed the following trends for accuracy across different configurations:

- Cosine Distance generally provided the highest accuracy, particularly for smaller values of K. As K increased, the accuracy remained stable, suggesting a well-generalized model.
- Minkowski Distance (with $p = 2$ or Euclidean distance) showed moderate performance, but the accuracy decreased as K increased, indicating a potential overfitting issue with small K values.
- Mahalanobis Distance consistently yielded the lowest accuracy, especially at higher values of K. This suggests that Mahalanobis distance may not be suitable for this dataset or task.

1.1.2 Precision

Precision is defined as the ratio of true positive predictions to the total predicted positives. It indicates how many of the predicted positive instances are actually correct:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

A higher precision implies fewer false positives, which is crucial in applications where false positives are costly. Our observations are as follows:

- Cosine Distance exhibited the highest precision across all configurations, meaning it had fewer false positives and made more accurate positive predictions.
- Minkowski Distance showed a moderate level of precision, performing better than Mahalanobis but still less effectively than Cosine.
- Mahalanobis Distance had the lowest precision, suggesting it tended to generate more false positive predictions.

1.1.3 Recall

Recall, also known as sensitivity, measures the proportion of actual positives correctly identified by the classifier:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

It is important when minimizing false negatives is the priority. The recall analysis revealed that:

- Cosine Distance generally had the highest recall, meaning it was better at capturing positive instances.
- Minkowski and Mahalanobis distances had lower recall, indicating that they missed more true positive instances, particularly at higher values of K.

1.1.4 F1 Score

The F1 score is the harmonic mean of precision and recall, providing a balanced evaluation of the classifier's performance. The formula for F1 score is:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

A high F1 score indicates that the classifier has a good balance between precision and recall. We found that:

- Cosine Distance achieved the highest F1 scores, showing that it balanced precision and recall better than the other distance metrics.
- Minkowski and Mahalanobis distances had lower F1 scores, indicating that they either had high precision at the cost of recall or vice versa.

1.1.5 95% Confidence Interval for Accuracy

The 95% Confidence Interval (CI) provides a measure of the variability or uncertainty of the model's performance. A narrow confidence interval suggests that the model's performance is consistent across folds, while a wide interval suggests more variability.

- Cosine Distance had a relatively narrow confidence interval, indicating stable performance across different cross-validation folds.
- Minkowski and Mahalanobis distances exhibited wider confidence intervals, especially with higher values of K, suggesting greater variability in their performance.

1.2 Results

The following table shows the average metrics across different values of K and distance metrics.

K	Metric	Mean Acc.	Mean Prec.	Mean Recall	Mean F1 Score	95% Conf. Interval
1	Cosine	0.9067	0.9128	0.9067	0.9061	±0.0523
1	Minkowski	0.8867	0.8992	0.8867	0.8842	±0.0553
1	Mahalanobis	0.8667	0.8749	0.8667	0.8638	±0.0675
3	Cosine	0.9333	0.9448	0.9333	0.9318	±0.0436
3	Minkowski	0.9133	0.9206	0.9133	0.9134	±0.0553
3	Mahalanobis	0.9067	0.9212	0.9067	0.9038	±0.0558
5	Cosine	0.9333	0.9426	0.9333	0.9320	±0.0477
5	Minkowski	0.8867	0.9068	0.8867	0.8838	±0.0479
5	Mahalanobis	0.8733	0.8930	0.8733	0.8722	±0.0687
7	Cosine	0.9467	0.9565	0.9467	0.9454	±0.0380
7	Minkowski	0.9000	0.9160	0.9000	0.8989	±0.0402
7	Mahalanobis	0.8533	0.8698	0.8533	0.8525	±0.0640
9	Cosine	0.9333	0.9474	0.9333	0.9305	±0.0515
9	Minkowski	0.8867	0.9006	0.8867	0.8840	±0.0479
9	Mahalanobis	0.8467	0.8712	0.8467	0.8416	±0.0586

