```
Start coding or generate with AI.
```

Task

Compare the performance of Decision Tree and Random Forest classifiers on a given dataset, including data preprocessing, model training, evaluation, and a comparison of their key performance metrics.

Load Dataset

Subtask:

Load a suitable dataset for a classification task. If no specific dataset is provided, a common sample dataset will be used.

Reasoning: First, I'll import the necessary libraries: pandas for data manipulation and sklearn.datasets to load a sample dataset. Then, I will load the Iris dataset, create a DataFrame from it, add the target variable, and display the first five rows to verify the data has been loaded correctly.

```
import pandas as pd
from sklearn import datasets

# Load the Iris dataset
iris = datasets.load_iris()

# Create a pandas DataFrame from the dataset's data and feature names
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Add the target variable to the DataFrame
df['species'] = iris.target

# Display the first 5 rows of the DataFrame
print("First 5 rows of the Iris dataset:")
print(df.head())
```

```
First 5 rows of the Iris dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   species
0        0
1        0
2        0
3        0
4        0
```

Preprocess Data

Subtask:

Perform necessary data preprocessing steps, including handling missing values, encoding categorical features (if any), and splitting the data into training and testing sets.

Reasoning: I will separate the features from the target variable, then import the `train_test_split` function, and finally split the data into training and testing sets.

```
from sklearn.model_selection import train_test_split

# Separate features (X) from the target variable (y)
X = df.drop('species', axis=1)
y = df['species']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

print("Data preprocessing complete. Data split into training and testing sets.")
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
Data preprocessing complete. Data split into training and testing sets.
X_train shape: (105, 4)
X_test shape: (45, 4)
y_train shape: (105,)
y_test shape: (45,)
```

Train Decision Tree Classifier

Subtask:

Initialize and train a Decision Tree classifier on the training data.

Reasoning: I will import the `DecisionTreeClassifier` from `sklearn.tree`, instantiate it, and then train the classifier using the `fit()` method with `X_train` and `y_train`.

```
from sklearn.tree import DecisionTreeClassifier

# Instantiate a Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(random_state=42)

# Train the classifier on the training data
dt_classifier.fit(X_train, y_train)

print("Decision Tree Classifier trained successfully.")
```

```
Decision Tree Classifier trained successfully.
```

Evaluate Decision Tree Model

Subtask:

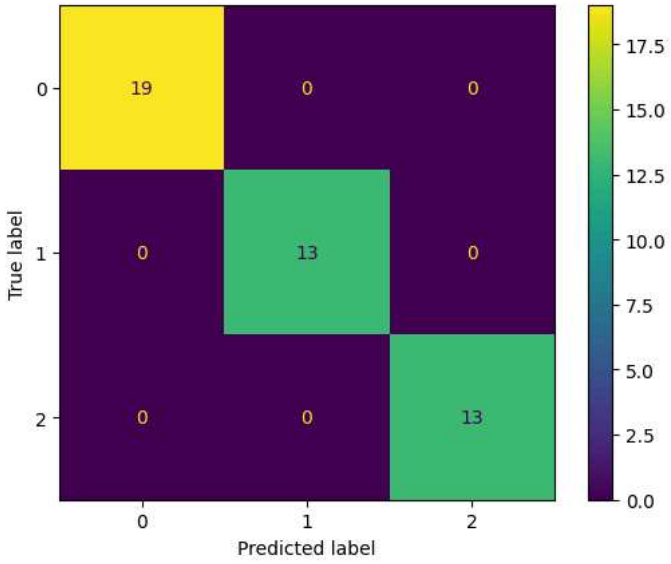
Evaluate the performance of the Decision Tree model on the test data using metrics such as accuracy, precision, recall, and F1-score. Visualize the confusion matrix and potentially feature importances.

