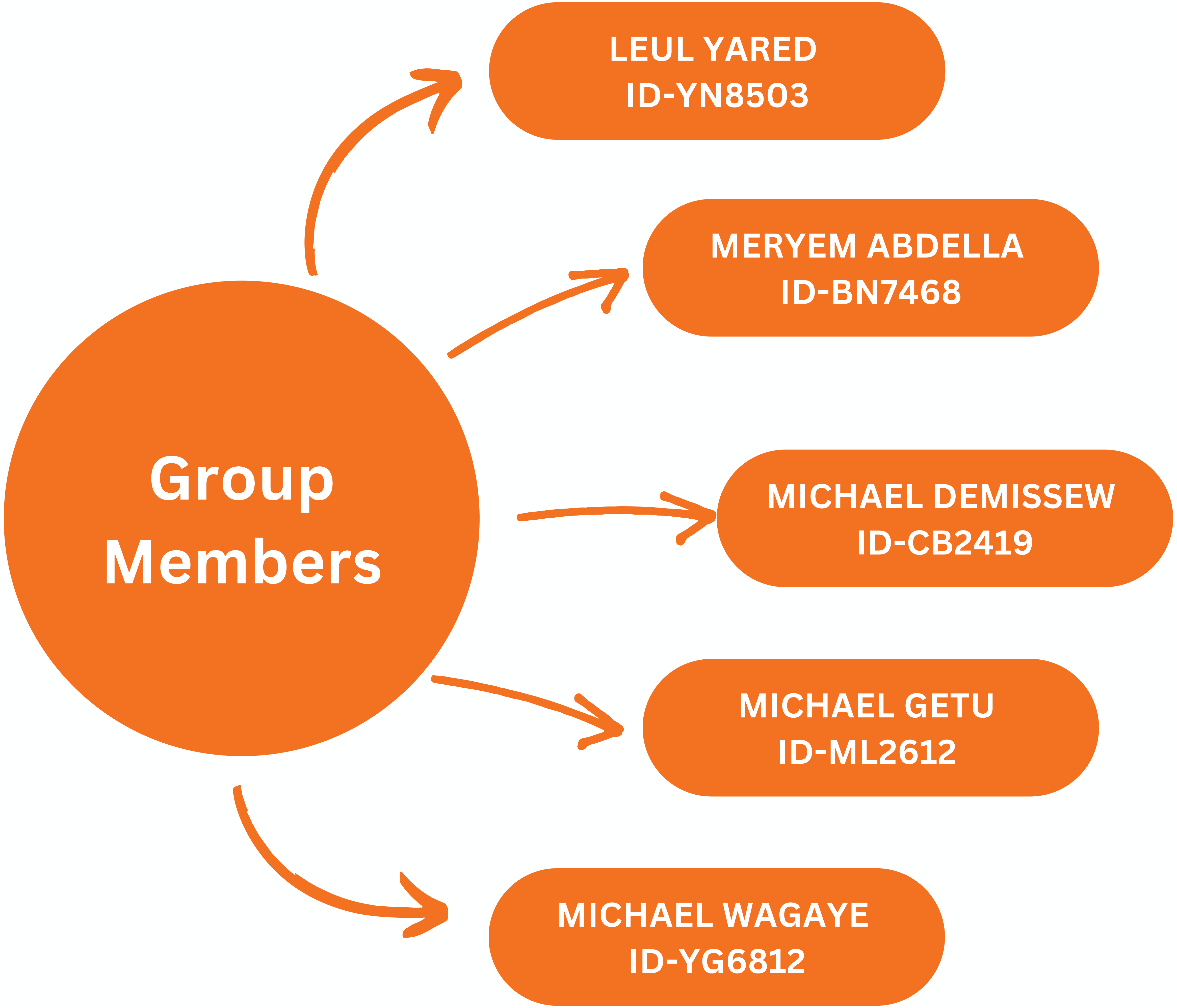
HILCOE

Machine Learning Problems

Artificial Intelligence (CS488) Presentation



Group Members

LEUL YARED
ID-YN8503

MERYEM ABDELLA
ID-BN7468

MICHAEL DEMISSEW
ID-CB2419

MICHAEL GETU
ID-ML2612

MICHAEL WAGAYE
ID-YG6812

Overview

1. Introduction
2. ***Problem 1:*** Decision Tree Depth Optimization
3. ***Problem 2:*** K-Nearest Neighbors Hyperparameter Tuning
4. ***Problem 3:*** Perceptron Learning Algorithm
5. ***Problem 4:*** Comparing Decision Tree, KNN, and Perceptron
6. ***Problem 5:*** K-Means Clustering and Visualization
7. ***Problem 6:*** K-Medoids Clustering and Comparison with K-Means

Introduction

Algorithm: A well-defined procedure to solve a specific problem.

- ***Learning Algorithm:*** Improves performance over time through experience with data, enabling predictions or decisions on new inputs.

  Categories: Supervised, Unsupervised, Semi-Supervised, Reinforcement Learning.

  Supervised Learning: Trained on labeled data.

  Goal: Map inputs to desired outputs.

  Unsupervised Learning: Trained on unlabeled data.

  Goal: Identify patterns or structures (e.g., clustering).

Key Terms

- ***Overfitting:*** Model captures noise rather than patterns, leading to poor generalization.
- ***Underfitting:*** Model is too simple, failing to capture patterns, resulting in poor performance.
- ***Decision Tree:*** A supervised learning algorithm for classification and regression.

  Structure: Tree-like model where each node represents a decision based on an attribute.
- ***K-Nearest Neighbors (KNN):*** A simple, non-parametric algorithm for classification and regression.

  Key Idea: Assigns class based on the majority among k nearest neighbors.

Introduction

Distance Metrics

- Euclidean Distance: Straight-line distance.
- Manhattan Distance: Distance measured along axes at right angles.

Perceptron Learning Algorithm

- The simplest neural network, a linear binary classifier, updating weights based on misclassification.

Clustering

- ***K-Means:*** Partitions data into k clusters based on nearest centroids.
- ***K-Medoids:*** Similar to K-Means but uses actual data points (medoids) as centers, making it more resistant to noise.

Performance Metrics

- ***Accuracy:*** Correct predictions / Total predictions.
- ***Precision***: Correct positive predictions / Total positive predictions.
- ***Recall***: Correct positive predictions / Actual positives.
- ***F1-Score***: Harmonic mean of Precision and Recall.

Evaluation Tools

- ***Confusion Matrix:*** Summarizes model performance across true/false positives and negatives.
- ***Silhouette Score:*** Measures clustering quality, ranging from -1 to 1.

Problem #1

Decision Tree Depth Optimization

Task: Train a decision tree classifier on the Iris dataset and optimize its performance by adjusting the maximum depth of the tree.

Instructions:

1. Split the dataset into training and testing sets.
2. Train a decision tree with various maximum depths.
3. Evaluate the performance of each model using accuracy, precision, recall, and F1-score.
4. Visualize the decision boundaries of the best-performing model.
5. Plot the performance metrics against the tree depth to analyze the impact.

Objective: Understand how tree depth affects overfitting and underfitting in decision trees.

Problem #1

Dataset Splitting

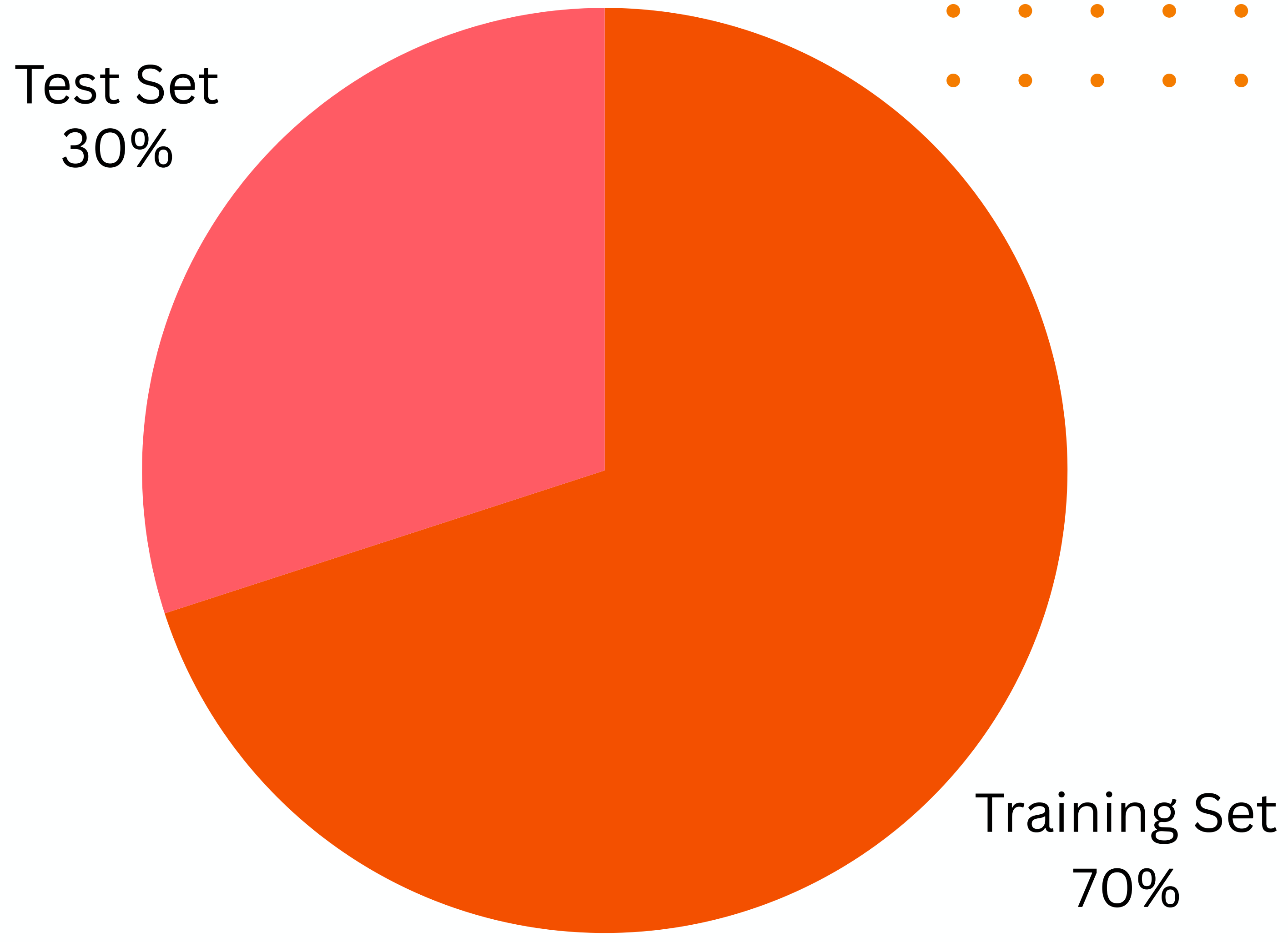
- Split dataset into training (70%) and testing (30%) using train_test_split.
- Ensures evaluation on unseen data.

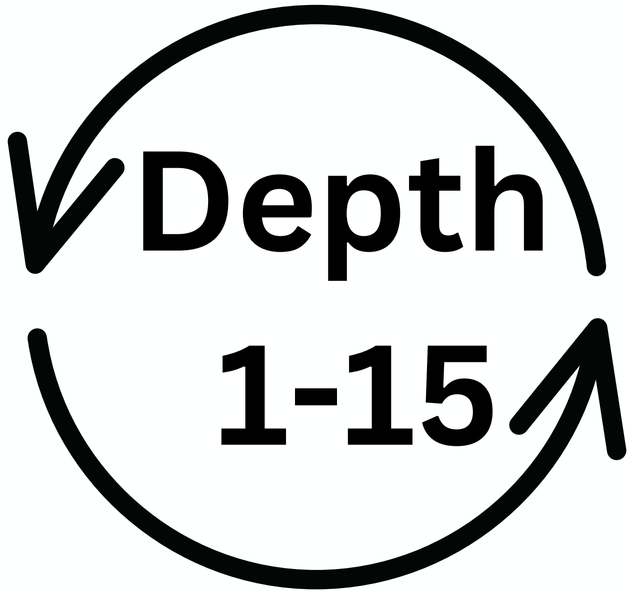
Training with Various Depths

- Loop iterates from depth 1 to 15.
- Trains decision tree classifiers at each depth.

Performance Evaluation

- Metrics: Accuracy, Precision, Recall, F1-Score.



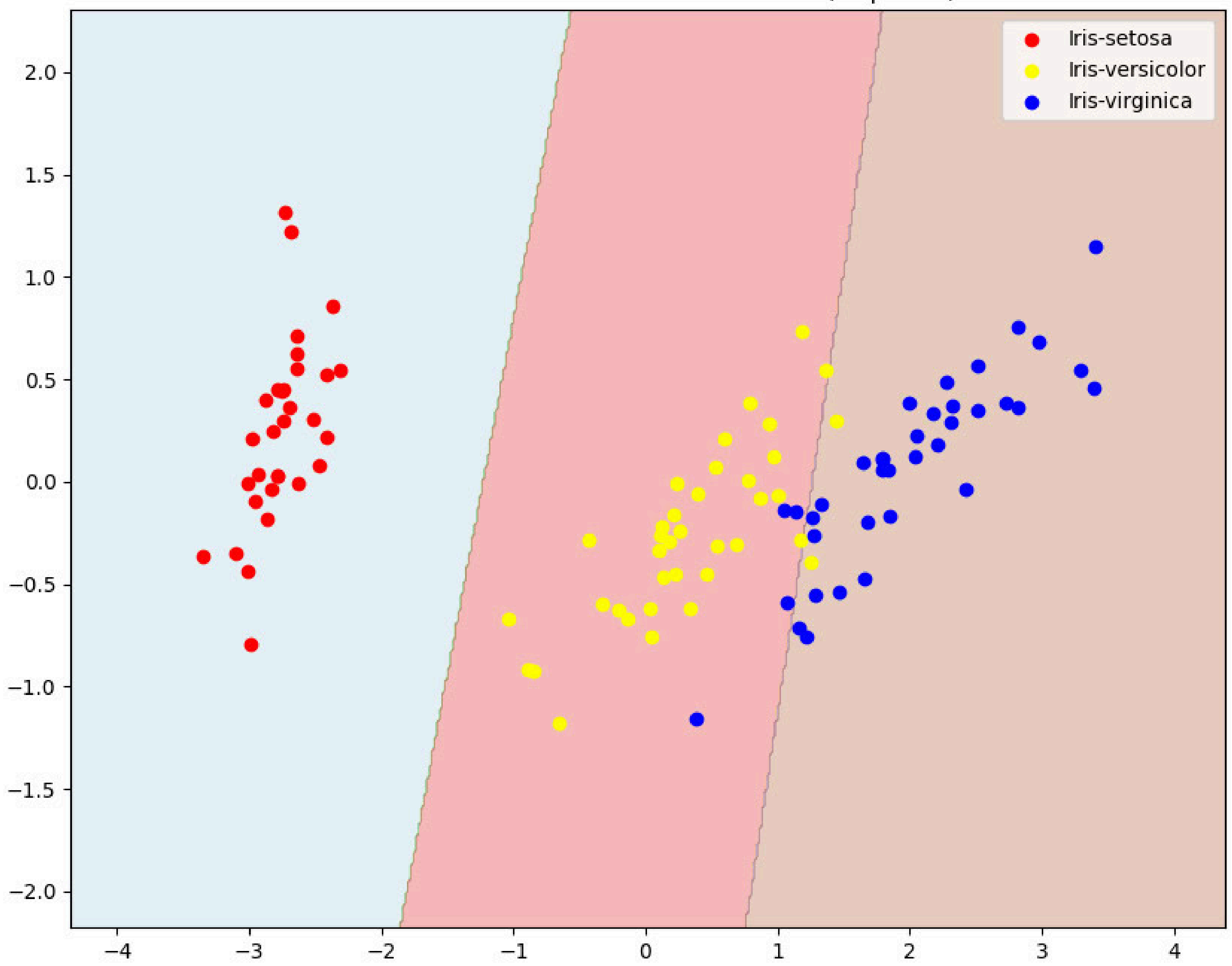


Problem #1

Visualization of decision boundary by the use of PCA

This graph depicts the decision boundaries of a decision tree classifier with a maximum depth of 3, trained using the Iris dataset with dimensions reduced via ***PCA.***

Decision Boundaries of the Best Model (Depth=3)



PCA (Principal Component Analysis) *reduces the number of features in a dataset while preserving the most important information.*

It transforms the original features into a smaller set of new, uncorrelated features called principal components. This simplifies data, and reduces noise.