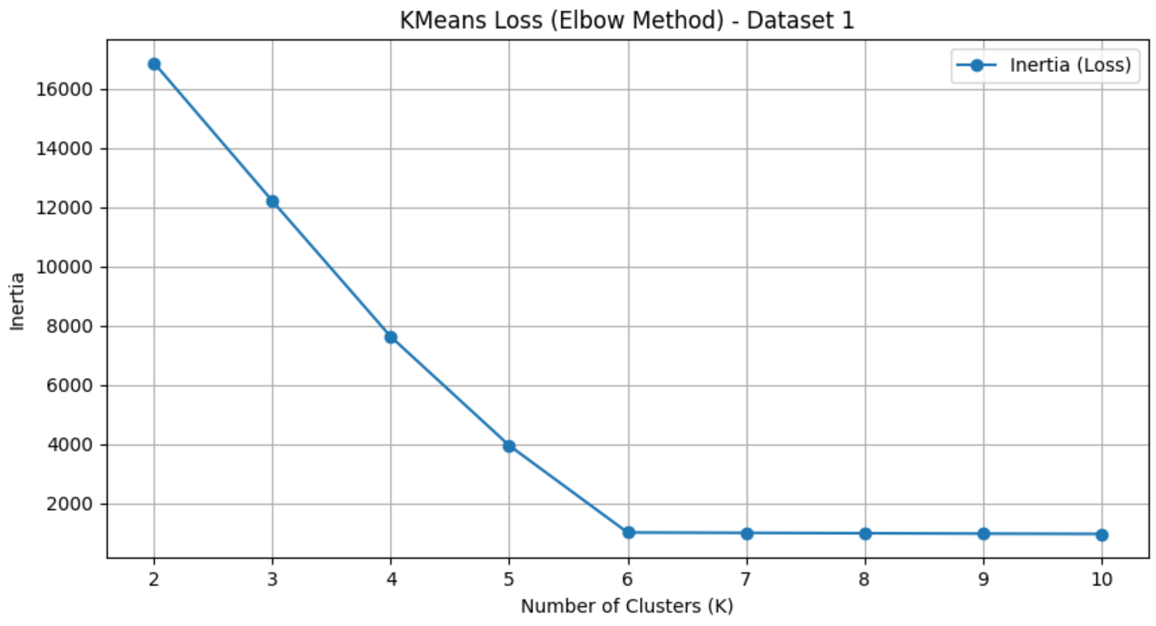
1.3 Conclusion

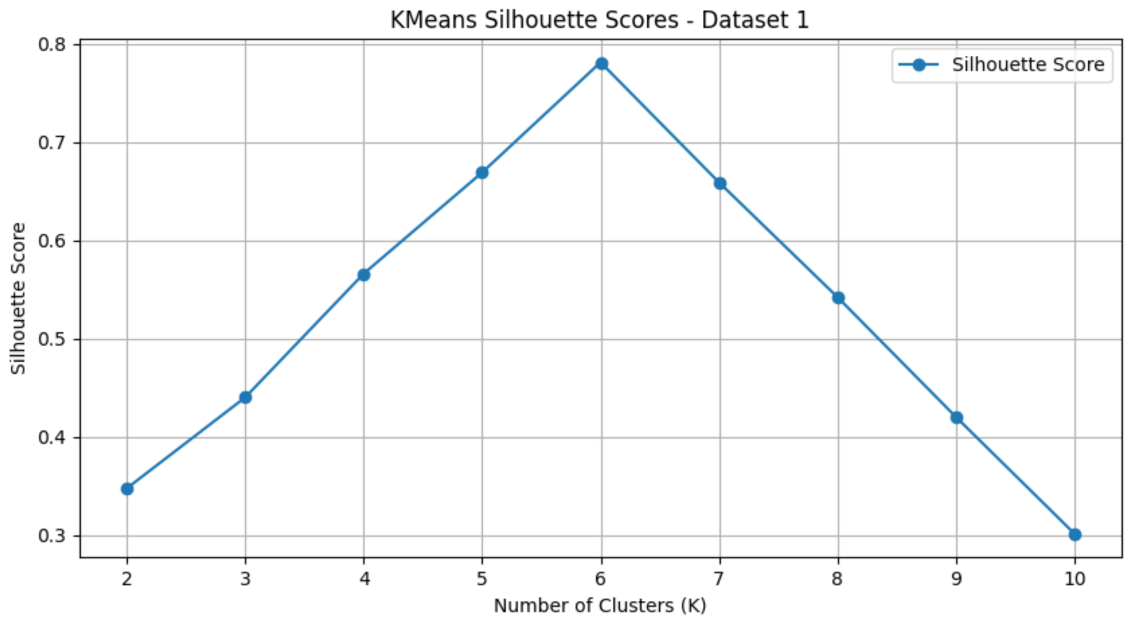
Based on the results, the Cosine Distance metric consistently outperforms the Minkowski and Mahalanobis metrics across all values of K, yielding the highest accuracy, precision, recall, and F1 scores. The performance of Mahalanobis Distance was generally inferior, particularly for higher values of K.

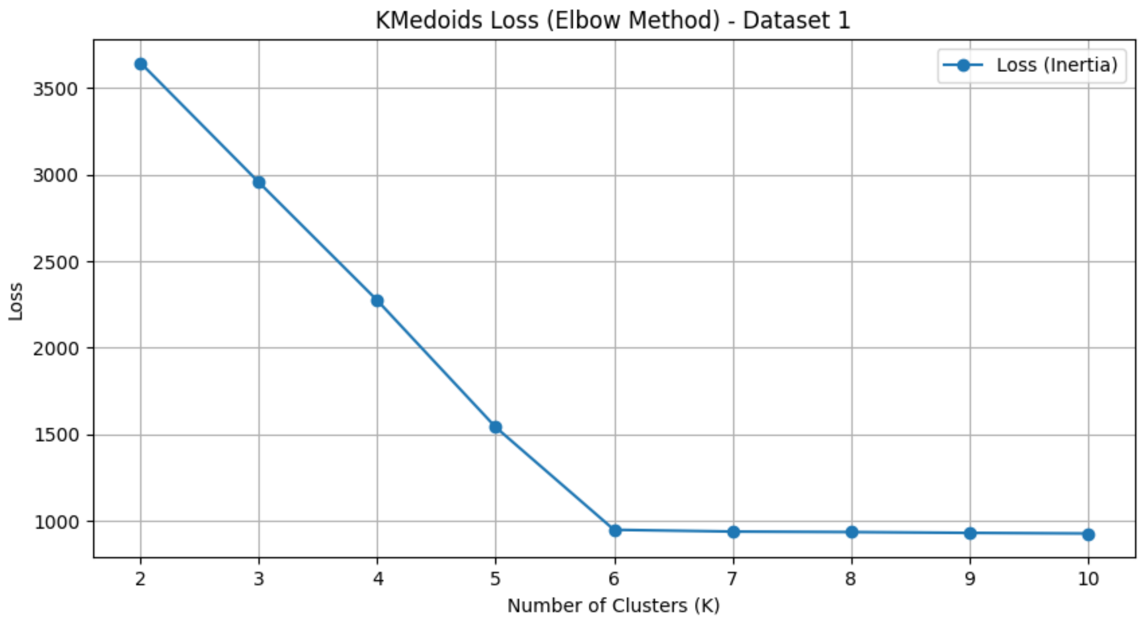
For practical applications, Cosine Distance with a larger K (such as $K = 7$) is recommended as it provides the best trade-off between bias and variance, yielding stable and high performance. Minkowski distance performs adequately but shows more variability, particularly with larger K values, and Mahalanobis distance should likely be avoided for this task due to its relatively poor performance.