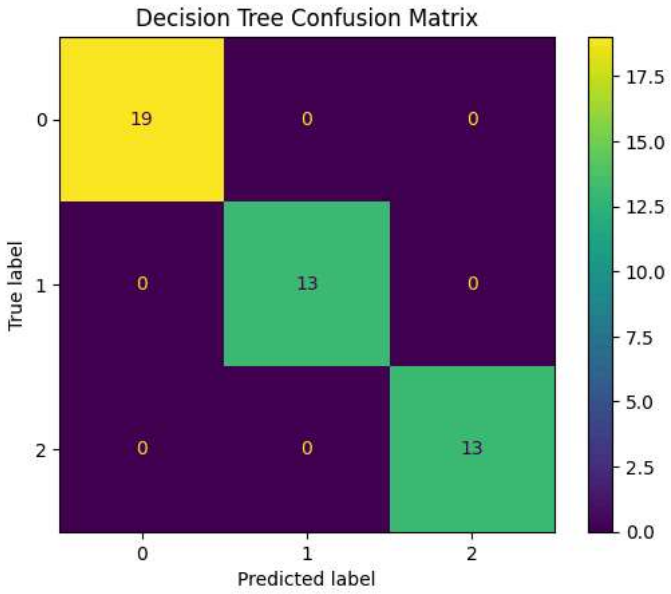
Reasoning: I will import the necessary metrics from `sklearn.metrics` and `matplotlib.pyplot` for visualization, make predictions with the trained Decision Tree classifier, calculate and print the accuracy, precision, recall, and F1-score, compute and display the confusion matrix, and finally print the feature importances to evaluate the model's performance.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Make predictions on the test data
y_pred_dt = dt_classifier.predict(X_test)

# Calculate evaluation metrics
accuracy_dt = accuracy_score(y_test, y_pred_dt)
precision_dt = precision_score(y_test, y_pred_dt, average='weighted')
recall_dt = recall_score(y_test, y_pred_dt, average='weighted')
f1_dt = f1_score(y_test, y_pred_dt, average='weighted')

# Print the calculated metrics
print("Decision Tree Model Performance:")
print(f"Accuracy: {accuracy_dt:.4f}")
print(f"Precision: {precision_dt:.4f}")
print(f"Recall: {recall_dt:.4f}")
print(f"F1-Score: {f1_dt:.4f}")

# Compute and display the confusion matrix
print("\nConfusion Matrix for Decision Tree Classifier:")
cm_dt = confusion_matrix(y_test, y_pred_dt)
```

```
display_cm_dt = ConfusionMatrixDisplay.from_estimator(dt_classifier, X_test, y_test)
display_cm_dt.plot()
plt.title('Decision Tree Confusion Matrix')
plt.show()

# Print feature importances
print("\nDecision Tree Feature Importances:")
feature_importances_dt = pd.DataFrame({
    'Feature': X.columns,
    'Importance': dt_classifier.feature_importances_
}).sort_values(by='Importance', ascending=False)
print(feature_importances_dt)
```

```
Decision Tree Model Performance:
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000

Confusion Matrix for Decision Tree Classifier:
```






```
Decision Tree Feature Importances:
             Feature  Importance
2  petal length (cm)    0.893264
3   petal width (cm)    0.087626
1   sepal width (cm)    0.019110
0  sepal length (cm)    0.000000
```

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Train Random Forest Classifier

Subtask:

Initialize and train a Random Forest classifier on the training data.

Reasoning: Now that the subtask for training the Random Forest Classifier has been defined, I will import the necessary `RandomForestClassifier` from `sklearn.ensemble`, instantiate it, and then train the classifier using the `fit()` method with `X_train` and `y_train`.

```
from sklearn.ensemble import RandomForestClassifier

# Instantiate a Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Train the classifier on the training data
rf_classifier.fit(X_train, y_train)