Model Simplicity

The decision boundaries are relatively simple and linear, reflecting a low-complexity model.

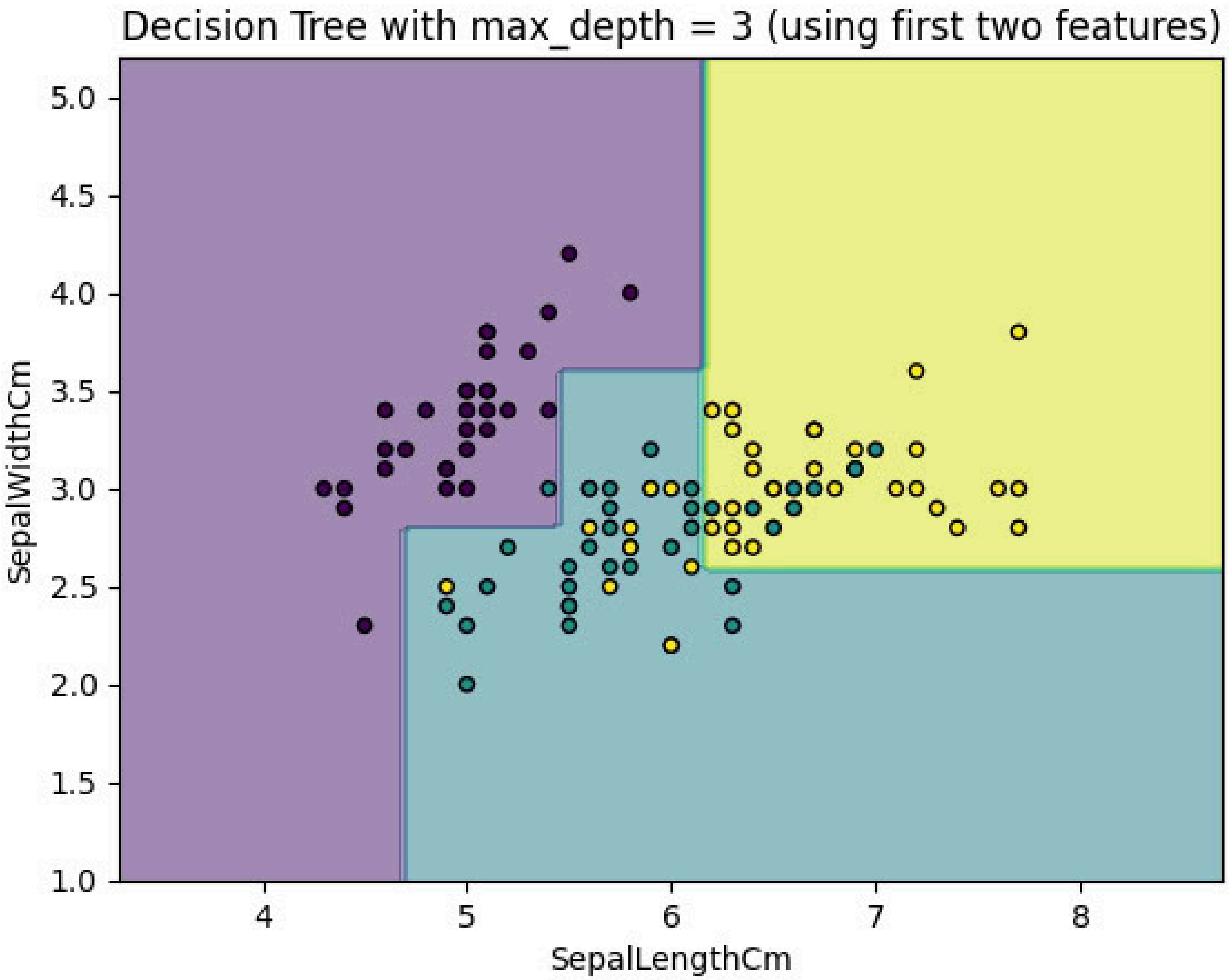
Class Separation

The model separates the classes well, indicating that depth 3 effectively captures the data's structure without overfitting.

Problem #1

Visualization of decision boundary by the use of explictly the first two features

This graph shows the decision boundaries of a decision tree classifier trained on the Iris dataset using only the first two features: Sepal Length and Sepal Width, with a maximum depth of 3.



Feature Influence

By using only the first two features, we see how these specific features contribute to class separation

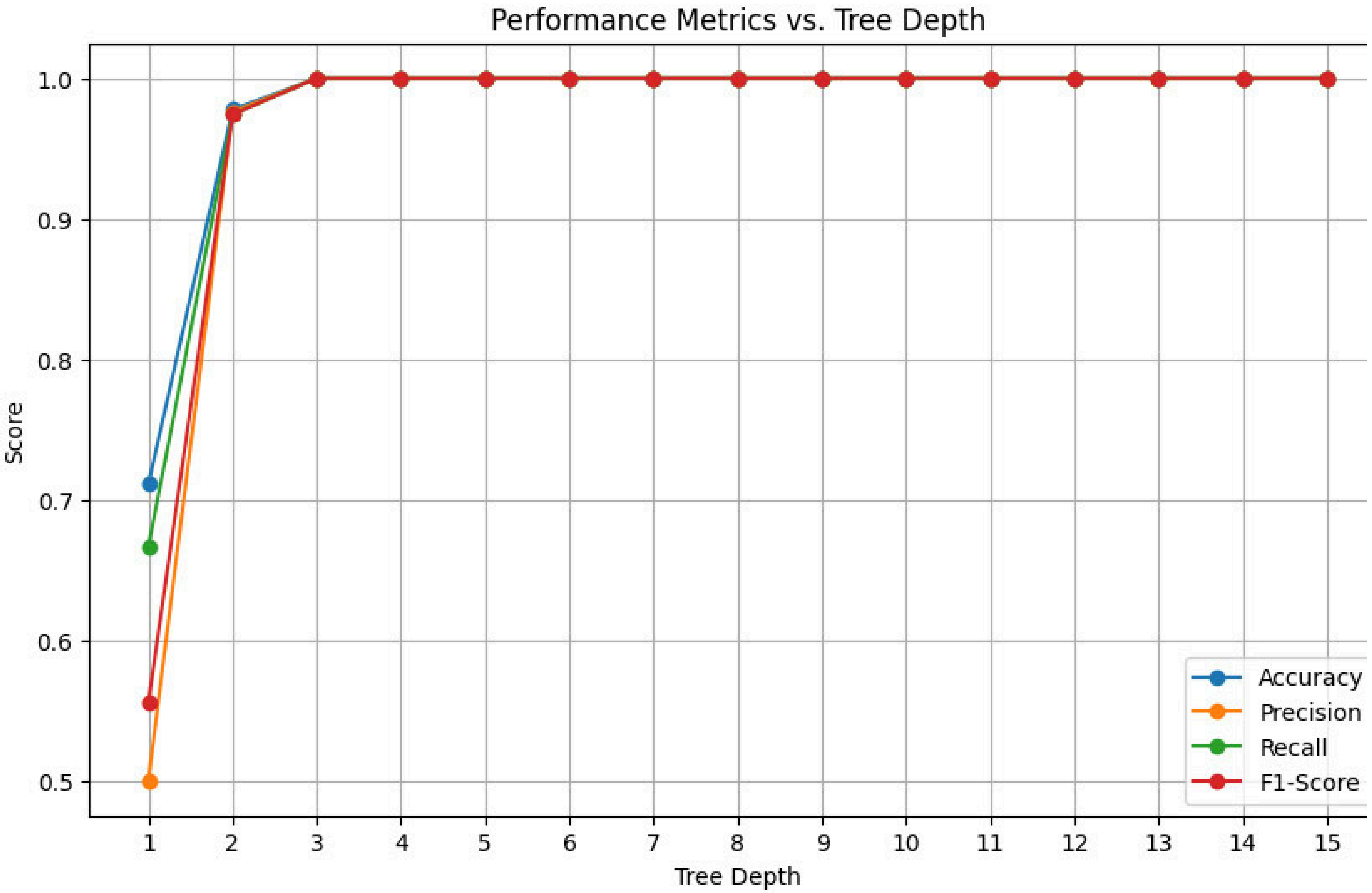
Boundary Sharpness

The boundaries are clear and straightforward, demonstrating the decision tree's decision-making process with limited feature input.

Problem #1

Performance metrics Vs Tree Depth

This plots shows how different performance metrics (accuracy, precision, recall, F1-score) vary with tree depth showing potential overfitting or underfitting.



Underfitting

Depths 1-2 are too shallow; model fails to capture data complexity, resulting in low scores.

Optimal Depth

Depth 3 achieves high performance, balancing complexity and accuracy.

No Overfitting

No performance drop observed beyond depth 4, indicating stable performance with increased depth.

Problem #2

K-Nearest Neighbors Hyperparameter Tuning

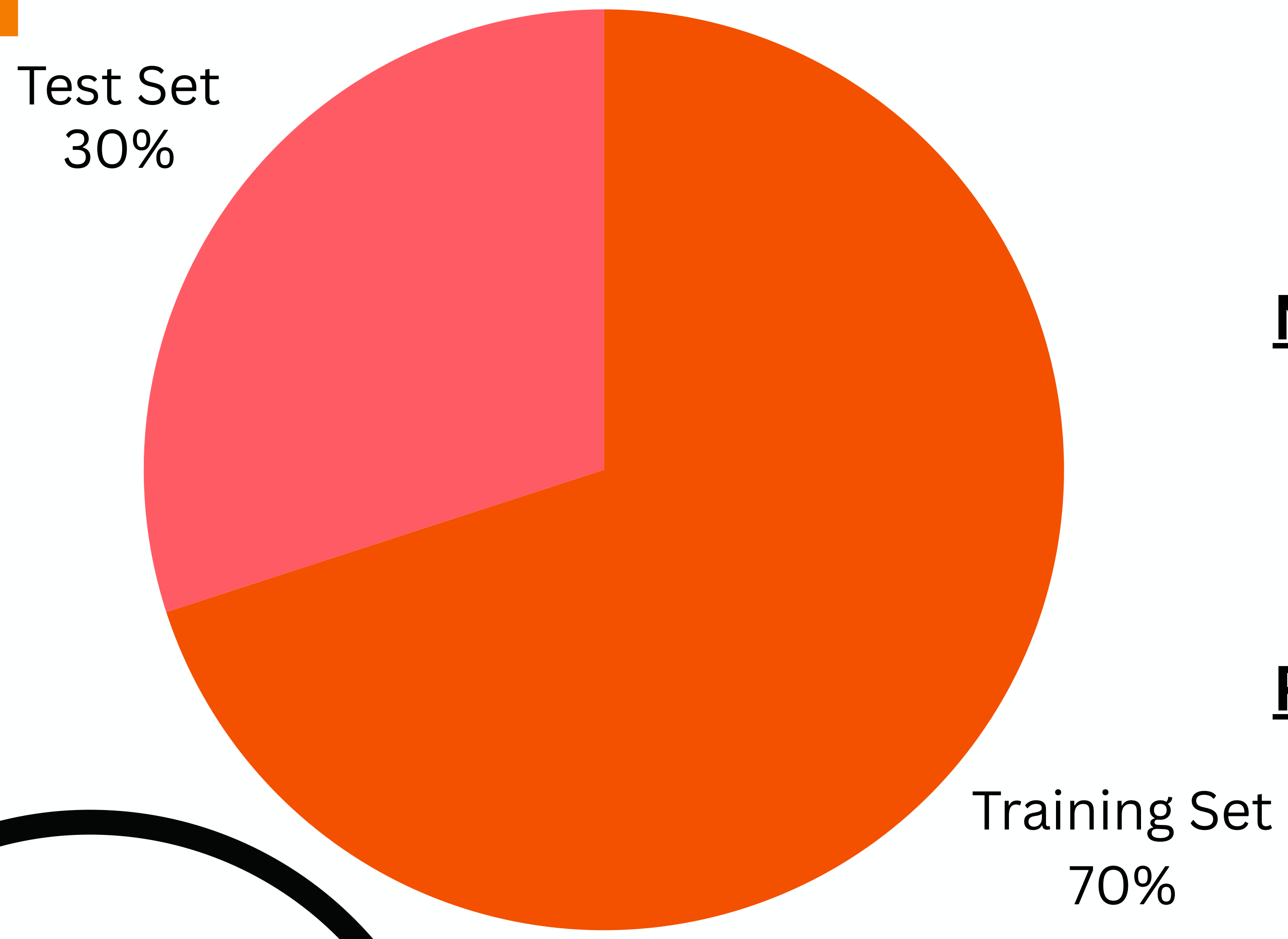
Task: Implement a k-nearest neighbors (KNN) classifier on the Iris dataset and optimize the number of neighbors (k) and distance metrics (e.g., Euclidean, Manhattan).

Instructions:

1. Split the dataset into training and testing sets.
2. Train multiple KNN classifiers with different values of k and distance metrics.
3. Evaluate the models using accuracy, precision, recall, and F1-score.
4. Visualize the decision boundaries for different k values.
5. Plot performance metrics against different k values and distance metrics.

Objective: Explore how the choice of k and distance metric impacts the performance of the KNN classifier.

Problem #2



K Values
1,3,5,7,9,11

Dataset Splitting

- Split dataset into training (70%) and testing (30%) using train_test_split.
- Ensures evaluation on unseen data.

Model Training

- Tests various k values (1, 3, 5, 7, 9, 11) and distance metrics (Euclidean, Manhattan).
- Efficiently loops through combinations.

Performance Evaluation

- Metrics: Accuracy, Precision, Recall, F1-Score.