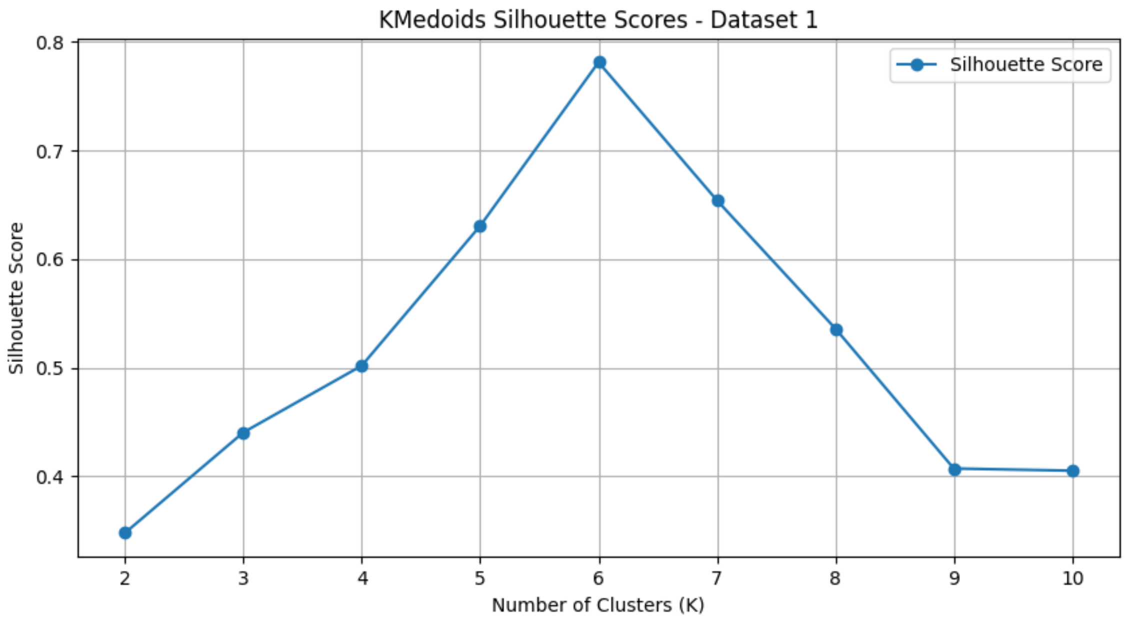
2 Part 2

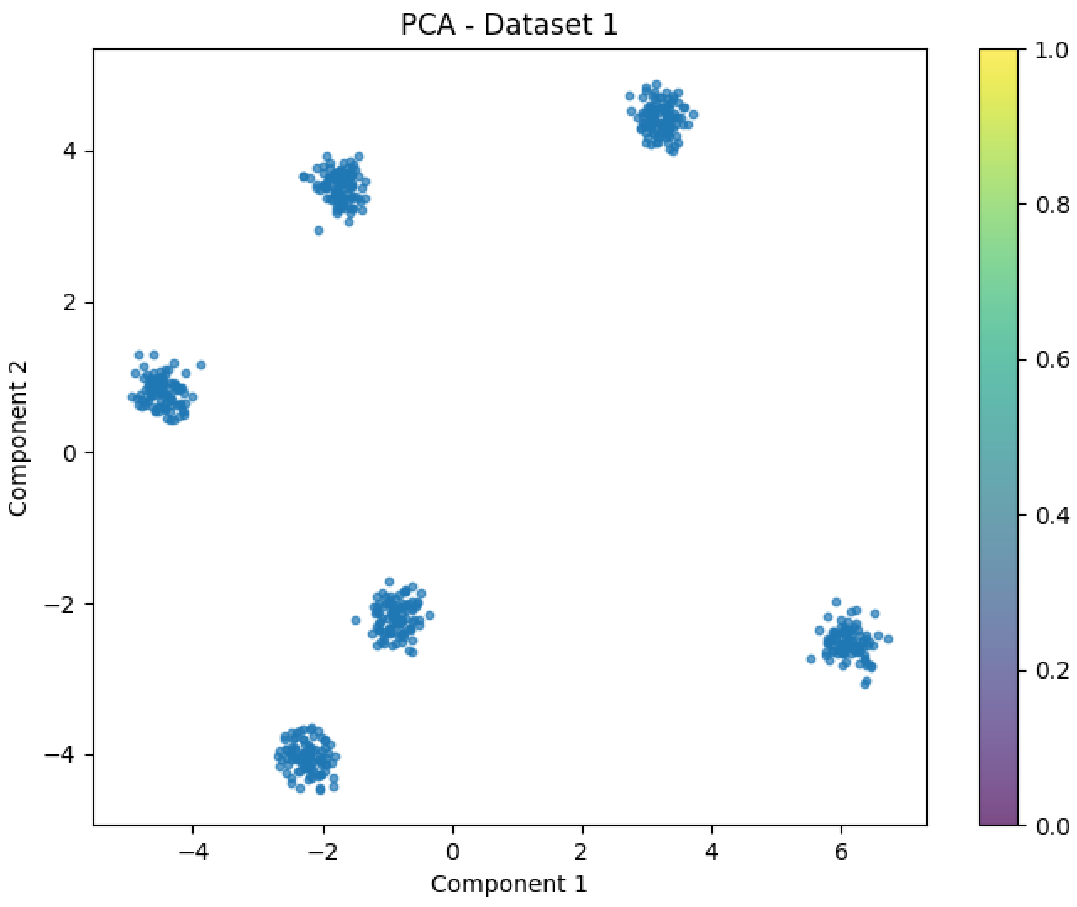
Dataset 1

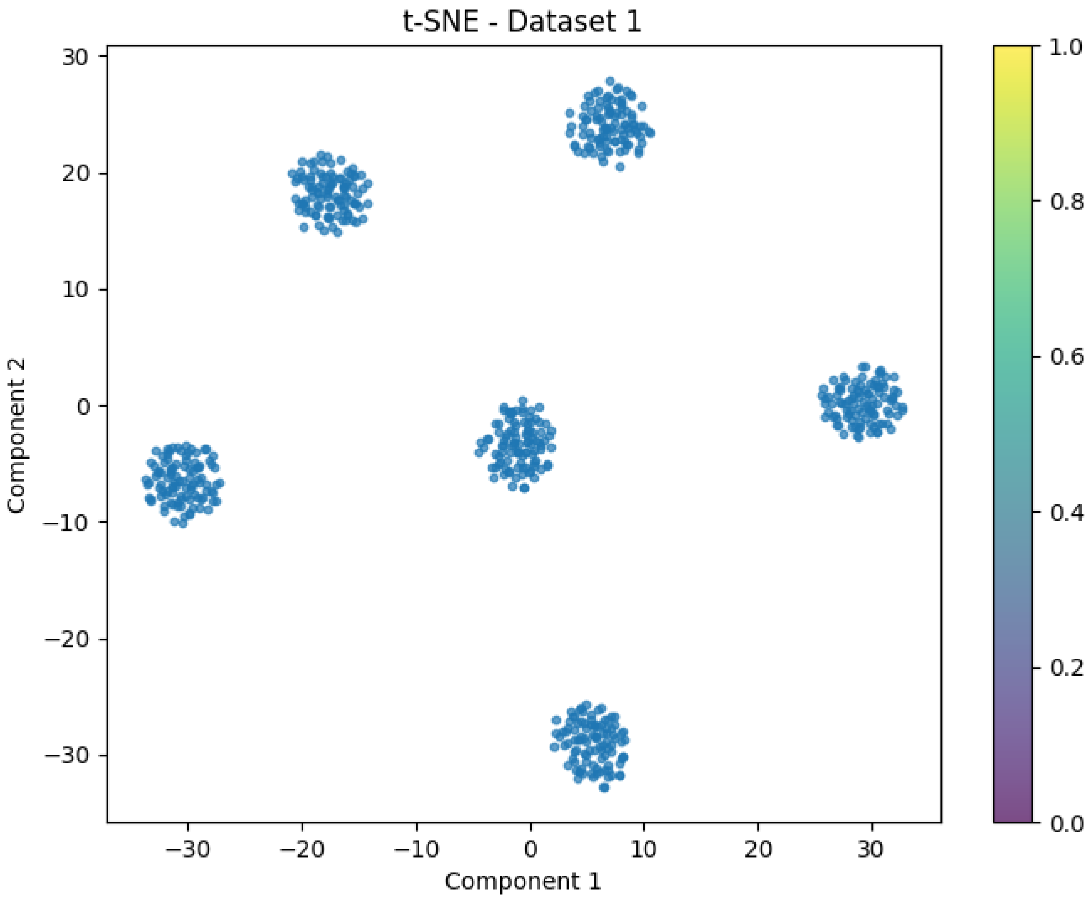


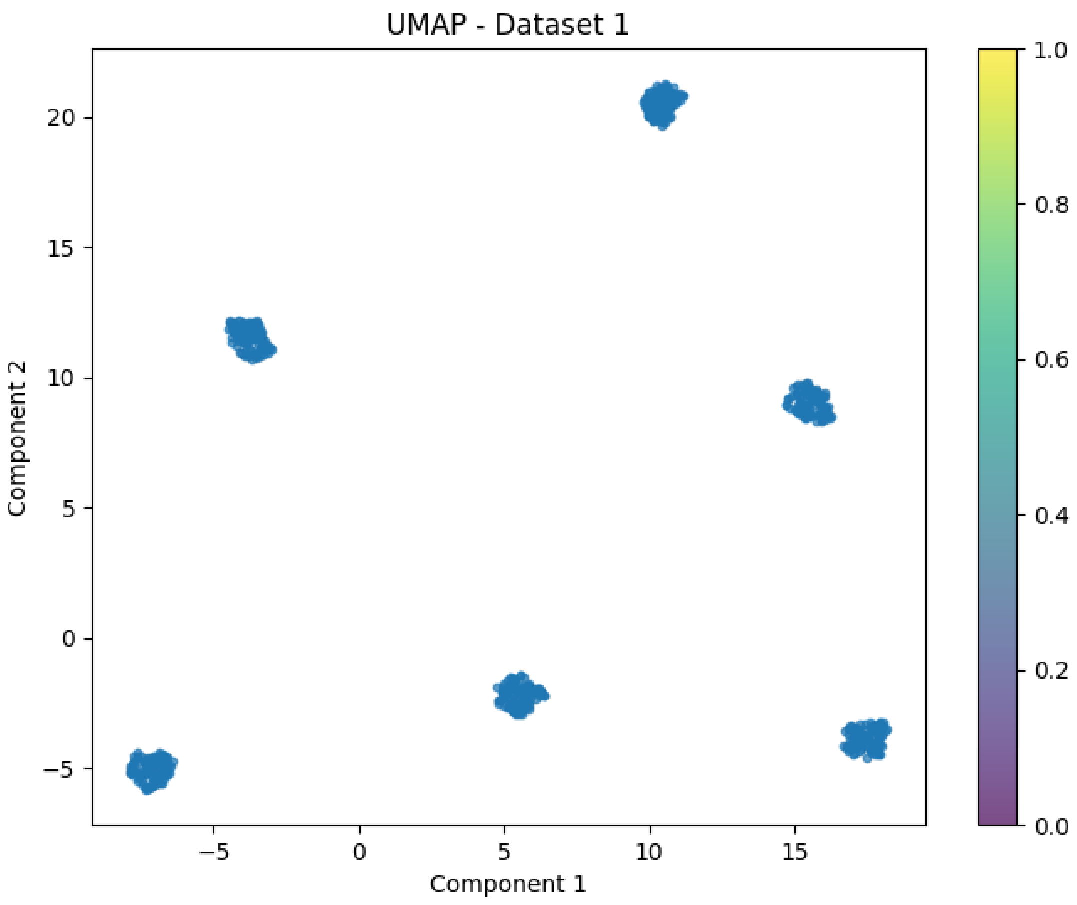