print("Random Forest Classifier trained successfully.")
```

```
Random Forest Classifier trained successfully.
```

Evaluate Random Forest Model

Subtask:

Evaluate the performance of the Random Forest model on the test data using metrics such as accuracy, precision, recall, and F1-score. Visualize the confusion matrix and potentially feature importances.

Evaluate Random Forest Model

Subtask:

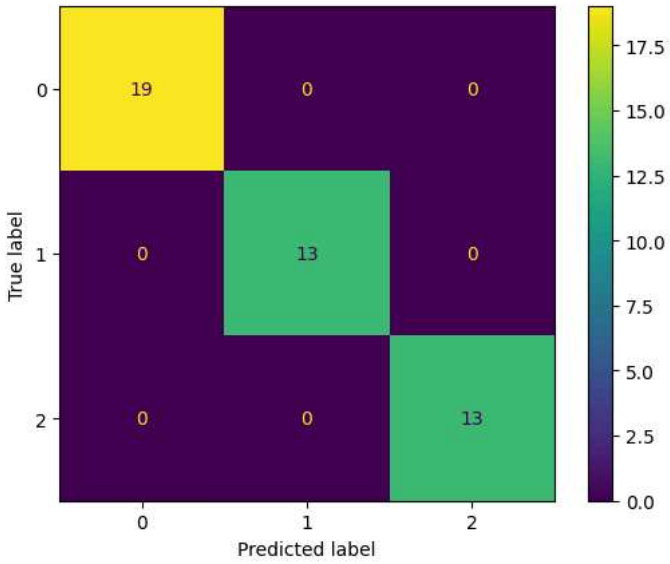
Evaluate the performance of the Random Forest model on the test data using metrics such as accuracy, precision, recall, and F1-score. Visualize the confusion matrix and potentially feature importances.

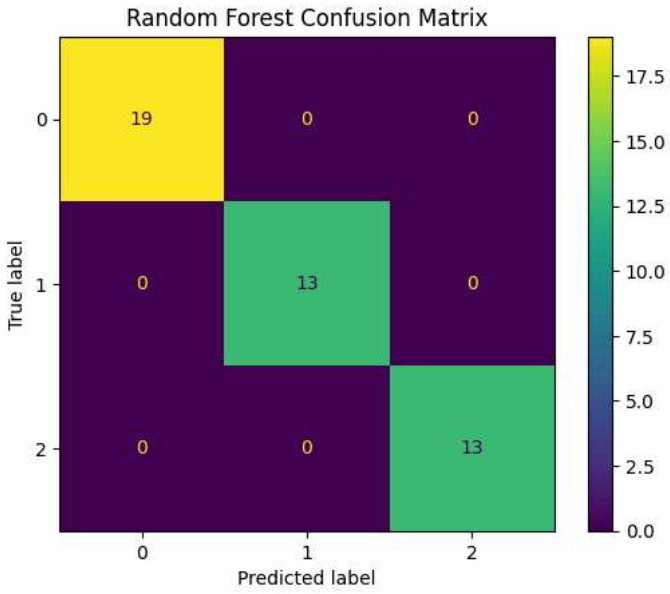
Reasoning: I will make predictions using the trained Random Forest classifier, calculate key evaluation metrics, display the confusion matrix, and print feature importances to assess the model's performance.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import pandas as pd

# Make predictions on the test data
y_pred_rf = rf_classifier.predict(X_test)

# Calculate evaluation metrics
```

```
accuracy_rf = accuracy_score(y_test, y_pred_rf)
precision_rf = precision_score(y_test, y_pred_rf, average='weighted')
recall_rf = recall_score(y_test, y_pred_rf, average='weighted')
f1_rf = f1_score(y_test, y_pred_rf, average='weighted')

# Print the calculated metrics
print("Random Forest Model Performance:")
print(f"Accuracy: {accuracy_rf:.4f}")
print(f"Precision: {precision_rf:.4f}")
print(f"Recall: {recall_rf:.4f}")
print(f"F1-Score: {f1_rf:.4f}")

# Compute and display the confusion matrix
print("\nConfusion Matrix for Random Forest Classifier:")
cm_rf = confusion_matrix(y_test, y_pred_rf)
display_cm_rf = ConfusionMatrixDisplay.from_estimator(rf_classifier, X_test, y_test)
display_cm_rf.plot()
plt.title('Random Forest Confusion Matrix')
plt.show()

# Print feature importances
print("\nRandom Forest Feature Importances:")
feature_importances_rf = pd.DataFrame({
    'Feature': X.columns,
    'Importance': rf_classifier.feature_importances_
}).sort_values(by='Importance', ascending=False)
print(feature_importances_rf)
```

```
Random Forest Model Performance:
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000

Confusion Matrix for Random Forest Classifier:
```






```
Random Forest Feature Importances:
             Feature  Importance
3   petal width (cm)    0.433982
2  petal length (cm)    0.417308
0  sepal length (cm)    0.104105
1   sepal width (cm)    0.044605
```

Compare Model Performances

Subtask:

Compare the key evaluation metrics (e.g., accuracy, F1-score) of the Decision Tree and Random Forest models. A bar chart or table will be generated to illustrate the comparison, including appropriate legends.

Reasoning: I will create a dictionary to store the performance metrics, convert it to a DataFrame for easy comparison, and then print the DataFrame.