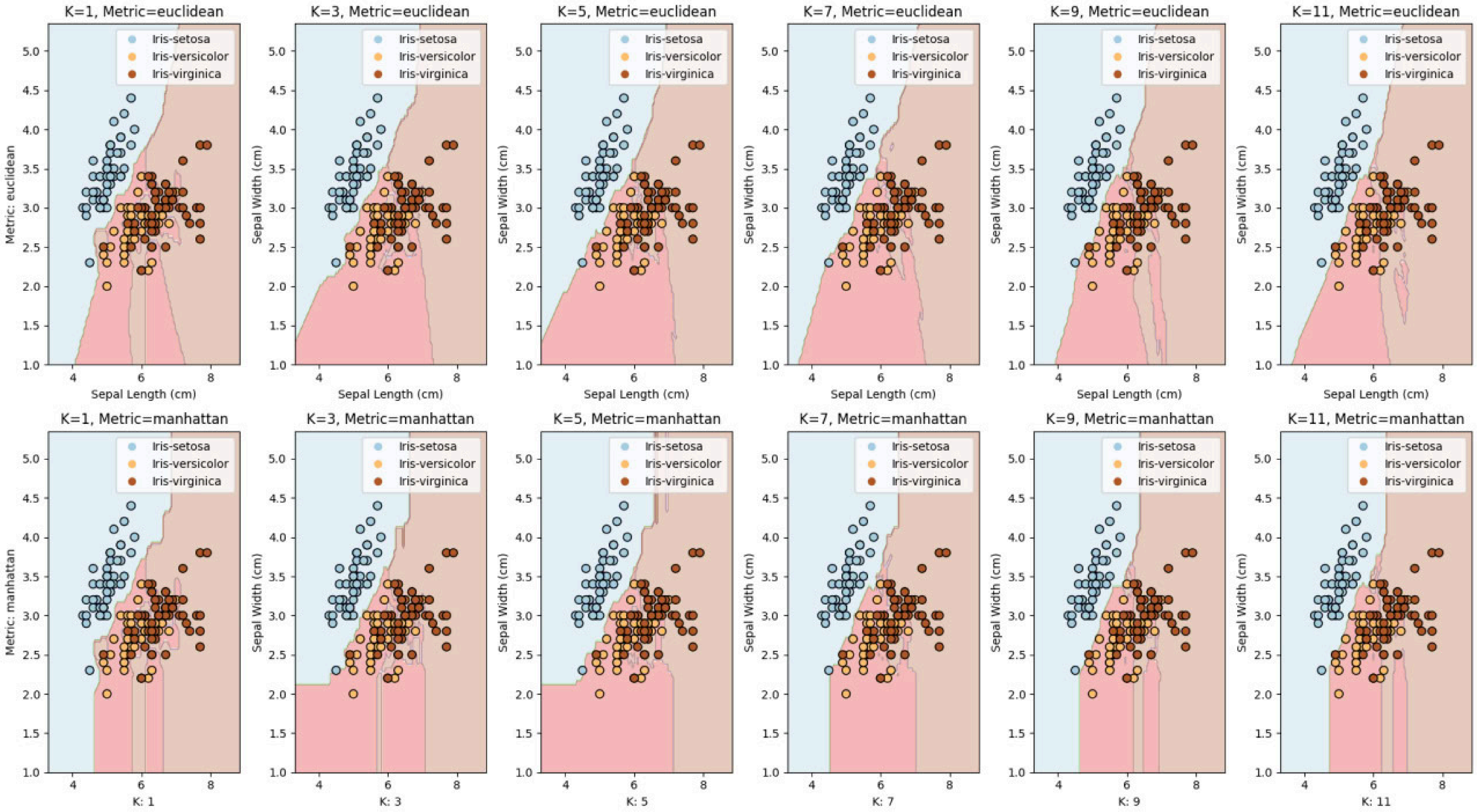
Decision Boundary Visualization

- Plots decision boundaries for different k values using Sepal Length and Width.

Note: Limited to 2 features this may not fully capture model behavior.

Problem #2

Visualization of decision boundaries of a K-Nearest Neighbors (KNN) classifier



Problem #2

The graph depicts the decision boundaries of a K-Nearest Neighbors (KNN) classifier on the Iris dataset, using different values of k (number of neighbors) and two different distance metrics: Euclidean and Manhattan.

Effect of k (Number of Neighbors)

- ***Low k***: Jagged, complex boundaries.

  Model is highly sensitive to individual data points

  E.g., k=1k=1k=1
- ***High k***: Smoother, more generalized boundaries.

  Model captures broader patterns

  E.g., k=11k=11k=11

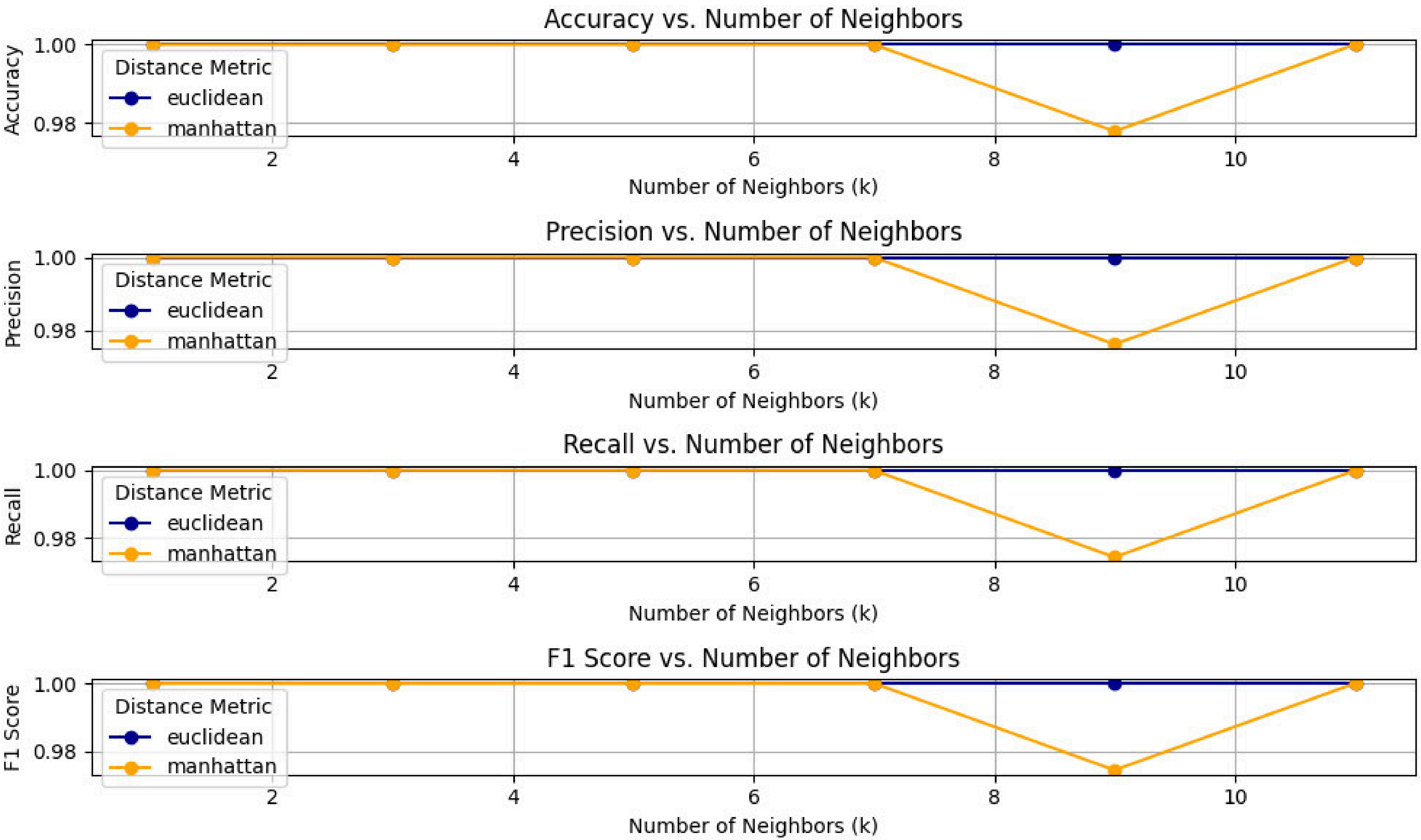
Effect of Distance Metric

- ***Euclidean Distance (Top Row)***: Produces smoother, rounded boundaries.
- ***Manhattan Distance (Bottom Row)***: Results in blocky, rectilinear boundaries.
- ***Reason***: Euclidean distance uses straight-line measurements, while Manhattan distance follows grid lines.

Problem #2

Performance Metrics vs. k and Distance Metric.

Four line plots display how performance metrics vary with different k values and distance metrics.



Consistency of Euclidean Metric

Maintains high performance across all ***k*** values for Accuracy, Precision, Recall, and F1 Score.

Instability of Manhattan Metric

Significant performance drop at ***k=9***, indicating issues with predictive power at this k value.

General Behavior

Performance generally stable as k increases, except for the Manhattan metric at ***k=9***, influenced by the nature of the dataset and distance calculation.

Problem #3

Perceptron Learning Algorithm

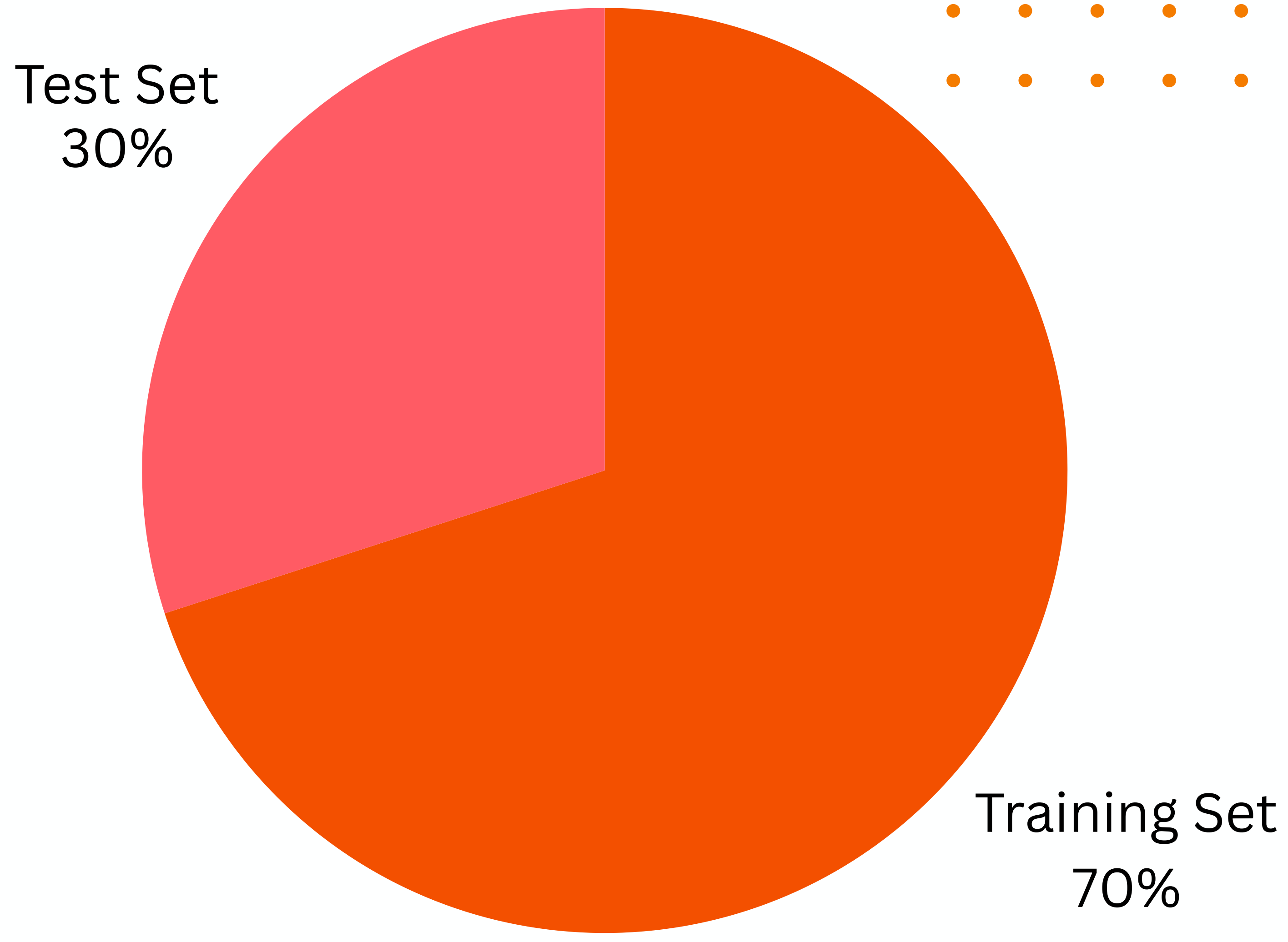
Task: Train a perceptron classifier on the Iris dataset and analyze its performance.

Instructions:

1. Split the dataset into training and testing sets.
2. Train a perceptron classifier on the training set.
3. Evaluate the model using accuracy, precision, recall, and F1-score.
4. Visualize the decision boundaries of the perceptron.
5. Plot the convergence of the perceptron learning algorithm over iterations.

Objective: Gain an understanding of how the perceptron learning algorithm works and its
limitations.

Problem #3

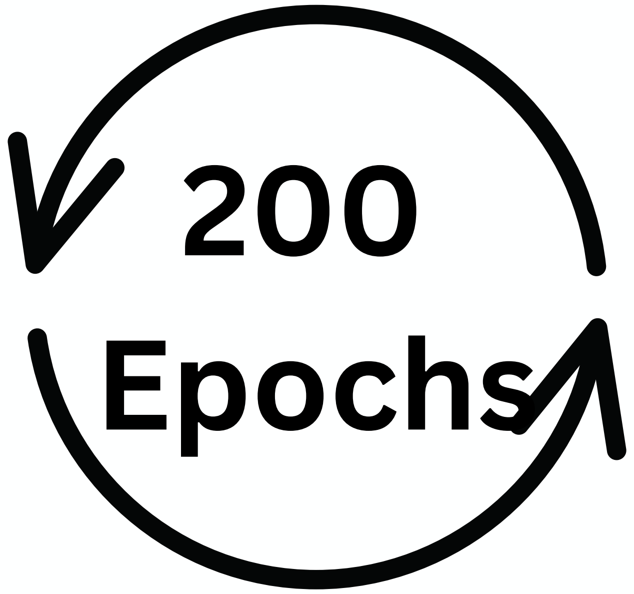


Dataset Splitting

- Split dataset into training (70%) and testing (30%) using train_test_split.
- Ensures evaluation on unseen data.

Training

- A Perceptron model is initialized with ***warm_start=True*** to allow incremental training and track convergence.
- The Perceptron is trained over ***200 epochs.*** During each epoch, it fits the model to the training data and records the number of misclassifications.