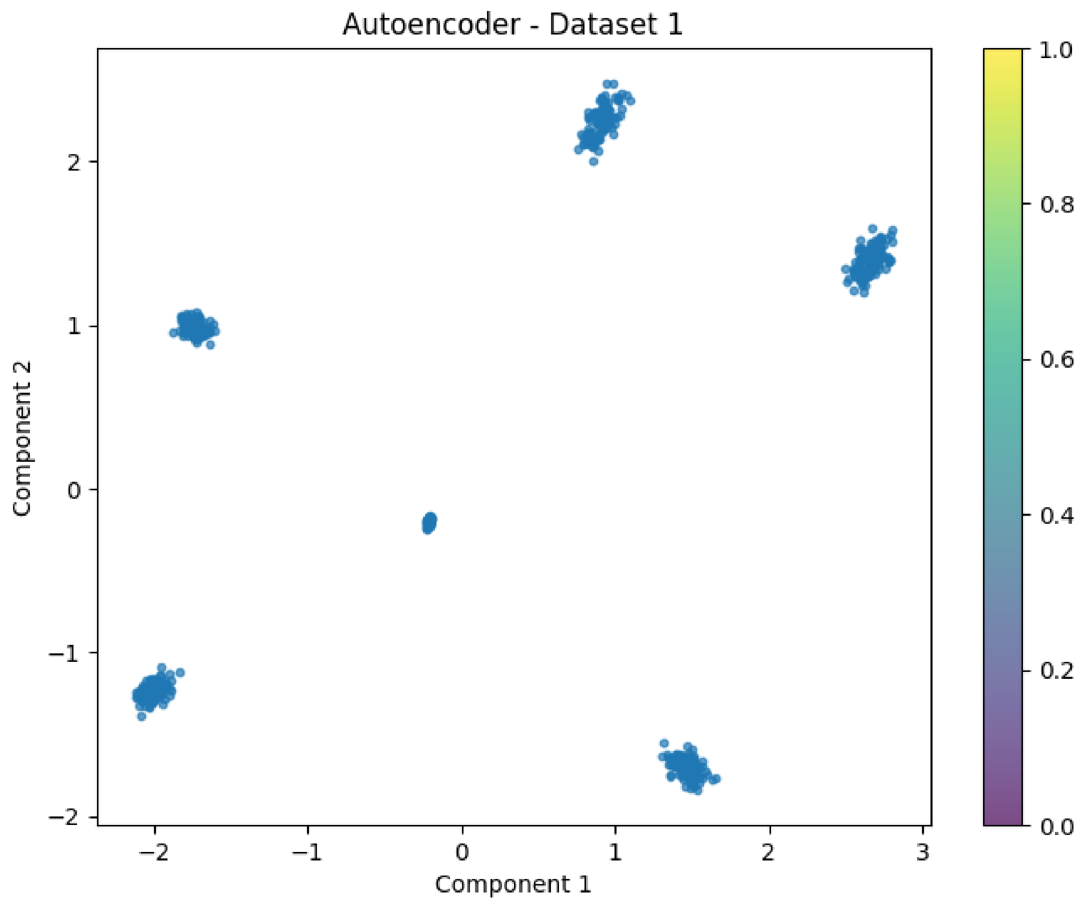

KMedoids Loss (Elbow Method) - Dataset 1
Loss (Inertia)
Loss
3500
3000
2500
2000
1500
1000
2
3
4
5
6
7
8
9
10
Number of Clusters (K)


