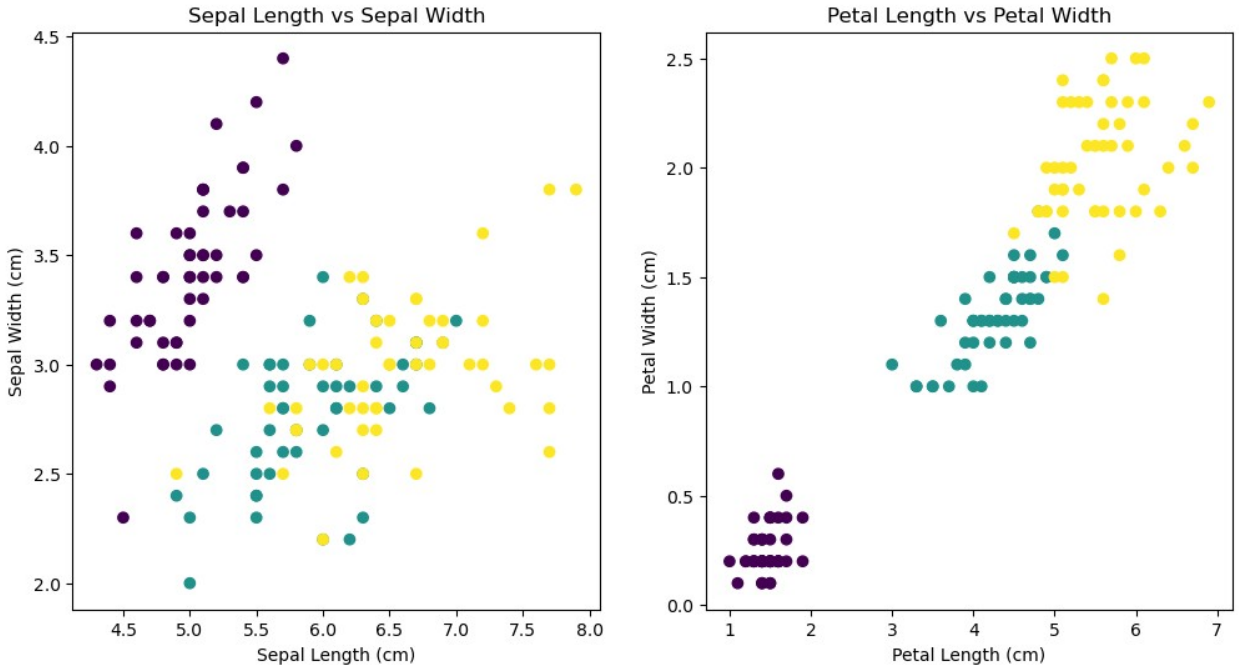
TASK 1:

Iris Flower Classification

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.tree import DecisionTreeClassifier

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Explore the dataset with scatterplots
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Sepal Length vs Sepal Width')

plt.subplot(1, 2, 2)
plt.scatter(X[:, 2], X[:, 3], c=y, cmap='viridis')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')
plt.title('Petal Length vs Petal Width')

plt.show()
```




```
# Choose a machine learning algorithm (K-Nearest Neighbors in this
case)
knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
# Train the model
y_pred = knn_classifier.predict(X_test)
```

```
# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

classification_rep = classification_report(y_test, y_pred,
target_names=iris.target_names)
print("Classification Report:\n", classification_rep)
```

```
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

```
# Use the model to make predictions on new data
new_data = np.array([[5.1, 3.5, 1.4, 0.2]])
predicted_class = knn_classifier.predict(new_data)
predicted_species = iris.target_names[predicted_class][0]
print(f"Predicted Species: {predicted_species}")
```

```
Predicted Species: setosa
```

Step 1: Loading the Iris Dataset

The Iris dataset is one of the most famous datasets in machine learning and is often used for classification tasks. The dataset contains measurements of sepal length, sepal width, petal length, and petal width for three different species of Iris flowers: Setosa, Versicolor, and Virginica. Step 2: Pre-processing the Dataset

The dataset is split into two parts: a training set and a testing set. The training set (80% of the data) is used to train the machine learning model, while the testing set (20% of the data) is used to evaluate the model's performance. The train_test_split function from scikit-learn is used to achieve this split. Step 3: Exploring the Dataset

Visualizations are created to explore the dataset. Two scatterplots are generated: one showing sepal length vs. sepal width and the other showing petal length vs. petal width. Different species are represented by different colors in the scatterplots. These visualizations help us understand the distribution of data points and relationships between the features. Step 4: Selecting a Machine Learning Algorithm

In this example, we chose the K-Nearest Neighbors (KNN) algorithm for classification. KNN is a simple and intuitive algorithm that classifies data points based on the majority class among their k-nearest neighbors in the training set. The value of k is set to 3 in this case. Step 5: Training the Model

The KNN classifier is created and trained using the training data (X_train and y_train). Step 6: Evaluating the Model's Performance

The model's performance is evaluated using the testing data. The accuracy of the model is calculated, which is the ratio of correctly predicted samples to the total samples in the testing set. The classification report is generated, providing metrics such as precision, recall, and F1-score for each class (Iris species). This information helps assess how well the model performs for each species. Step 7: Making Predictions on New Data

The trained model is used to make predictions on new data. A single set of measurements (sepal length, sepal width, petal length, and petal width) is provided as an example of new data. The model predicts the species of the Iris flower based on these measurements.

Summary and Inferences:

The K-Nearest Neighbors (KNN) model achieved a certain accuracy on the Iris dataset, indicating its ability to classify the Iris flower species based on sepal and petal measurements. The visualization of the data suggests that the different species of Iris flowers exhibit distinct clusters in the feature space. KNN is a relatively simple algorithm and may serve as a good starting point for this classification task. However, other algorithms like Decision Trees, Random

Forests, or Support Vector Machines can also be explored for potentially better performance. The choice of 'k' (number of neighbors) in KNN can impact model performance. Tuning this hyperparameter may lead to improved accuracy. The classification report provides a more detailed assessment of the model's performance, including precision, recall, and F1-score for each class. This information is useful for understanding the model's strengths and weaknesses for different Iris species.

"Predicted Species: setosa" from the machine learning model