200 Epochs

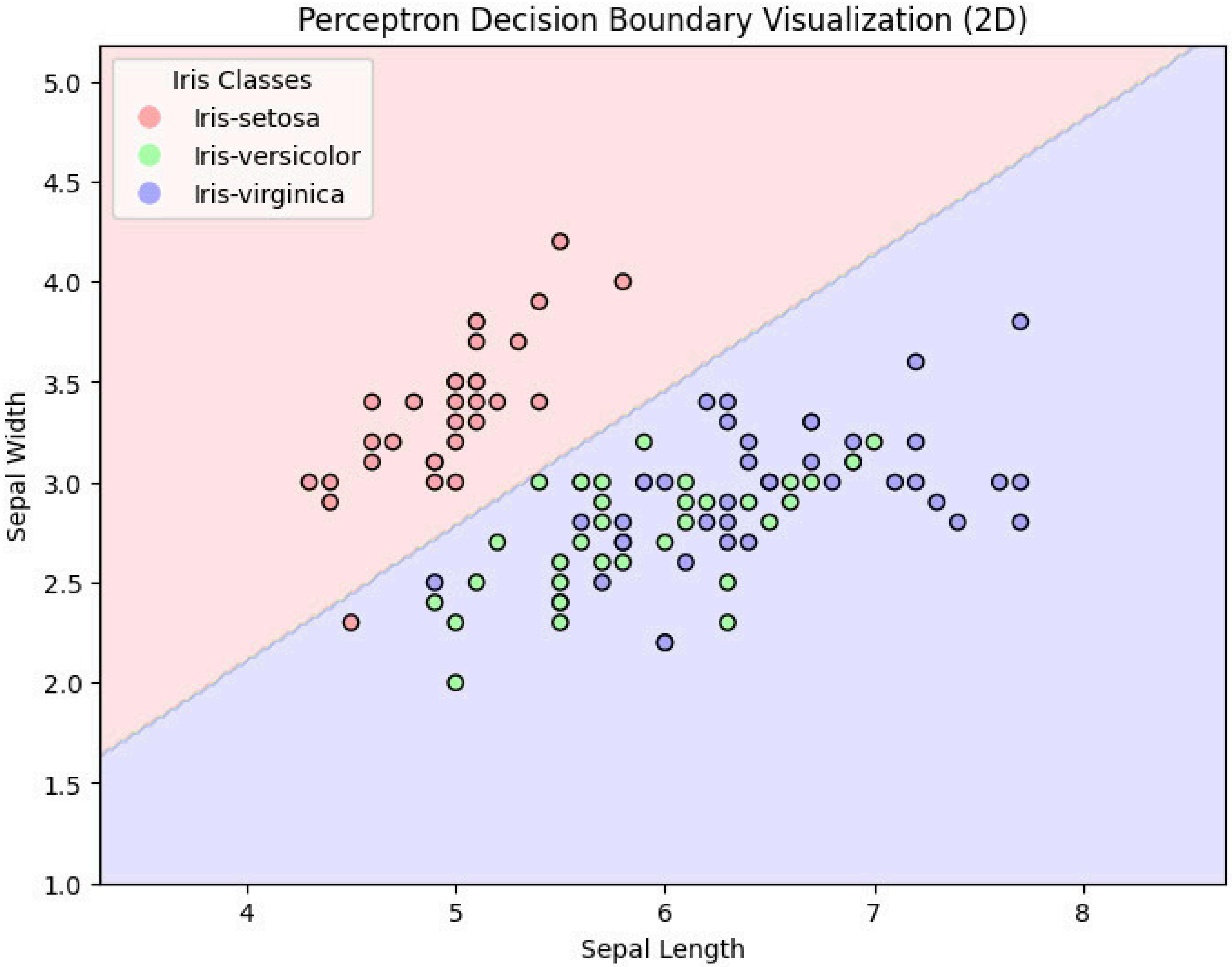
Performance Evaluation

- Metrics: Accuracy, Precision, Recall, F1-Score.
  - ***Accuracy*: *0.76*** ***Precision*: *0.77***
  - ***Recall*: *0.72*** ***F1 Score*: *0.70***

Problem #3

Visualization of Perceptron Decision Boundaries

This plot shows how the Perceptron classifier has separated the data points in a 2D space based on Sepal Length and Sepal Width.



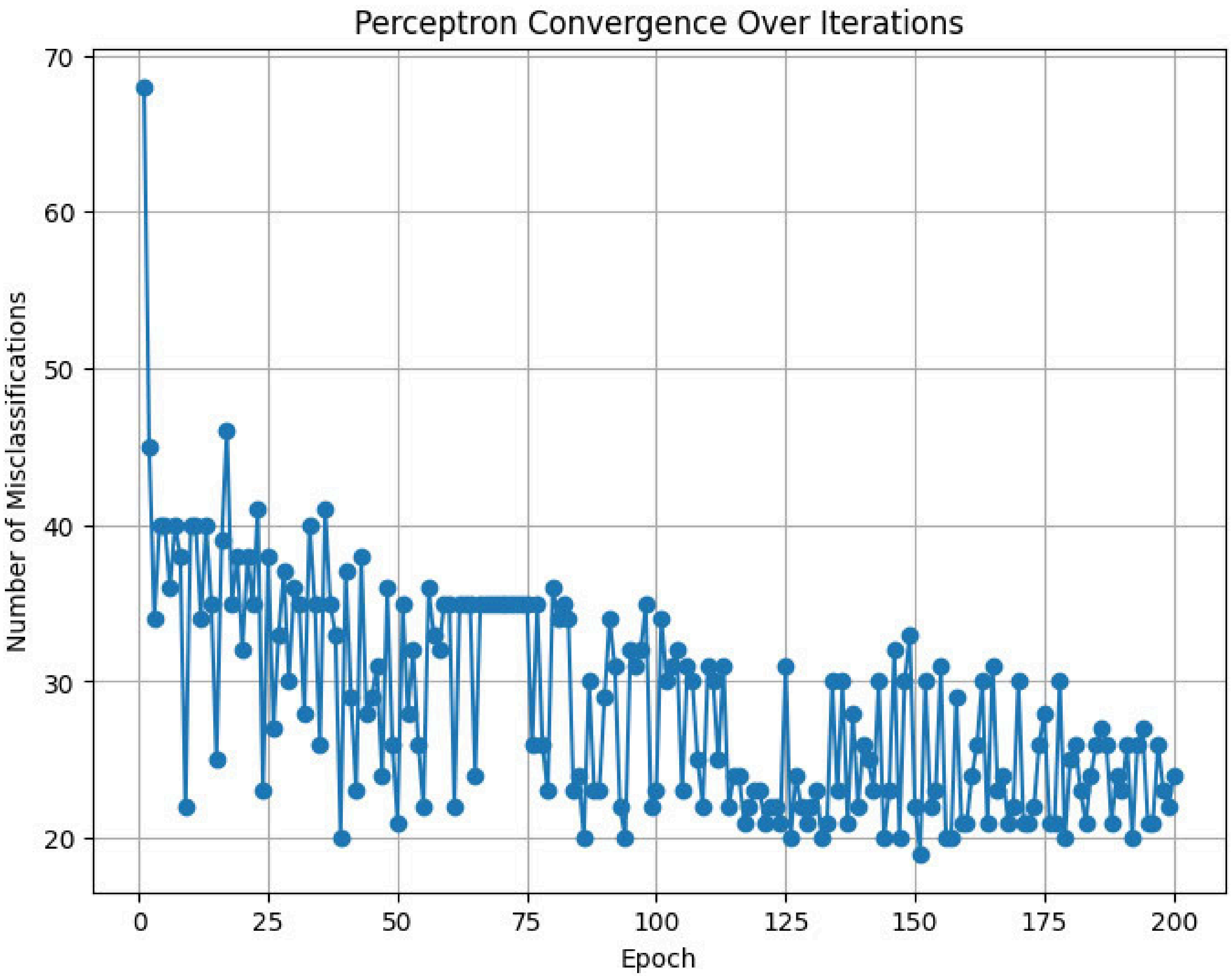
Linear Separability

The graph shows how effectively the perceptron separates Iris-setosa from the other classes based on sepal dimensions.
It visually demonstrates the linear separability of the classes in the feature space.

Problem #3

Visualization of Convergence

This plot tracks the number of misclassifications made by the Perceptron model over each epoch (iteration).



Initial Phase
High number of misclassifications, indicating poor training.

Training Progress
Decreasing misclassifications show improved accuracy as the model learns.

Fluctuations
Some instability in convergence, likely due to data noise or complexity.

Final Phase
Misclassifications stabilize, indicating successful convergence.

Problem #4

Comparing Decision Tree, KNN, and Perceptron

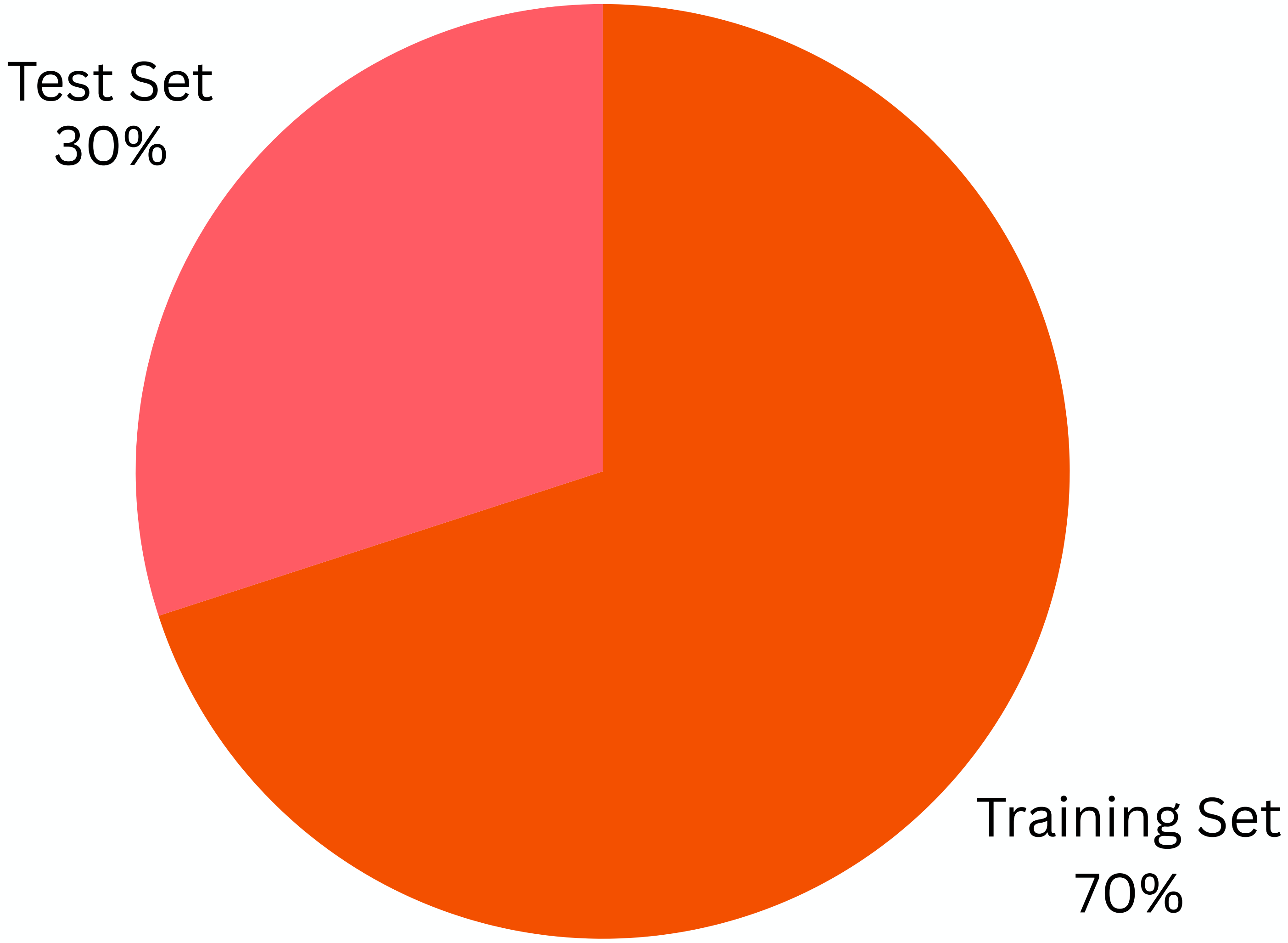
Task: Compare the performance of decision tree, KNN, and perceptron classifiers on the Iris dataset.

Instructions:

1. Split the dataset into training and testing sets.
2. Train a decision tree, KNN, and perceptron classifier on the training set.
3. Evaluate each model using accuracy, precision, recall, and F1-score.
4. Visualize the decision boundaries for all three classifiers.
5. Plot the performance metrics for each classifier.

Objective: Understand the strengths and weaknesses of each classifier and how they perform on the same dataset.

Problem #4



Dataset Splitting

- Split dataset into training (70%) and testing (30%) using train_test_split.
- Ensures evaluation on unseen data.

Classifier Training

- Three classifiers are initialized:
  - ○ *DecisionTreeClassifier(),*
  - ○ *KNeighborsClassifier(),*
  - ○ *Perceptron().*
- Each classifier is trained on the training data using the . *fit()* method.

Performance Evaluation

- ***Metrics:*** Accuracy, Precision, Recall, F1-Score.

- ***Decision Tree:*** provides interpretable results with its depth indicating model complexity.
- ***KNN***: perform well with varying numbers of neighbors but can be computationally expensive.
- ***Perceptron***: As a simple linear model, does not perform as well on this dataset unless the classes are linearly separable.

Problem #4

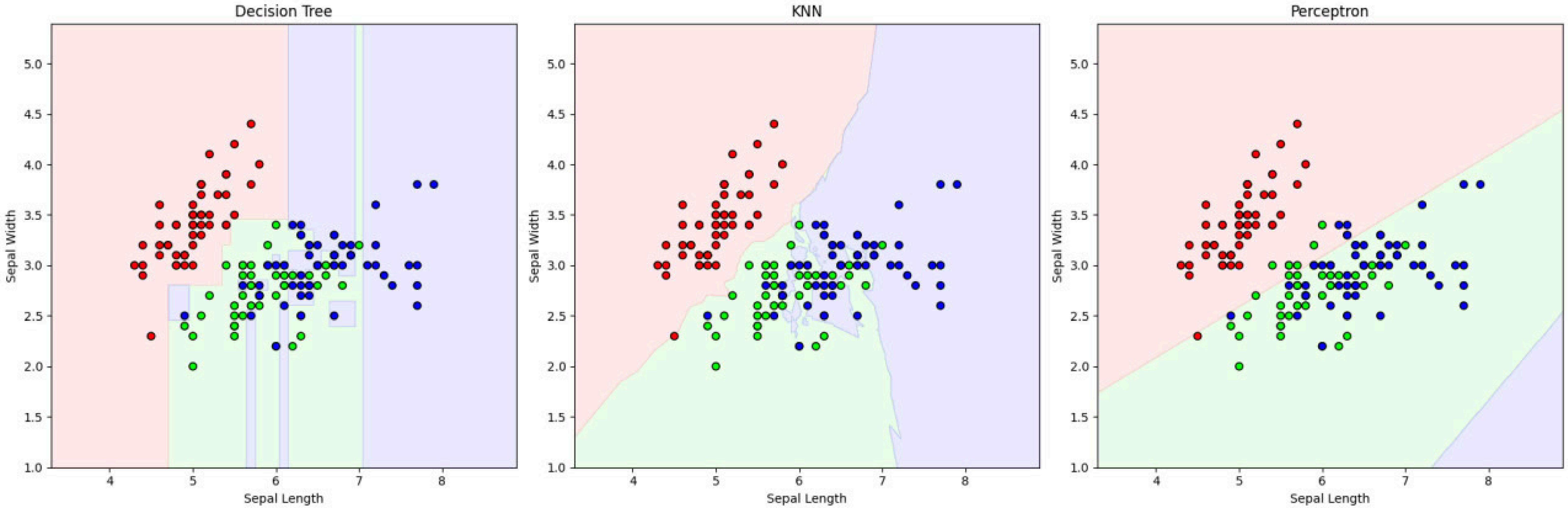
Model Performance

Classifier	Accuracy	Precision	Recall	F1 Score
Decision Tree	1.0	1.000000	1.000000	1.000000
KNN	1.0	1.000000	1.000000	1.000000
Perceptron	0.8	0.863636	0.769231	0.737815

Problem #4

Visualization of the decision boundaries for all three classifiers

This image consists of three subplots, each showing the decision boundaries created by the Decision Tree, KNN, and Perceptron classifiers..