KMedoids Silhouette Scores - Dataset 1
Silhouette Score
Silhouette Score
0.8
0.7
0.6
0.5
0.4
2
3
4
5
6
7
8
9
10
Number of Clusters (K)


PCA - Dataset 1
Component 1
Component 2
4
2
0
−2
−4
−4
−2
0
2
4
6
1.0
0.8
0.6
0.4
0.2
0.0


t-SNE - Dataset 1
Component 1
Component 2
30
20
10
0
−10
−20
−30
1.0
0.8
0.6
0.4
0.2
0.0


UMAP - Dataset 1
Component 1
Component 2
−5
0
5
10
15
20
0.0
0.2
0.4
0.6
0.8
1.0



For Dataset 1, we evaluated the clustering results of **KMeans** and **KMedoids**, followed by dimensionality reduction using **PCA**, **t-SNE**, **UMAP**, and **Autoencoder**. The findings are summarized as follows:

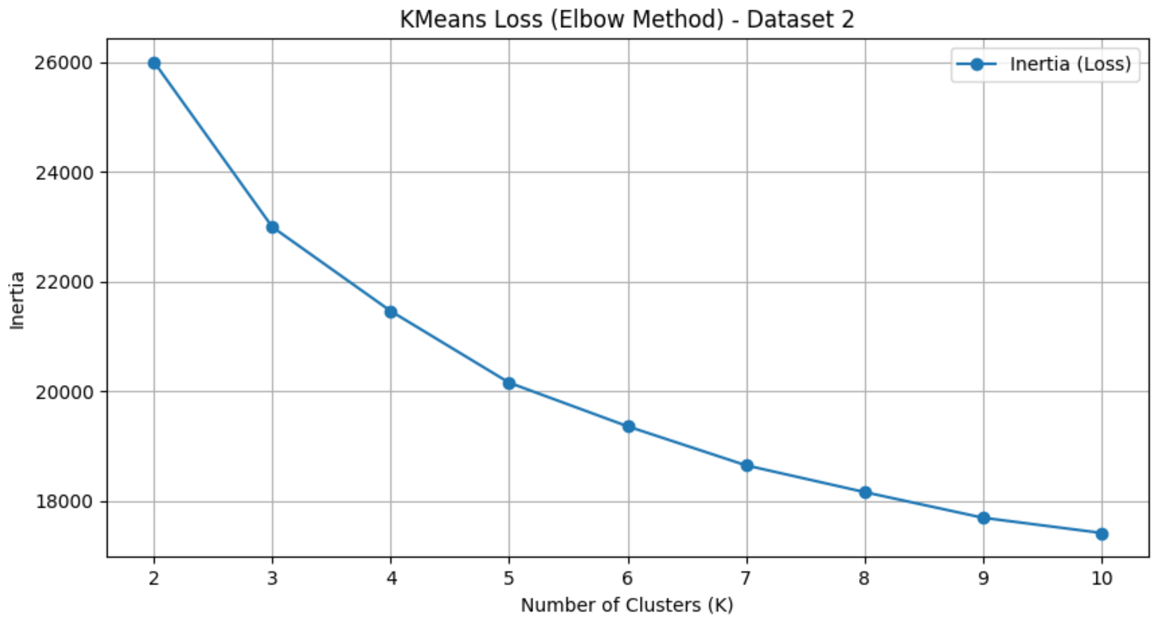
Clustering Results

- **KMeans**: The clustering patterns were distinct and well-separated, indicating effective partitioning of the data. Clusters showed a high degree of compactness and low overlap.
- **KMedoids**: The clustering results were comparable to KMeans, though with slightly less compact clusters. This is expected, as KMedoids minimizes dissimilarity using medoids, making it robust to outliers but slightly less optimal in compactness.

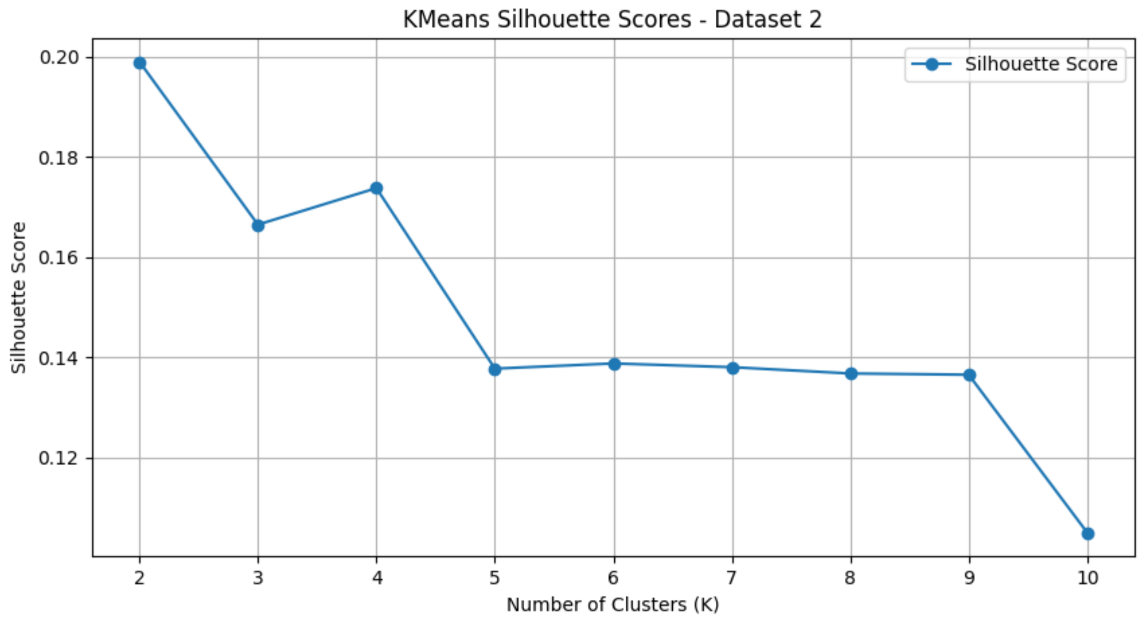
Dimensionality Reduction Results

- **PCA**: The clusters appeared linear and somewhat overlapping, as PCA relies on variance maximization and may not capture nonlinear structures in the data.
- **t-SNE**: Provided well-separated clusters with clear boundaries, highlighting its ability to handle nonlinear structures effectively. However, the method is computationally expensive.
- **UMAP**: Similar to t-SNE, UMAP provided distinct and well-separated clusters but achieved this with a lower computational cost. The clusters appeared slightly more cohesive compared to t-SNE.
- **Autoencoder**: The clusters were distinct but less separated compared to t-SNE and UMAP. This may indicate that the autoencoder architecture could benefit from further optimization.