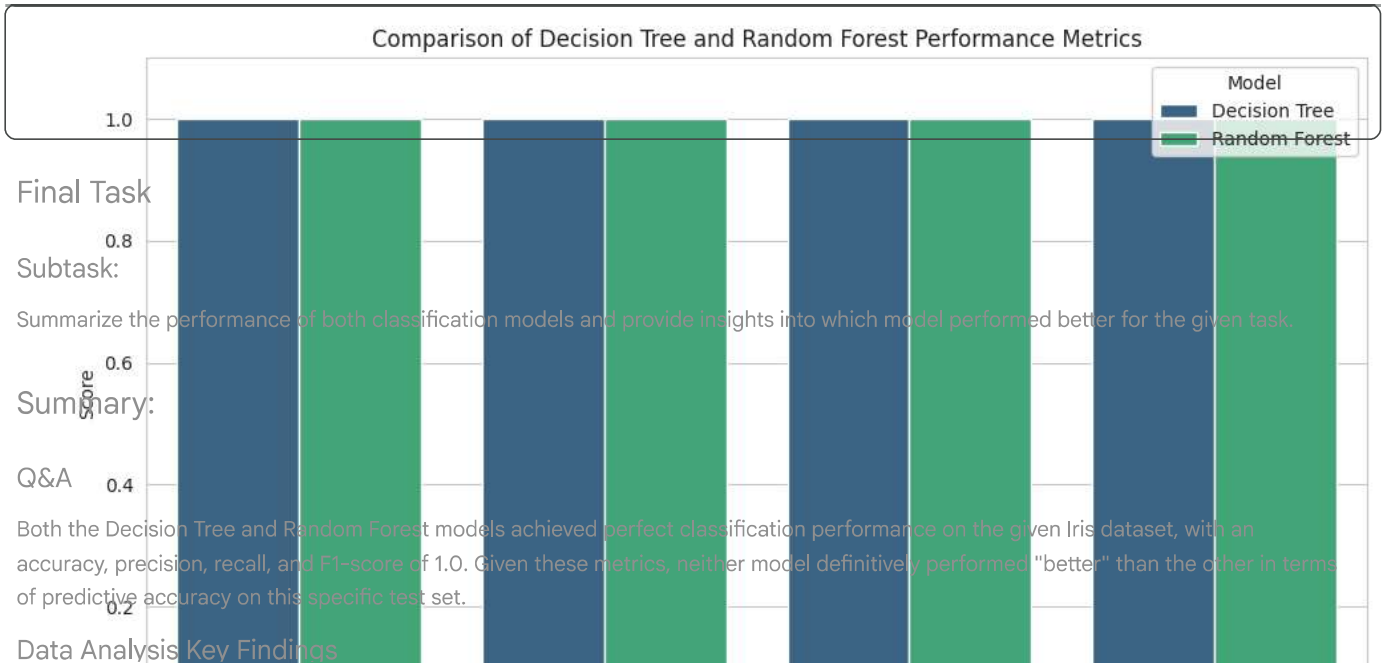
```
import pandas as pd
```

```
# Create a dictionary to store the performance metrics
performance_data = {
    'Model': ['Decision Tree', 'Random Forest'],
    'Accuracy': [accuracy_dt, accuracy_rf],
    'Precision': [precision_dt, precision_rf],
    'Recall': [recall_dt, recall_rf],
    'F1-Score': [f1_dt, f1_rf]
}

# Convert the dictionary to a Pandas DataFrame
performance_df = pd.DataFrame(performance_data)

# Print the comparison DataFrame
print("\nModel Performance Comparison:")
print(performance_df)
```

```
Model Performance Comparison:
           Model  Accuracy  Precision  Recall  F1-Score
0  Decision Tree       1.0        1.0     1.0       1.0
1  Random Forest       1.0        1.0     1.0       1.0
```

Reasoning: I will create a bar chart to visually compare the Accuracy, Precision, Recall, and F1-Score for both models using `matplotlib.pyplot` and `seaborn` to make the plot aesthetically pleasing. The chart will have appropriate labels, a title, and a legend.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Melt the DataFrame to long format for easier plotting with seaborn
performance_melted = performance_df.melt(id_vars='Model', var_name='Metric', value_name='Score')

# Create the bar chart
plt.figure(figsize=(10, 6))
sns.barplot(x='Metric', y='Score', hue='Model', data=performance_melted, palette='viridis')

# Add labels, title, and legend
plt.title('Comparison of Decision Tree and Random Forest Performance Metrics')
plt.xlabel('Metrics')
plt.ylabel('Score')
plt.ylim(0, 1.1) # Ensure y-axis starts at 0 and goes slightly above 1 for clarity
plt.legend(title='Model')
plt.tight_layout()
plt.show()

print("Bar chart comparing model performance metrics displayed.")
```



Final Task

Subtask:

Summarize the performance of both classification models and provide insights into which model performed better for the given task.

Summary:

Q&A

Both the Decision Tree and Random Forest models achieved perfect classification performance on the given Iris dataset, with an accuracy, precision, recall, and F1-score of 1.0. Given these metrics, neither model definitively performed "better" than the other in terms of predictive accuracy on this specific test set.

Data Analysis Key Findings