Decision Tree (Left Plot)

- ***Characteristics:*** Segmented and irregular regions, indicating potential overfitting.
- ***Features:*** Different colors for classes based on Sepal Length and Width.

K-Nearest Neighbors (Middle Plot)

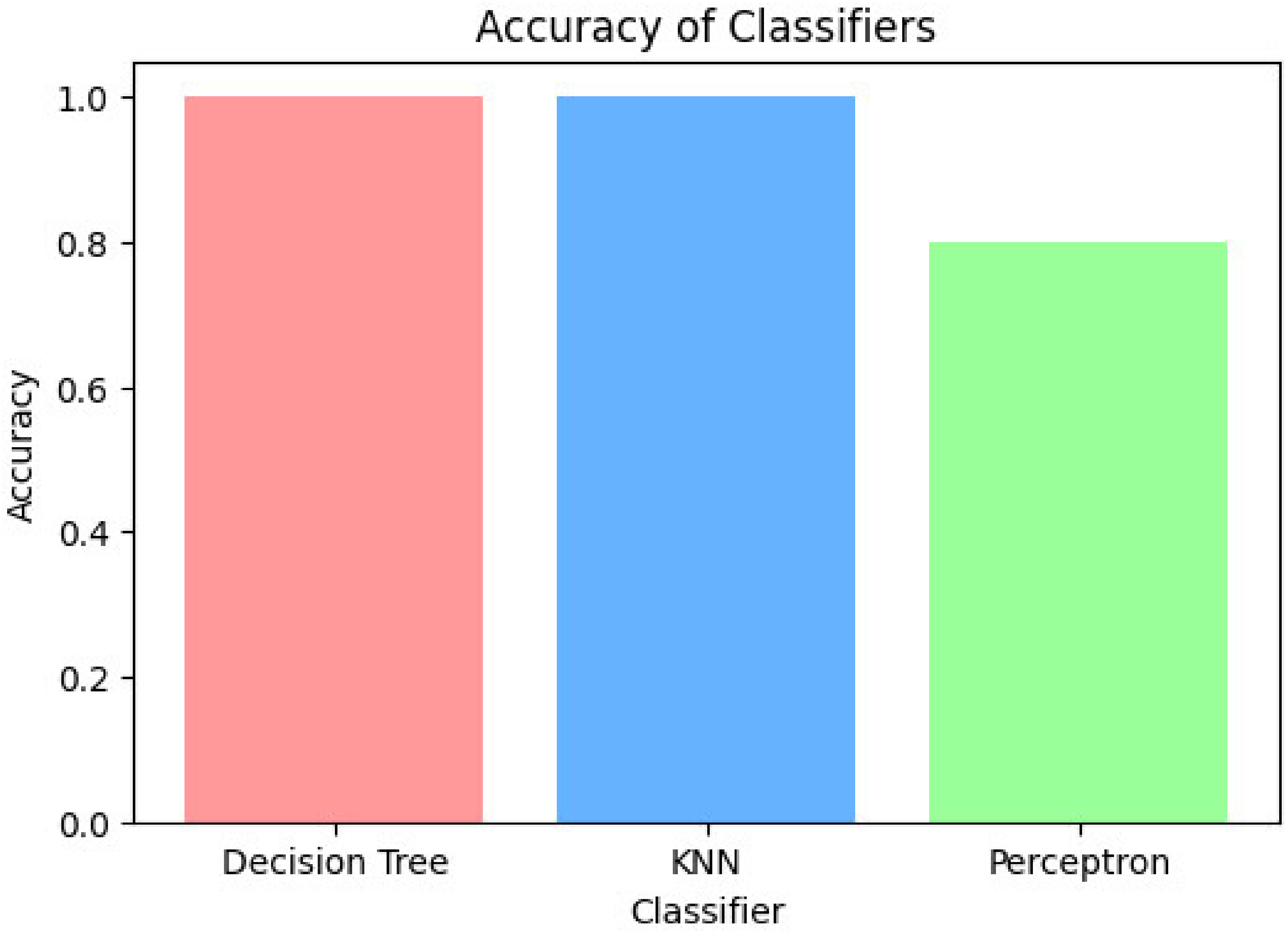
- ***Characteristics:*** Smoother and more continuous boundaries.
- ***Features:*** Reflects local neighborhood reliance, suggesting better generalization.

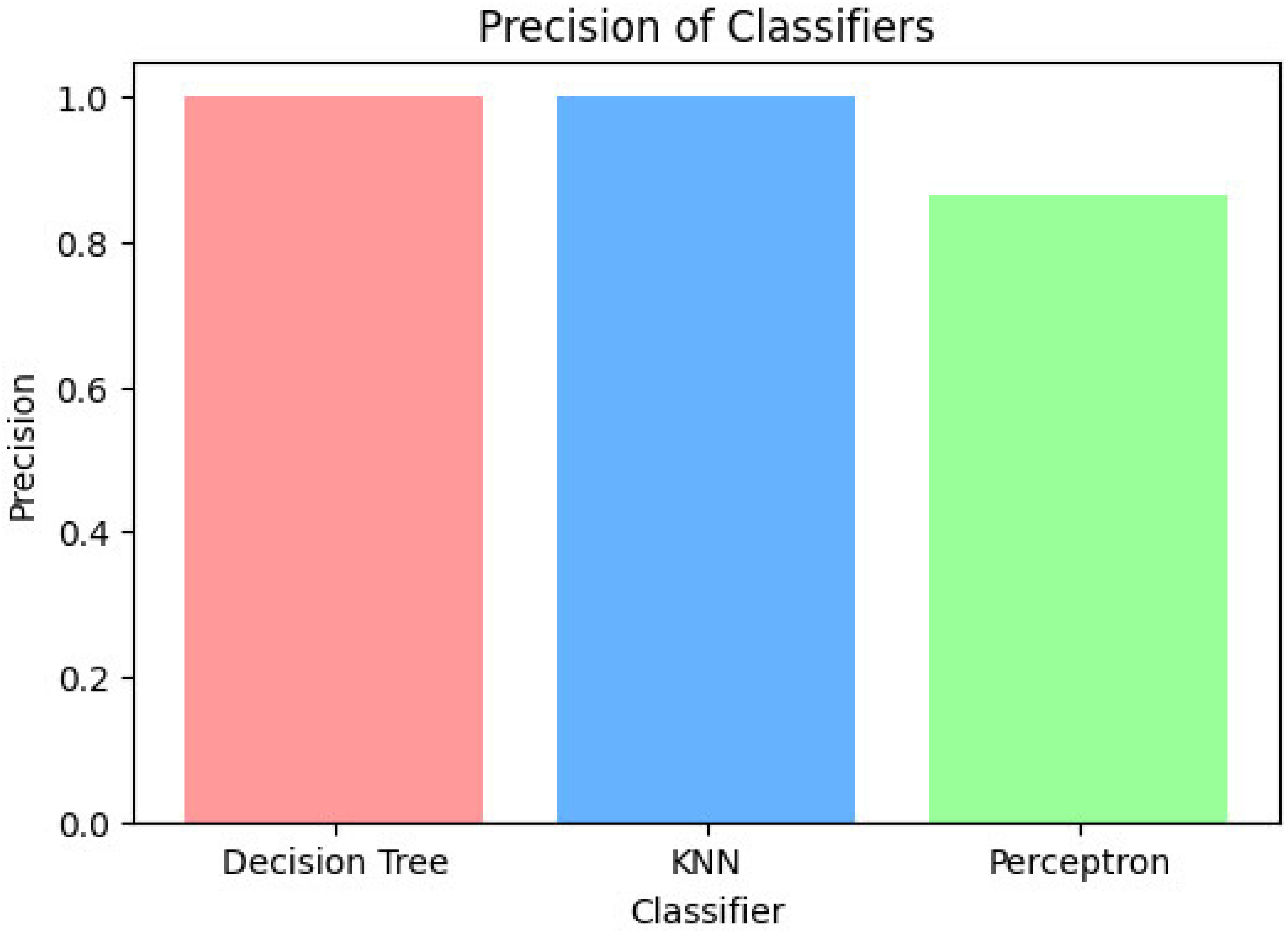
Perceptron (Right Plot)

- ***Characteristics:*** Linear boundaries, limited to linearly separable classes.
- ***Features:*** Struggles with complex regions, failing to classify all points correctly.

Problem #4

Visualization of the performance metrics for each classifier





Accuracy

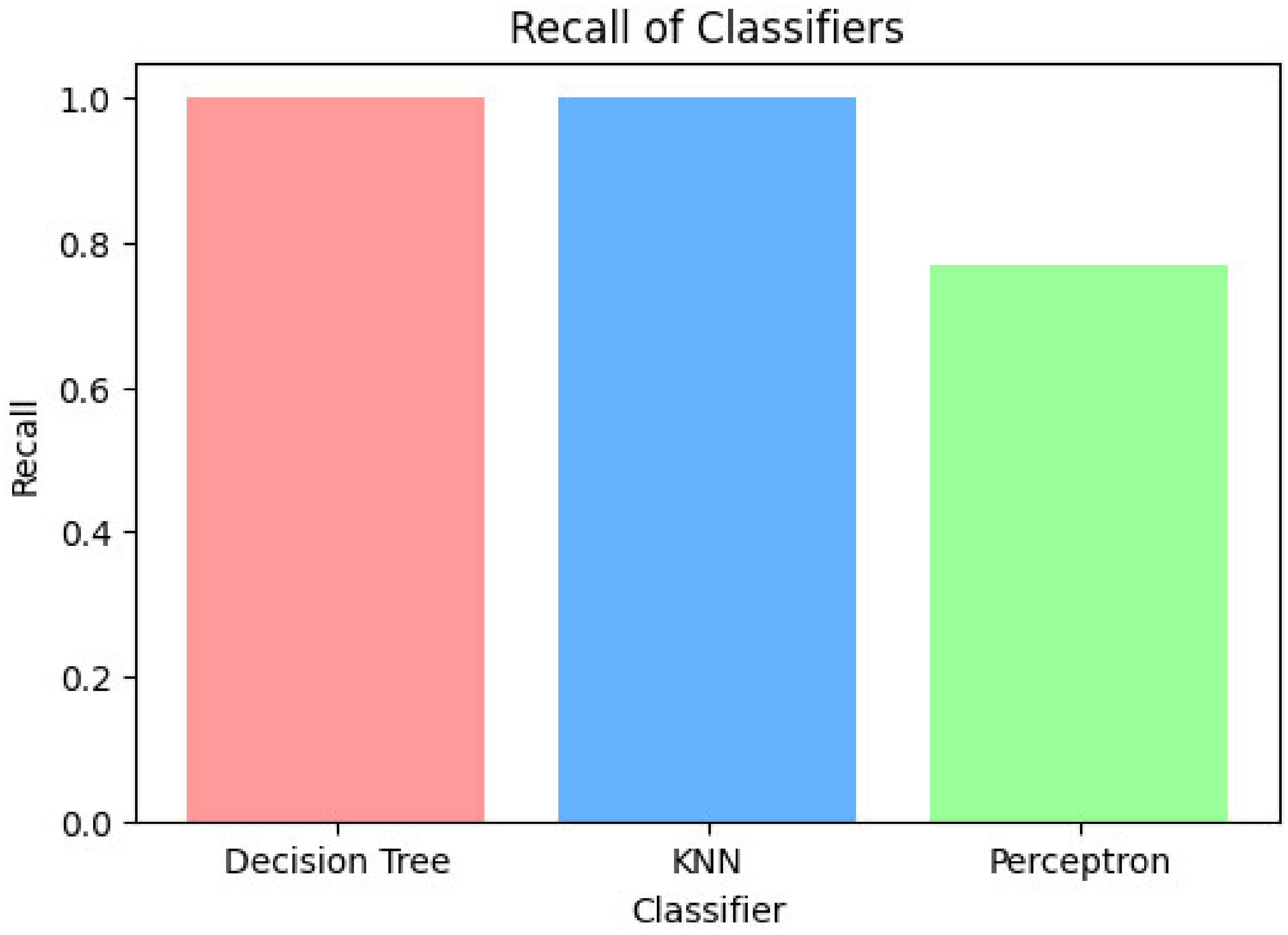
- Proportion of correctly classified instances out of total instances.
- ***Chart Insight:*** Decision Tree and KNN classifiers show near-perfect accuracy.
  - Perceptron has lower accuracy.