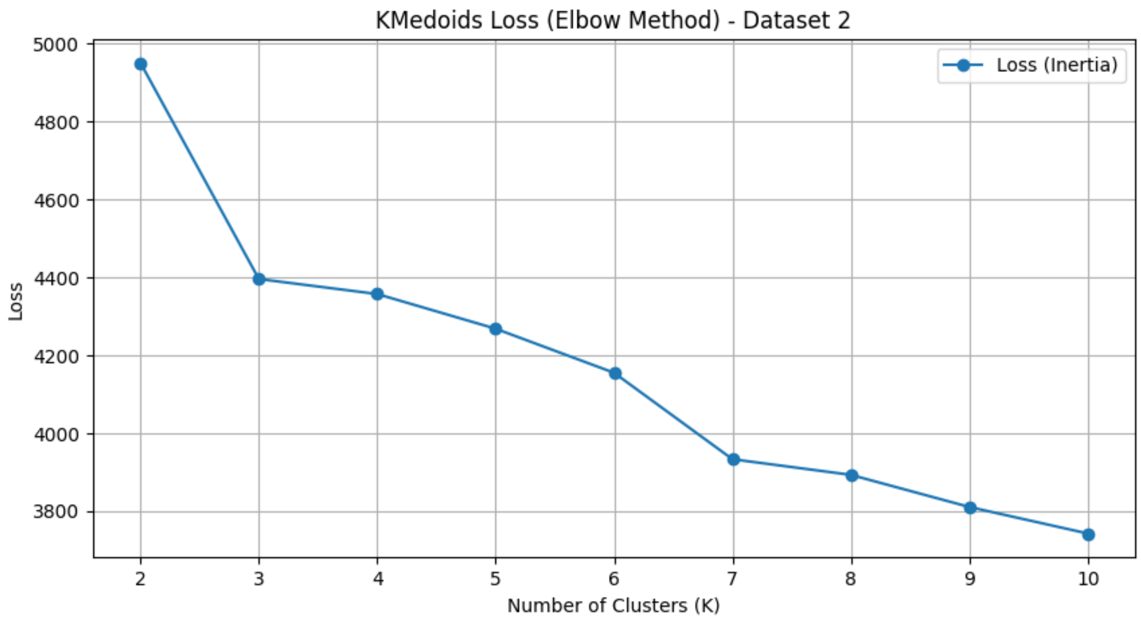
Dataset 2

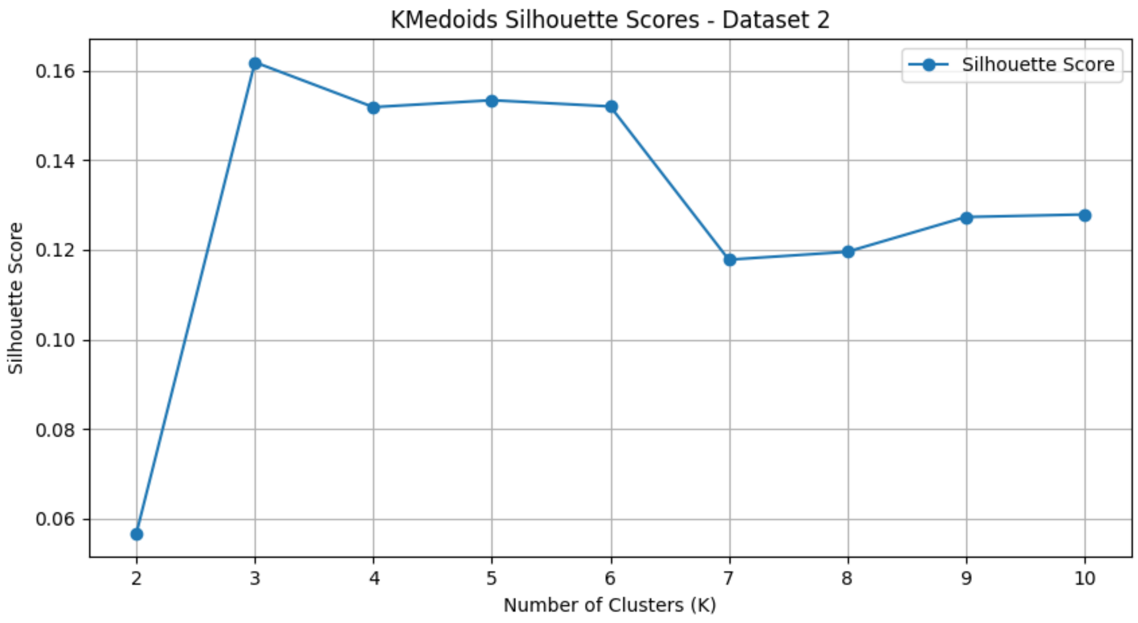


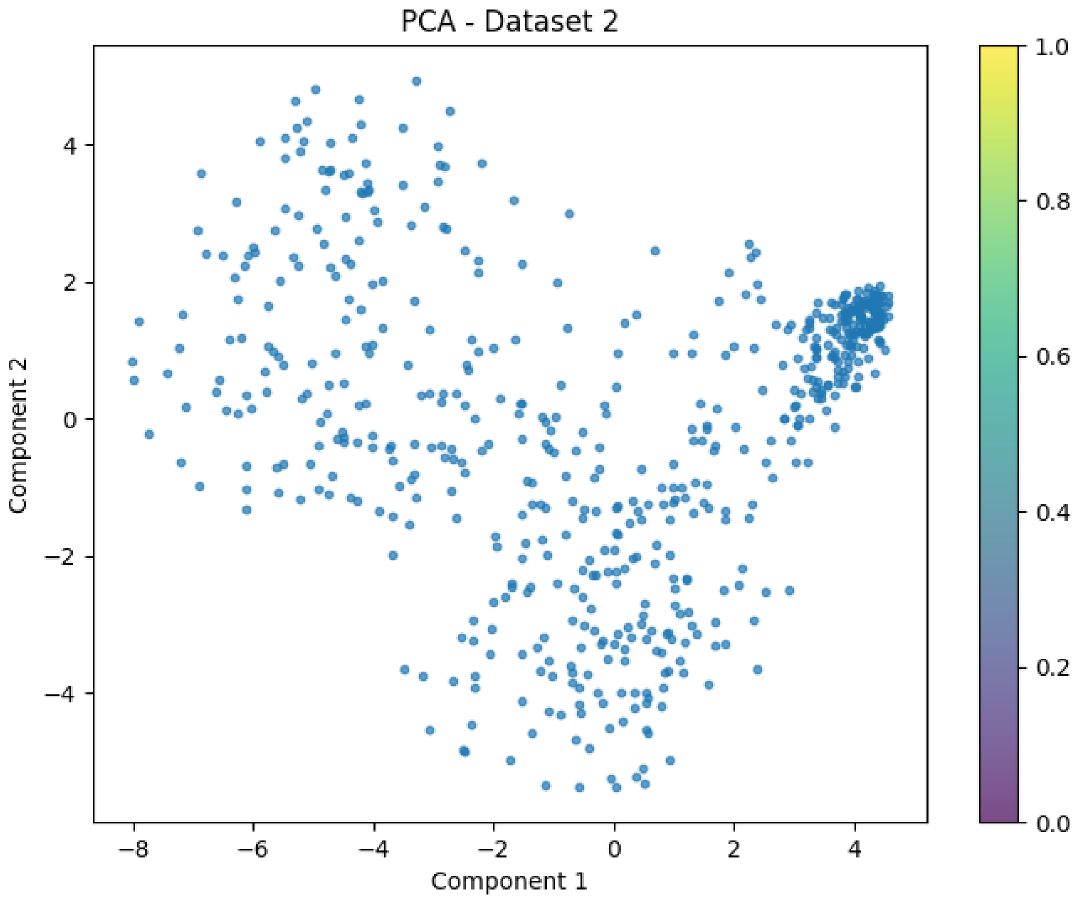
KMeans Loss (Elbow Method) - Dataset 2
Inertia (Loss)
Inertia
26000
24000
22000
20000
18000
2
3
4
5
6
7
8
9
10
Number of Clusters (K)

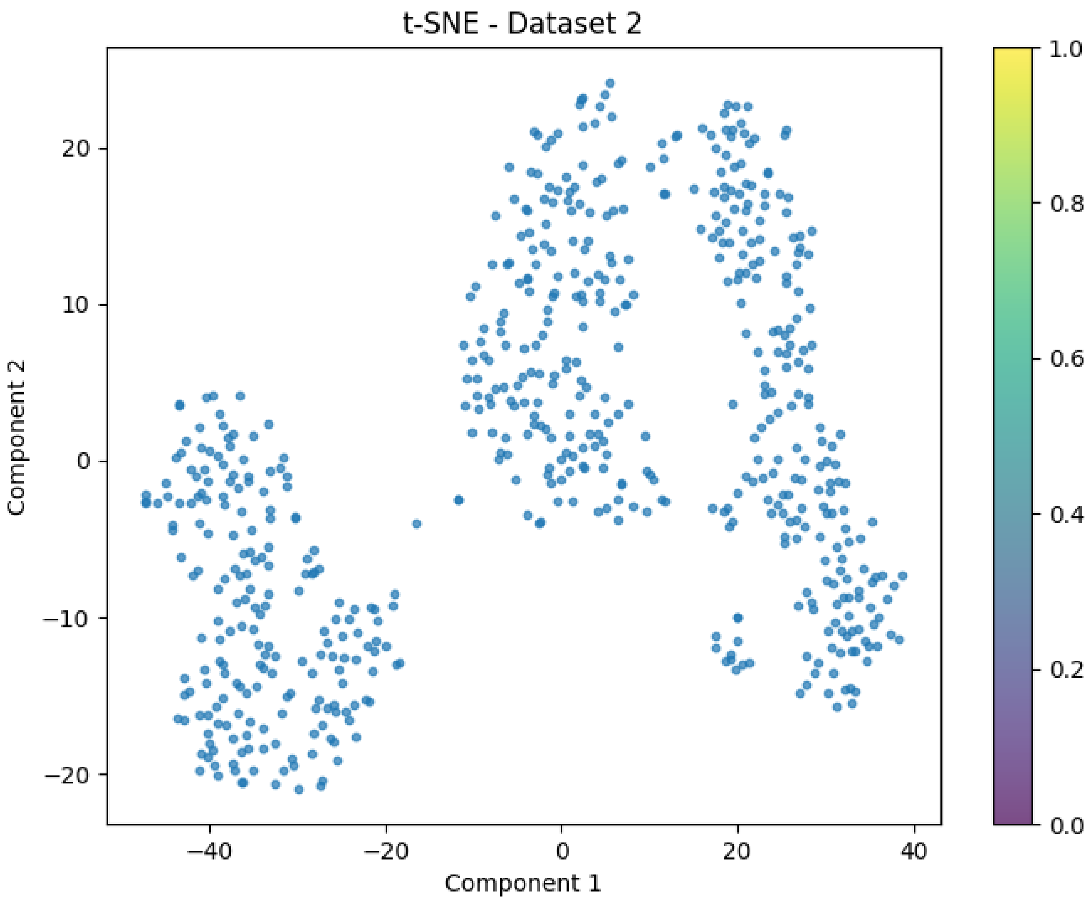


KMeans Silhouette Scores - Dataset 2
Silhouette Score
Silhouette Score
0.20
0.18
0.16
0.14
0.12
2
3
4
5
6
7
8
9
10
Number of Clusters (K)