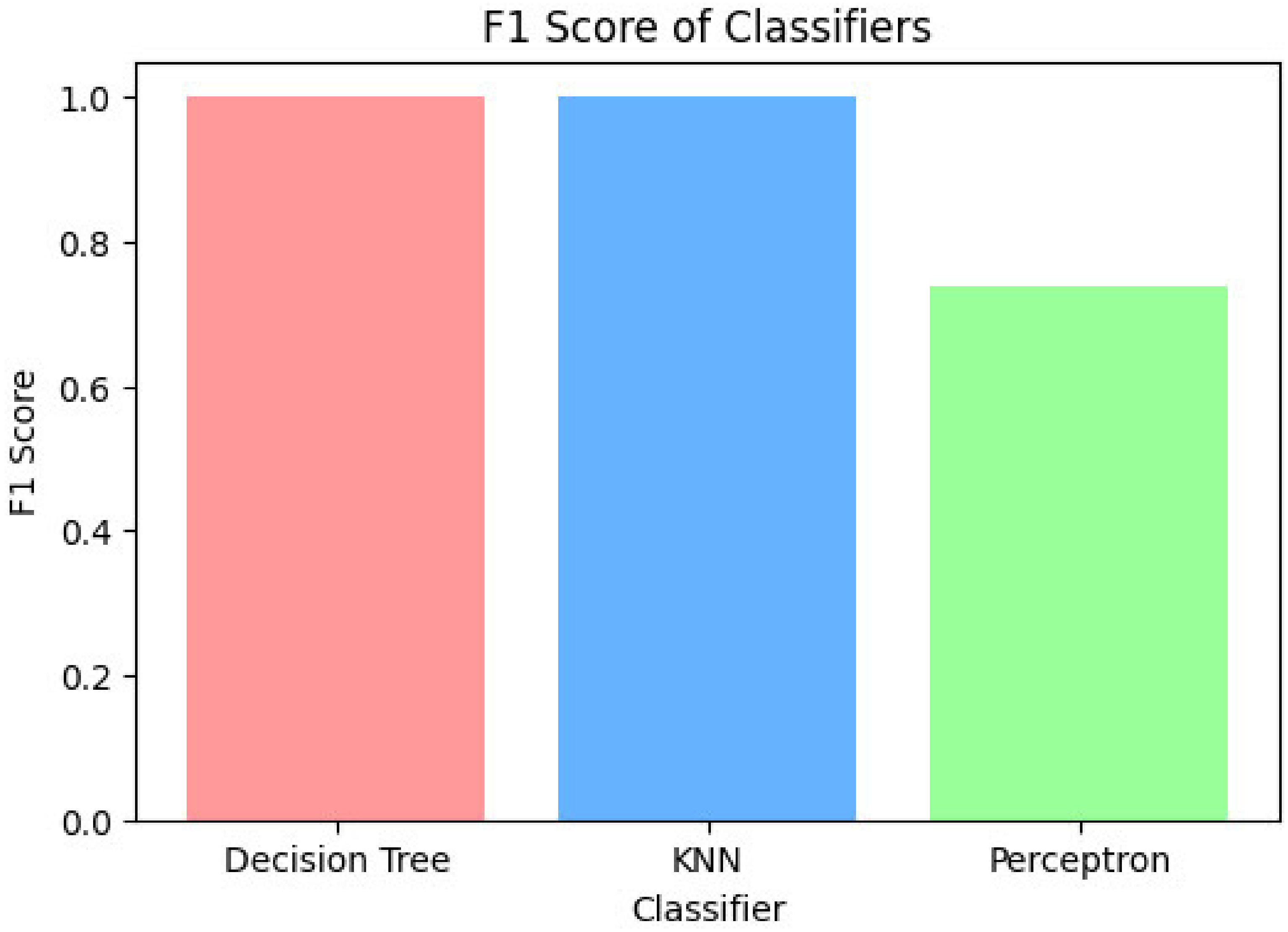
Precision

- Proportion of true positive predictions among all positive predictions made.
- ***Chart Insight:*** High for all classifiers;
  - Slightly lower for Perceptron.

Problem #4

Visualization of the performance metrics for each classifier





Recall

- Proportion of actual positives correctly identified.
- ***Chart Insight:*** High for Decision Tree and KNN; lower for Perceptron.

F1 Score

- Harmonic mean of precision and recall, indicating Perceptron struggles more with this dataset.
- ***Chart Insight:*** Highest for Decision Tree and KNN; lower for Perceptron.

Problem #4

Visualization of the performance metrics for each classifier

- The ***Decision Tree*** and ***KNN*** classifiers are performing better overall, as seen by the high values in accuracy, precision, recall, and F1 score, and their decision boundaries are more aligned with the data.

- The ***Perceptron*** classifier, being a simple linear model, is less effective, particularly on this dataset where the decision boundaries are not linear.

Problem #5

K-Means Clustering and Visualization

***Task*:** Apply k-means clustering to the Iris dataset and visualize the results.

***Instructions*:**

1. Normalize the Iris dataset.
2. Apply k-means clustering with k=3.
3. Visualize the cluster assignments and the centroids in 2D and 3D plots.
4. Compare the cluster assignments with the actual class labels.
5. Plot the silhouette scores to evaluate the quality of the clusters.

***Objective*:** Understand how k-means clustering works and how to evaluate clustering performance.

Problem #5

Normalization

Uses ***StandardScaler*** to normalize features.
This ensures that all features *(Sepal Length, Sepal Width, Petal Length, Petal Width)* are scaled to have 0 mean and 1 variance, which is crucial for clustering algorithms to function correctly.

K-Means Clustering

Applied with *KMeans(n_clusters=3)* and *random_state=42.*

Visualization

- ***2D Plot:*** PCA reduces data to 2D; clusters and centroids are shown.
- ***3D Plot***: Scatter plot of clusters and centroids using Sepal Length, Sepal Width, and Petal Length.

Cluster Comparison

Maps clusters to actual species, calculates confusion matrix and accuracy to evaluate clustering performance.

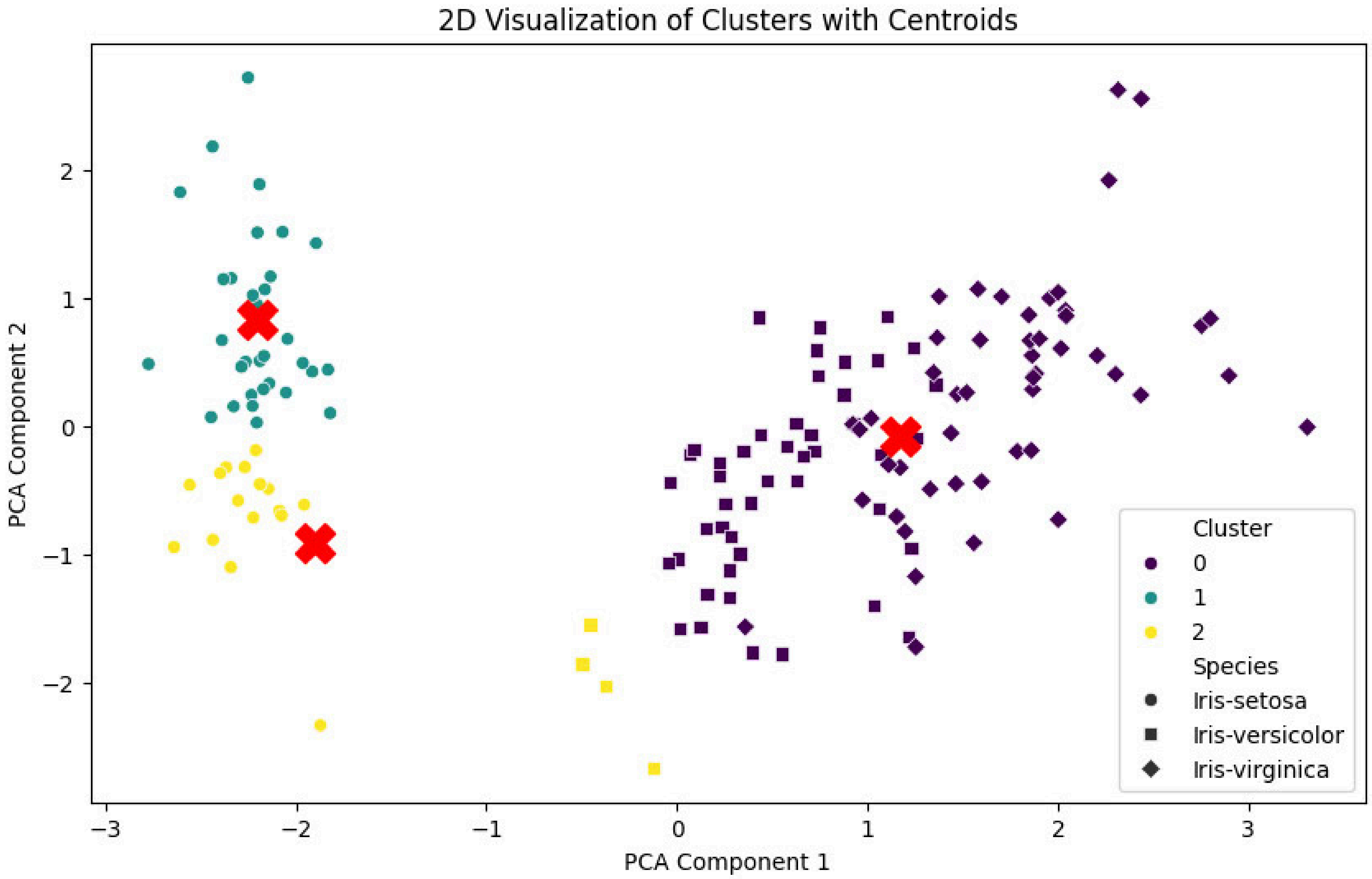
Output:

Accuracy Score: 0.6666666666666666
Average Silhouette Score: 0.4787241921049546

Problem #5

2D Visualization of Clusters with Centroids

This scatter plot shows the results of k-means clustering on the Iris dataset, reduced to two dimensions using PCA



This graph helps to visually assess how well the data points are grouped into clusters in a reduced 2D space.

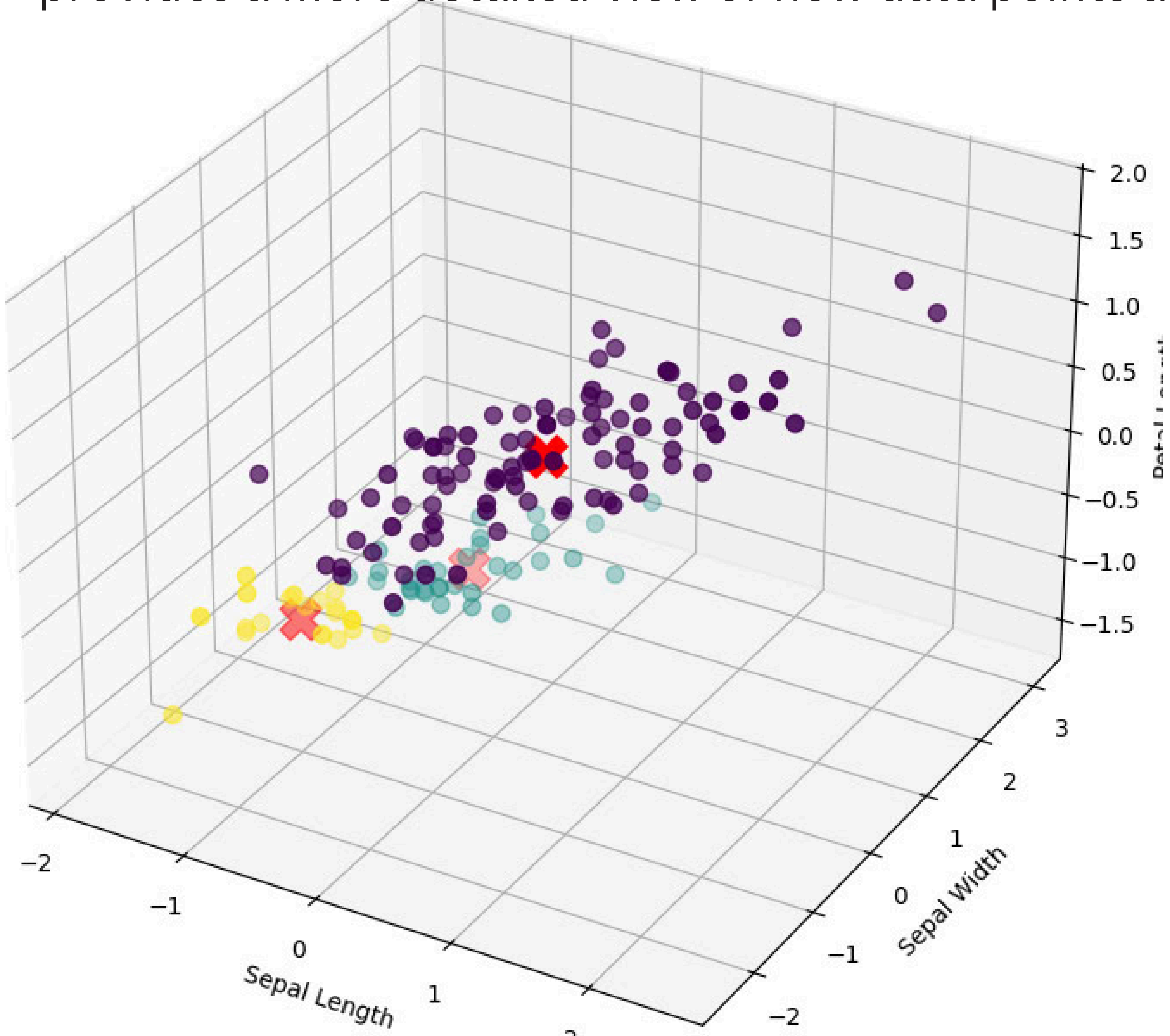
The proximity of points within the same cluster and the separation between different clusters can indicate how distinct the clusters are.

It also shows how closely the clusters align with the actual species labels.

Problem #5

3D Visualization of Clusters with Centroids

This 3D scatter plot visualizes the clustering using three features of the dataset. This provides a more detailed view of how data points are grouped in higher dimensions.



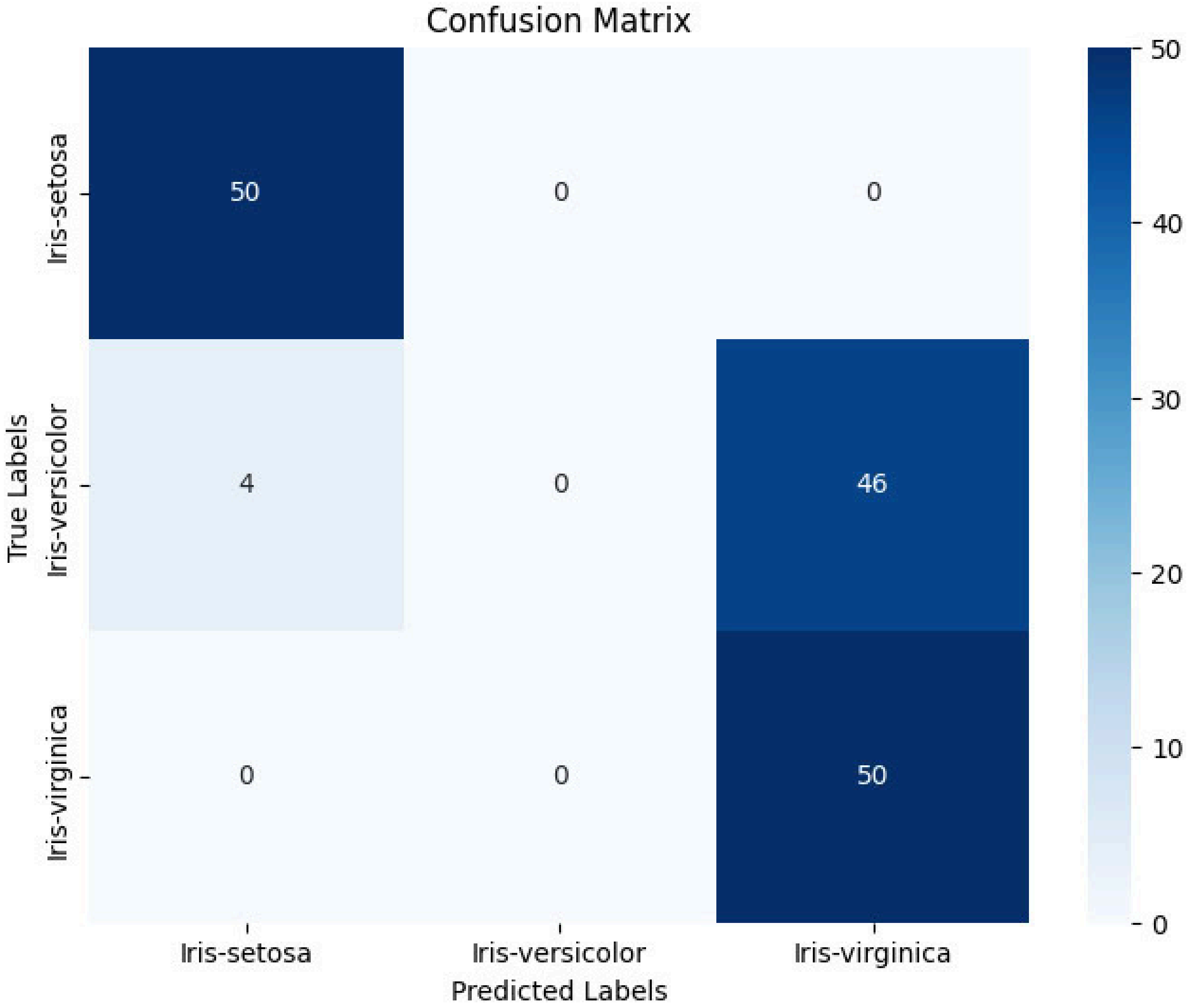
By adding an extra dimension, this plot provides a more nuanced understanding of the clustering. It can reveal overlapping clusters or outliers that may not be as apparent in 2D.

This can help in evaluating the effectiveness of the clustering when more features are considered.

Problem #5

Confusion Matrix

This heatmap compares the predicted cluster assignments to the actual species labels. It shows how well the clustering aligns with the true classifications of the Iris dataset. The diagonal elements indicate correct predictions.



	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	50	0	0
Iris-versicolor	4	0	46
Iris-virginica	0	0	50

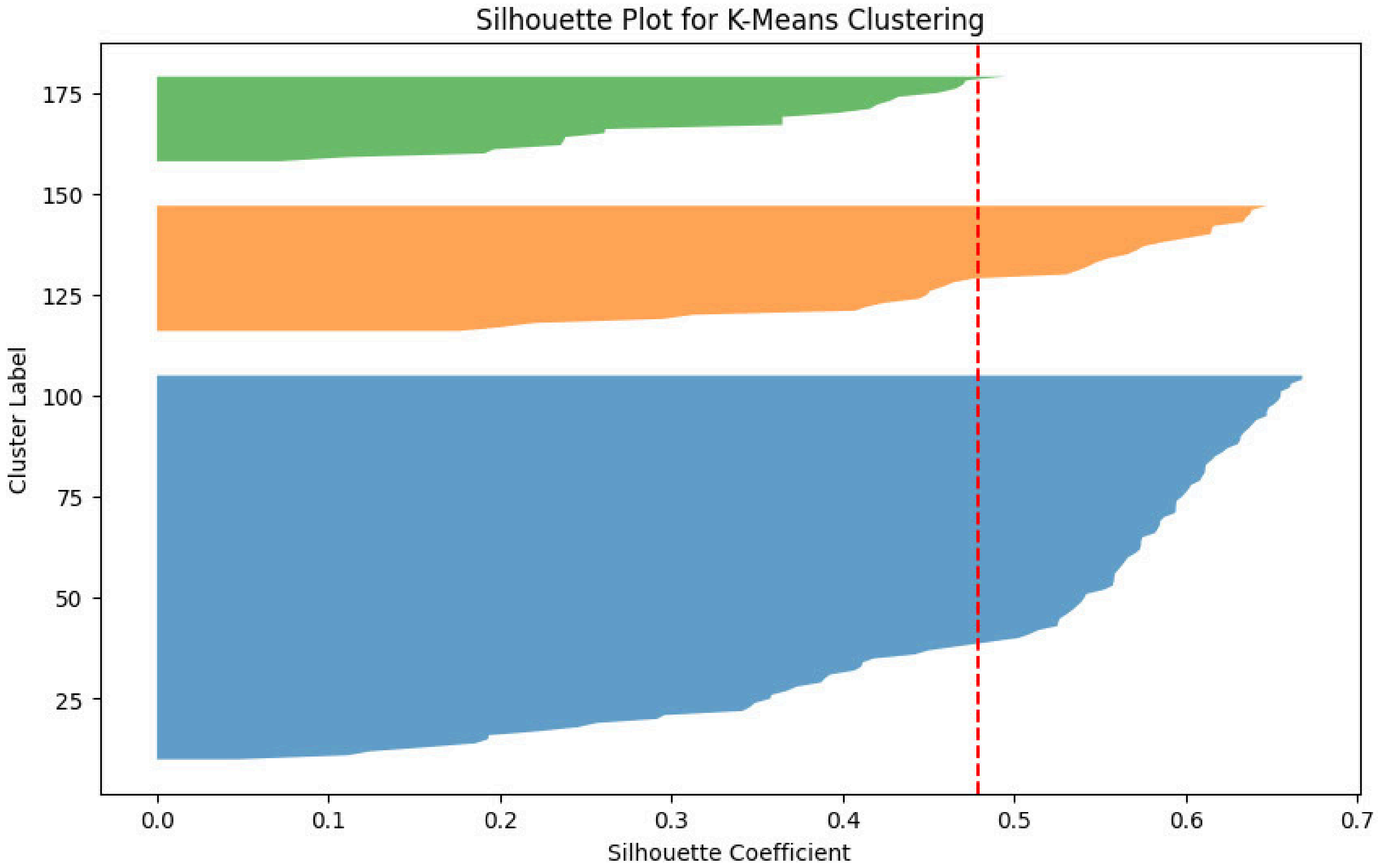
It quantitatively evaluates the clustering accuracy by comparing predicted clusters to actual species.

The matrix shows which species are most often confused with each other, highlighting potential weaknesses in the clustering algorithm, such as misclassifications between similar species.

Problem #5

Silhouette Plot for K-Means Clustering

This plot shows the silhouette coefficients for each sample in the dataset. It visualizes how well-separated the clusters are, with a vertical red dashed line indicating the average silhouette score.



This plot provides a measure of how well-separated the clusters are. A higher silhouette score indicates better-defined clusters.

The plot also reveals the consistency within clusters, helping to identify clusters that might be too broad or too tight.

Problem #6

K-Medoids Clustering and Comparison with K-Means

Task: Apply k-medoids clustering to the Iris dataset and compare it with k-means clustering.

Instructions:

1. Normalize the Iris dataset.
2. Apply k-medoids clustering with k=3.
3. Visualize the cluster assignments and the medoids in 2D and 3D plots.
4. Compare the cluster assignments with the actual class labels.
5. Compare the performance of k-means and k-medoids using silhouette scores and other clustering metrics.

Objective: Understand the differences between k-means and k-medoids clustering and how they perform on the same dataset.

Problem #6

Normalization

Uses ***StandardScaler*** to normalize features.

This ensures that all features *(Sepal Length, Sepal Width, Petal Length, Petal Width)* are scaled to have 0 mean and 1 variance, which is crucial for clustering algorithms to function correctly.

K-Medoids Clustering

Initializes medoids, assigns points to nearest medoid, updates medoids to minimize intra-cluster distances, and stops when medoids stabilize or maximum iterations are reached.

Visualization

- ***2D Plot*:** Sepal Length vs. Sepal Width with clusters and medoids/centroids marked.
- ***3D Plot*:** Sepal Length, Sepal Width, and Petal Length with clusters and medoids/centroids highlighted.

Comparison with Actual Class Labels

- ***Confusion Matrix*:** Shows alignment with true classes.
- ***Purity*:** Fraction of correctly classified points in each cluster.
- ***Accuracy***: Additional comparison metric.

Problem #4

Performance Comparison (K-Means vs. K-Medoids)

***Silhouette Score*:** Measures cluster cohesion vs. separation.

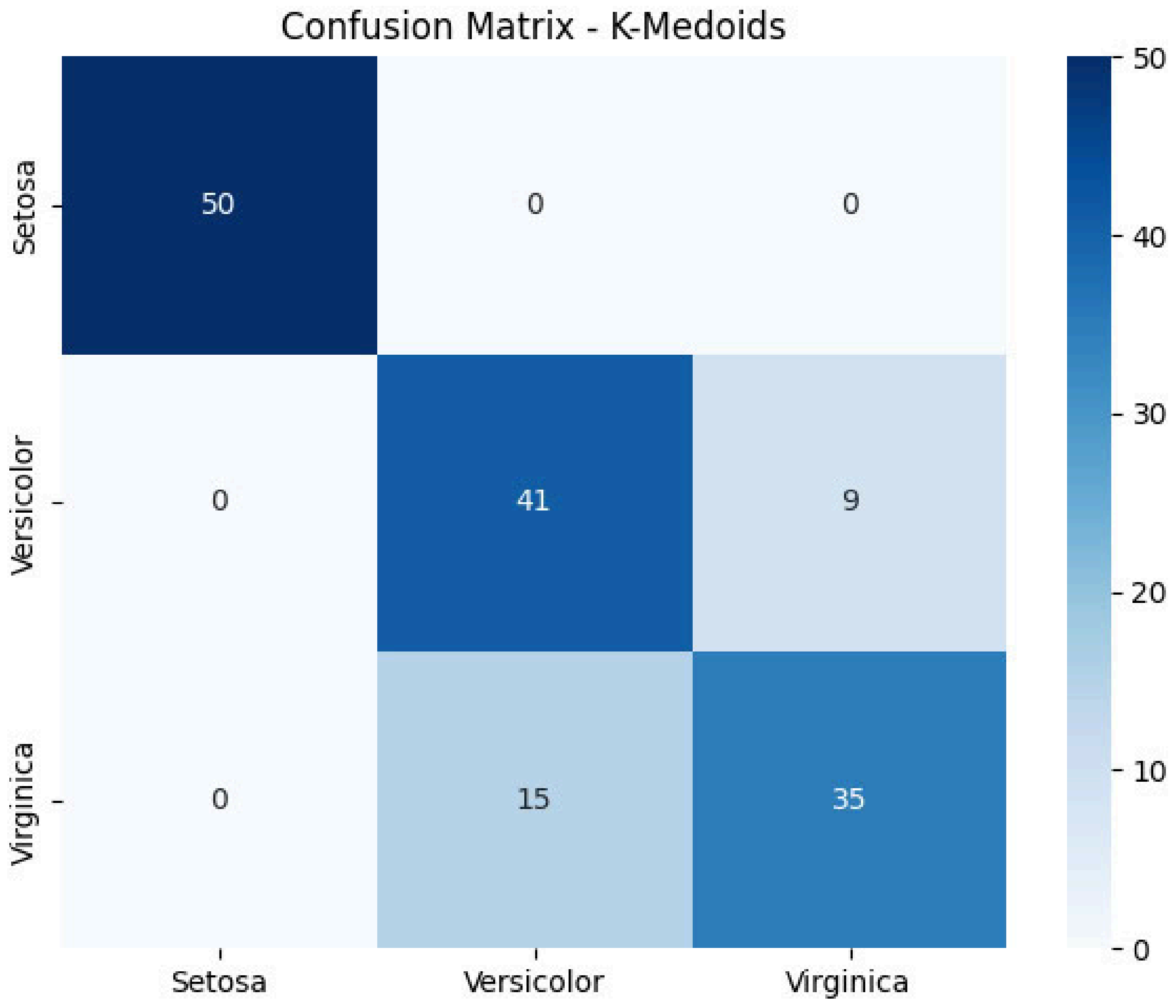
***Adjusted Rand Index (ARI)*:** Compares similarity between true labels and cluster assignments.

Output:

```
Convergence reached after 5 iterations.
Silhouette Score for K-Medoids: 0.4593
Silhouette Score for K-Means: 0.4787
Adjusted Rand Index for K-Medoids: 0.6676
Adjusted Rand Index for K-Means: 0.4290
Purity for K-Medoids: 0.8600
Purity for K-Means: 0.6667
```

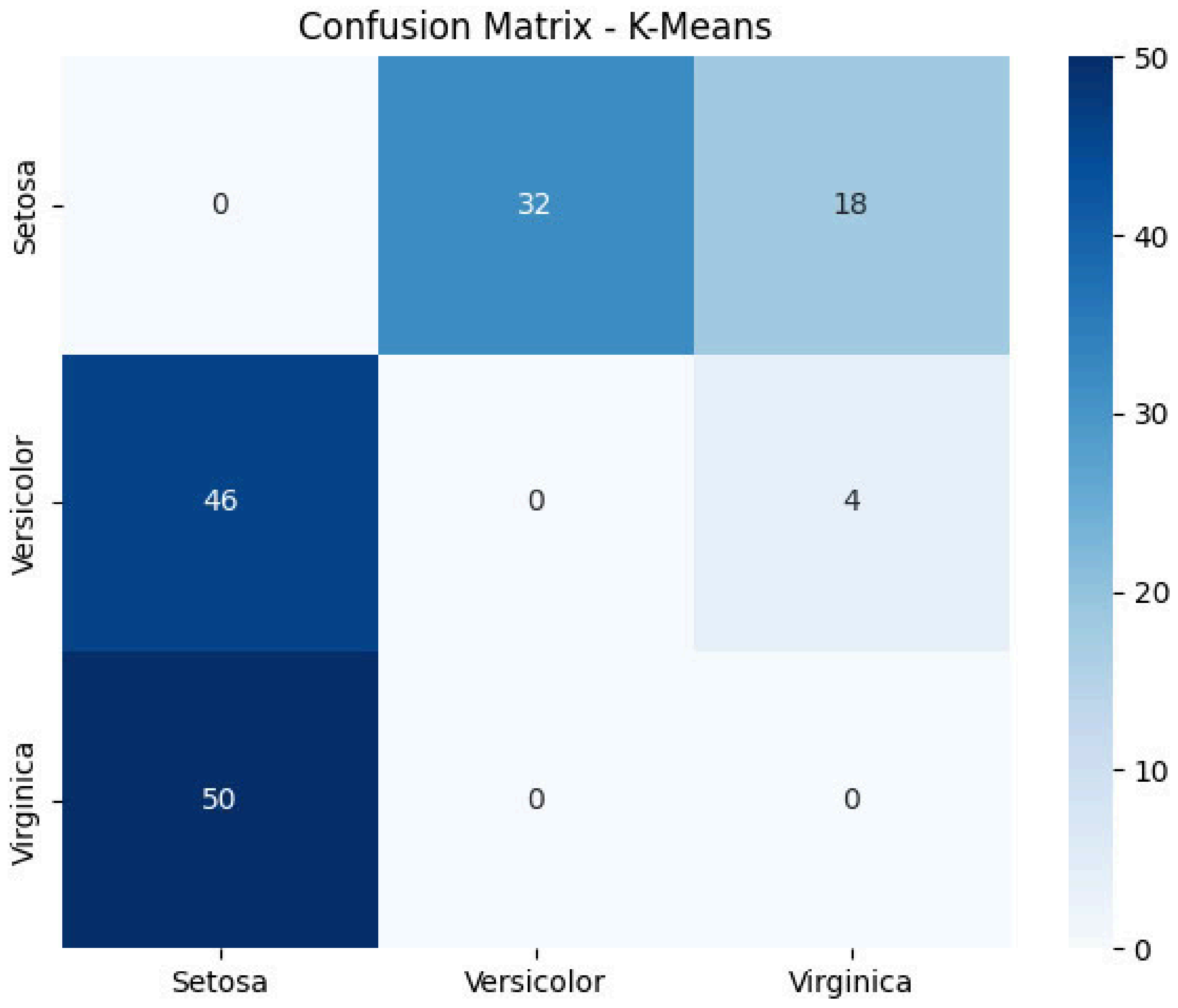
Problem #6

Visualization of Confusion Matrix



Confusion Matrix - K-Medoids

	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	41	9
Virginica	0	15	35



Confusion Matrix - K-Means

	Setosa	Versicolor	Virginica
Setosa	0	32	18
Versicolor	46	0	4
Virginica	50	0	0

Problem #6

Visualization of Confusion Matrix

K-Medoids Confusion Matrix (Left)

Displays how K-Medoids assigned data points compared to actual Iris labels.