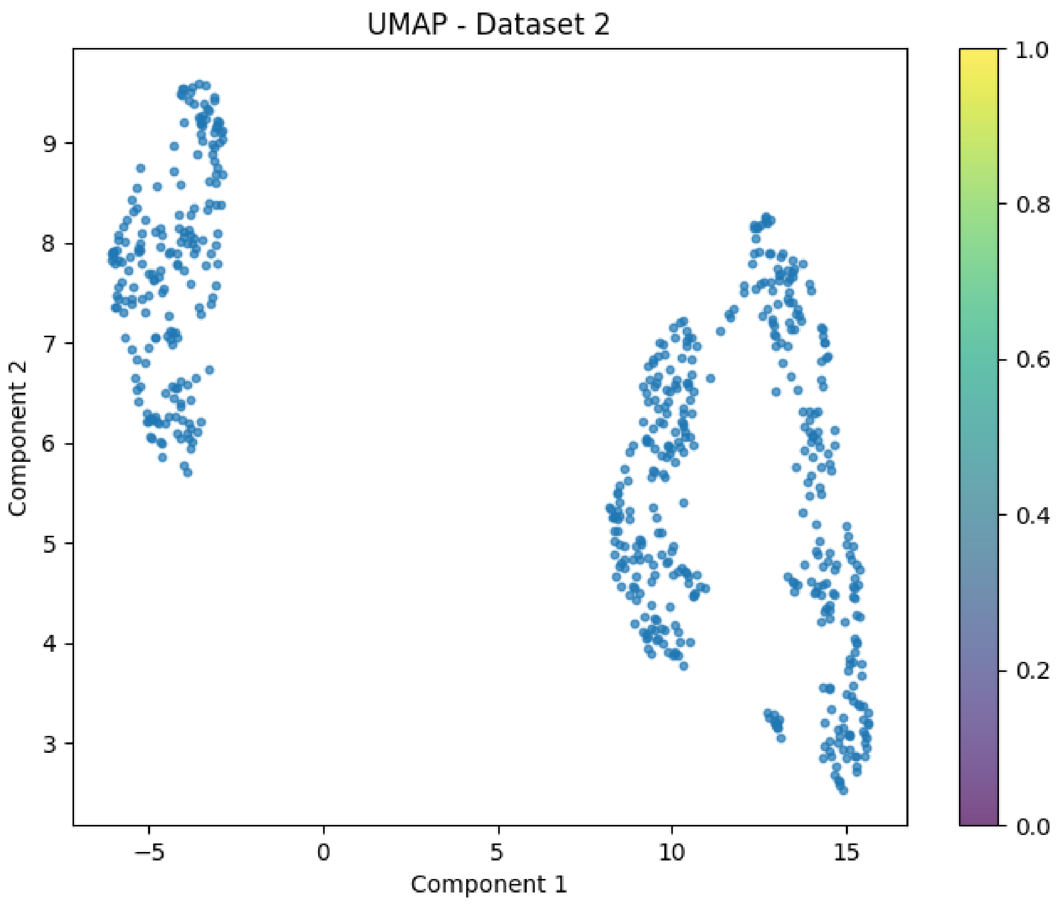

KMedoids Loss (Elbow Method) - Dataset 2
Loss (Inertia)
Loss
5000
4800
4600
4400
4200
4000
3800
2
3
4
5
6
7
8
9
10
Number of Clusters (K)

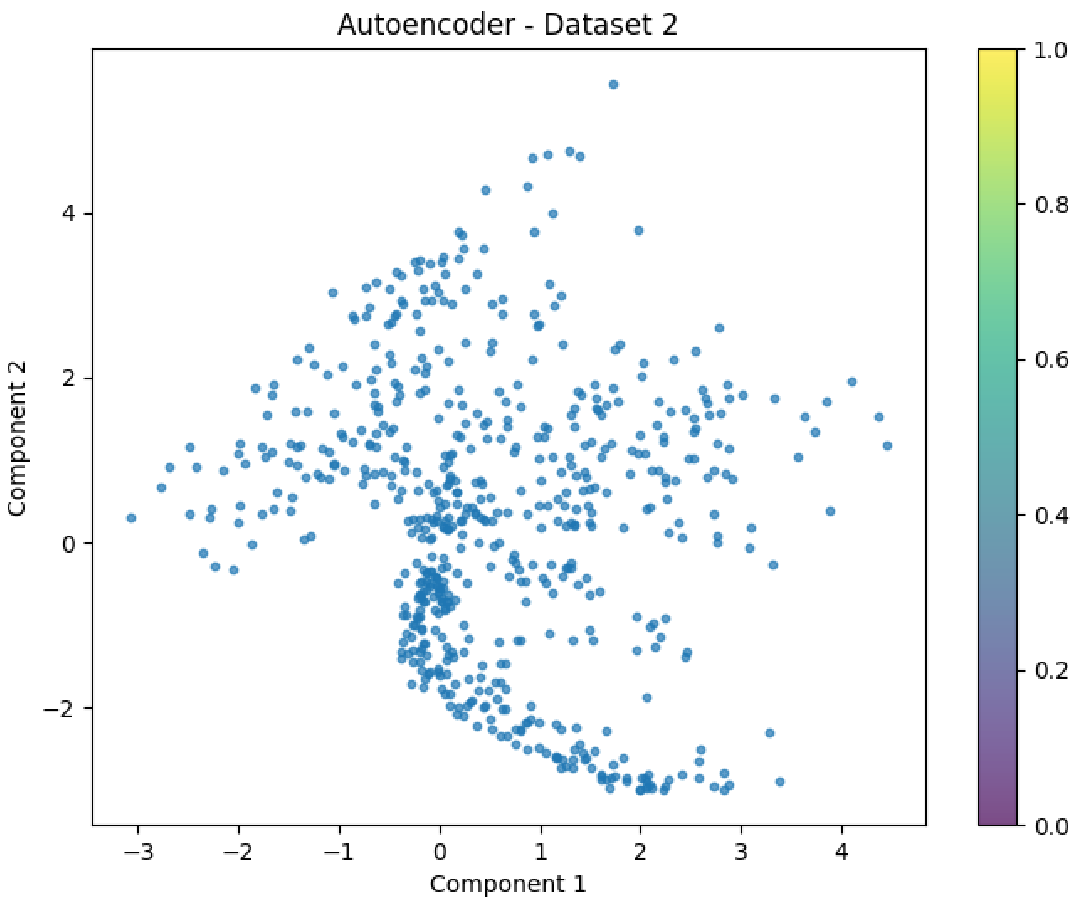

KMedoids Silhouette Scores - Dataset 2
Silhouette Score
Silhouette Score
0.16
0.14
0.12
0.10
0.08
0.06
2
3
4
5
6
7
8
9
10
Number of Clusters (K)


PCA - Dataset 2
Component 1
Component 2
−8
−6
−4
−2
0
2
4
−4
−2
0
2
4
0.0
0.2
0.4
0.6
0.8
1.0



t-SNE - Dataset 2
Component 1
Component 2
−40
−20
0
20
40
20
10
0
−10
−20
1.0
0.8
0.6
0.4
0.2
0.0


UMAP - Dataset 2
Component 1
Component 2
−5
0
5
10
15
3
4
5
6
7
8
9
0.0
0.2
0.4
0.6
0.8
1.0



For Dataset 2, we observed different clustering and dimensionality reduction behaviors due to the dataset's inherent characteristics. The findings are summarized as follows:

Clustering Results

- **KMeans**: Clustering was less distinct, with overlapping regions indicating that the data distribution might not align well with the assumptions of the KMeans algorithm.
- **KMedoids**: Performed slightly better than KMeans, showing robustness to overlapping clusters and outliers.

Dimensionality Reduction Results

- **PCA**: As with Dataset 1, PCA produced overlapping clusters, highlighting its limitation in handling nonlinear data distributions.
- **t-SNE**: Delivered well-separated clusters, effectively capturing the complex structure of the data. This supports t-SNE's strength in visualizing high-dimensional data in 2D.
- **UMAP**: Similar to t-SNE, UMAP produced clear cluster separations but with slightly less distinct boundaries. Its computational efficiency makes it a practical alternative to t-SNE.
- **Autoencoder**: Clusters were moderately distinct, but some overlap was observed. Further tuning of the autoencoder could potentially enhance the results.

Comparison and Insights

- **KMeans vs. KMedoids**: While both methods performed comparably, KMedoids demonstrated slightly better robustness to overlapping clusters, especially in Dataset 2.
- **Dimensionality Reduction**:

- t-SNE and UMAP consistently outperformed PCA and Autoencoder in producing distinct and meaningful cluster visualizations.
- UMAP provided similar results to t-SNE but with reduced computational cost, making it a more efficient choice for large datasets.
- Autoencoder results suggest the need for further architectural or training optimizations to achieve comparable performance to t-SNE and UMAP.