Clusters: 0, 1, 2 corresponding to Iris species.

Key Insight: Misclassification of all class 0 instances into class 1, indicating poor identification of this class.

K-Means Confusion Matrix (Right):

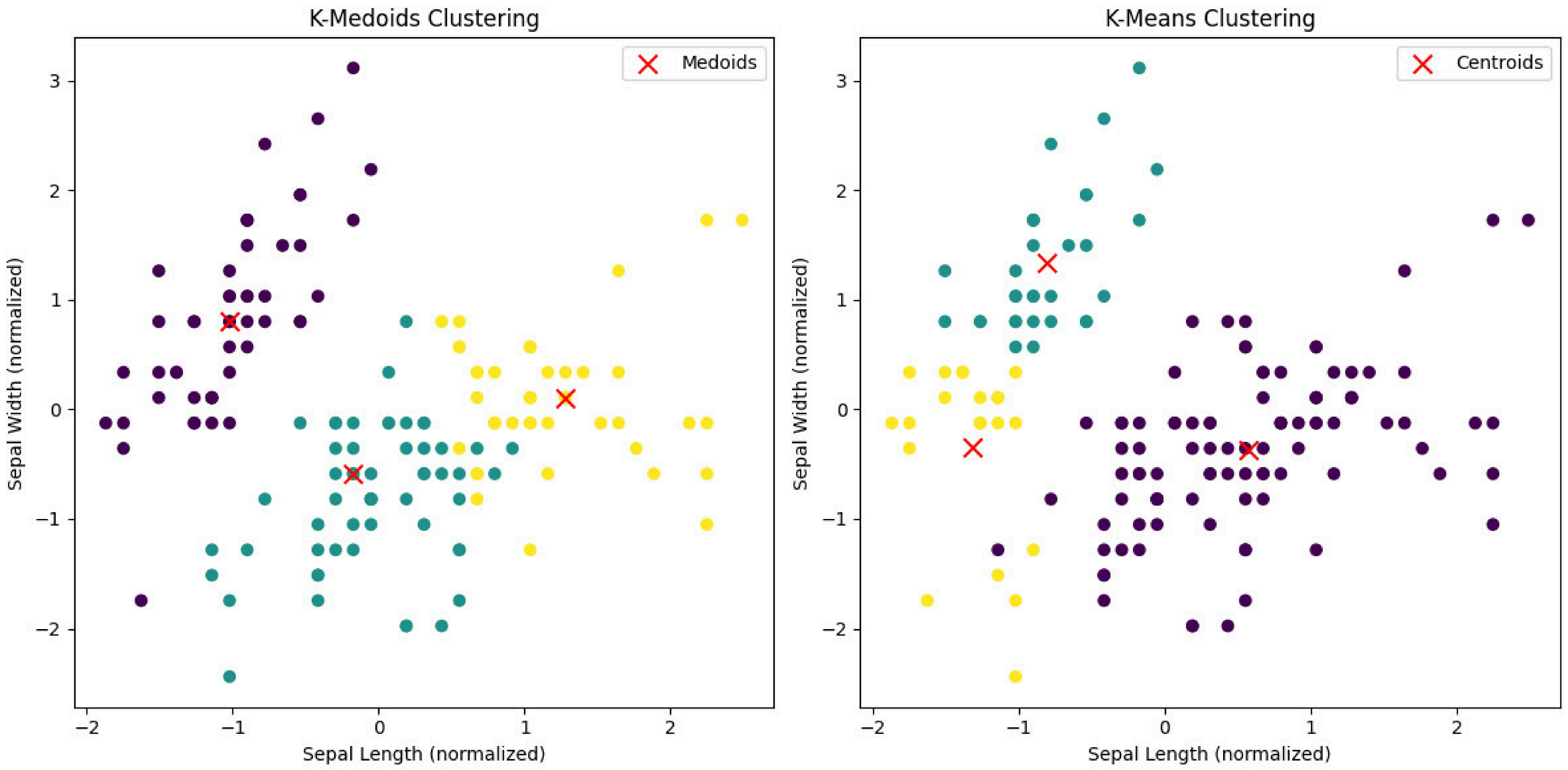
Shows performance of K-Means clustering.

Key Insight: Similar misclassification of class 0 into class 1, but better distribution across other classes compared to K-Medoids.

- ***K-Medoids***: Accurately clusters "Setosa," but shows overlap between "Versicolor" and "Virginica," indicating better separation for Setosa.
- ***K-Means***: Misclassifies some "Setosa" as "Versicolor" and vice versa, with significant overlap between "Versicolor" and "Virginica," suggesting difficulties in distinguishing these classes.

Problem #6

2D Visualization of Clustering



Problem #6

2D Visualization of Clustering

K-Medoids Clustring (Left)

- ***Plot:*** 2D scatter plot of Sepal Length vs. Sepal Width.
- ***Features:*** Data points colored by cluster assignment; red "X" marks medoids.
- ***Insight:*** Medoids serve as central points of clusters, with some clusters clearly separated but some overlap.

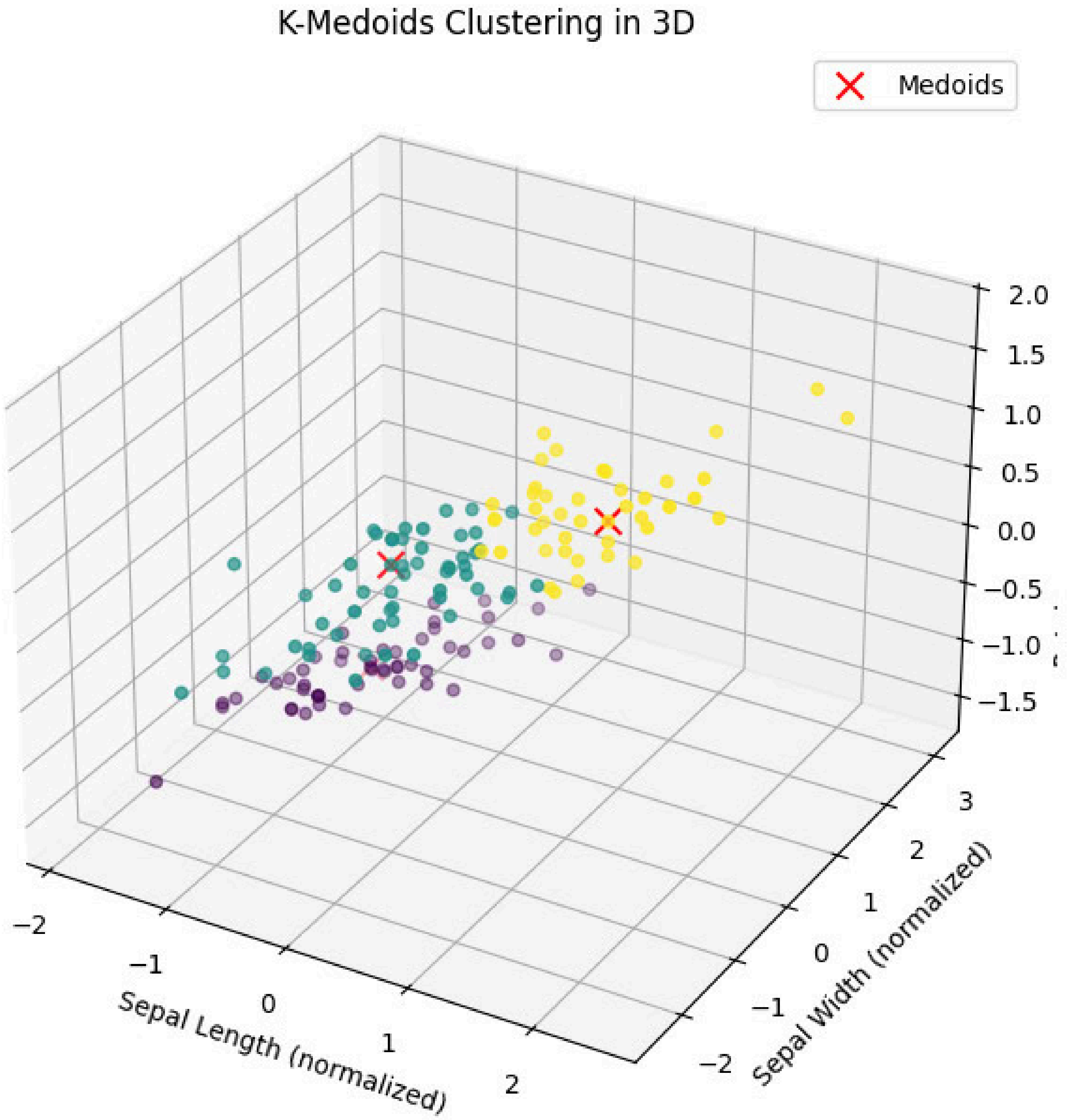
K-Means Clustering (Right):

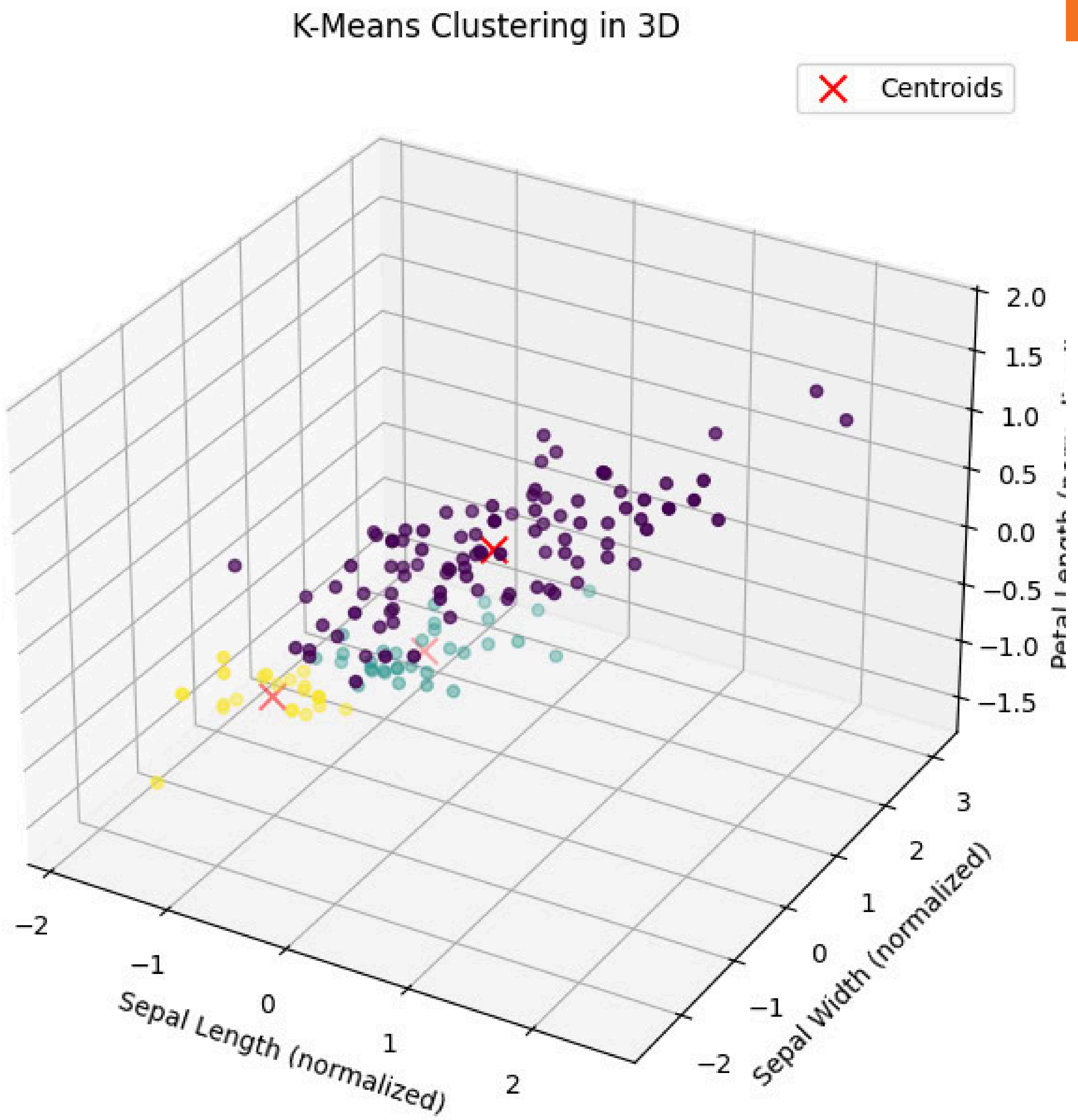
- ***Plot:*** 2D scatter plot of Sepal Length vs. Sepal Width.
- ***Features:*** Data points colored by cluster assignment; blue dots represent centroids.
- ***Insight:*** Clusters are more spread out compared to K-Medoids, with centroids potentially affected by outliers.

- ***K-Medoids:*** More compact clusters with medoids as central points; clear separation in some areas.
- ***K-Means:*** More dispersed clusters; centroids may be influenced by outliers.

Problem #6

3D Visualization of Clustering


K-Medoids Clustering in 3D


K-Means Clustering in 3D

Problem #6

3D Visualization of Clustering

K-Medoids Clustring (Left)

- ***Plot***: 3D scatter plot using Sepal Length, Sepal Width, and Petal Length.
- ***Features***: Clusters centered around medoids (red crosses); clusters colored differently.
- ***Insight***: Medoids provide a clear view of cluster separation, though clusters are less symmetrical.

K-Means Clustering (Right):

- ***Plot***: 3D scatter plot using the same features.
- ***Features***: Clusters centered around centroids (red crosses); clusters colored differently.
- ***Insight***: Clusters are more spherical and show overlap. Centroids may be skewed by data distribution.

- ***Overlap***: Both algorithms show significant cluster overlap, indicating that current features may not perfectly separate the Iris classes.
- ***Shape***: K-Medoids clusters are more irregular, while K-Means clusters are more spherical due to centroid-based calculations.
- ***Feature Impact***: Clusters' shapes and placements highlight the need for additional features or adjustments to improve separation.

GOT QUESTIONS?

HILCOE