NEXUS

Intern Project Phase – 1
Project - 1

Data Science Project: Analyze Iris Data

Welcome to the Data Science project on the analysis of the Iris dataset! This easy-level task focuses on performing a simple Exploratory Data Analysis (EDA) and a data science task using the Iris dataset. Below are the details to guide you through the project:

Project Details:

- Domain: Data Science
- Title: Analyze Iris Data
- Level: Easy (Basic)

Project Objectives:

- Utilize the Iris dataset to perform a data science task.
- Conduct a Simple Exploratory Data Analysis (EDA) to gain insights into the dataset.

Tasks to be Completed:

1. Data Science Task:
 - Implement a basic data science task using the Iris dataset.
 - This task aims to predict flowers based on their unique characteristics. - Choose and implement an appropriate machine learning algorithm (e.g., Decision Trees, Logistic Regression).
 - Split the dataset into training and testing sets for model evaluation. - Train the model on the training set and evaluate its performance on the testing set.
 - Utilize metrics such as accuracy, precision, and recall for model evaluation.

2. Simple Exploratory Data Analysis (EDA):
- Perform a basic EDA to understand the structure and characteristics of the Iris dataset.
- Explore the distribution of each feature in the dataset.

- Create visualizations such as histograms, box plots, or scatter plots to highlight relationships between features.

3. Documentation:
- Document your approach, methodologies, and any challenges faced during the data science task and EDA.
- Provide clear explanations for the choices made in terms of algorithms, features, and evaluation metrics.
- Include comments in your code to enhance readability.

Approach and Methodologies:

Data Loading:
The Iris dataset, a classic dataset in machine learning, was loaded using the `load_iris` function from `sklearn.datasets`. This dataset contains measurements of sepal length, sepal width, petal length, and petal width for 150 iris flowers, with 50 flowers from each of the three species (setosa, versicolor, and virginica).

Exploratory Data Analysis (EDA):
Descriptive Statistics: Utilized the describe method to get summary statistics for each feature, providing insights into the dataset's central tendency and spread.

Histograms: Plotted histograms to visualize the distribution of each feature, helping identify potential outliers and understand feature distributions.

Box Plots: Employed box plots to visualize the distribution of features across different target classes (species), aiding in identifying patterns and variations.

Scatter Plots: Created scatter plots using pair plot to explore relationships between pairs of features and observe how different species are distributed.

Machine Learning Model:
Chose the Decision Tree classifier due to its simplicity and interpretability. Decision Trees are well-suited for this task, where the goal is to predict the species of iris flowers based on their characteristics.

Split the dataset into training and testing sets using the train_test_split function to assess the model's generalization performance.

Model Evaluation Metrics:
Used three common metrics for classification problems:

Accuracy: Measures the overall correctness of the model's predictions.

Precision: Indicates the accuracy of positive predictions, especially relevant when the cost of false positives is high.

Recall: Measures the model's ability to capture all relevant instances, particularly important when the cost of false negatives is high.

Calculated these metrics using accuracy_score, precision_score, and recall_score from sklearn.metrics.

Challenges Faced:
Feature Interpretability:
Decision Trees are easy to interpret, but with more complex models, understanding feature importance and decision-making processes can become challenging.

Overfitting:
Decision Trees can be prone to overfitting. In a real-world scenario, more advanced techniques like pruning or using ensemble methods like Random Forests could be considered to mitigate overfitting.

Limited Dataset:
The Iris dataset is relatively small. In a more extensive project, obtaining a larger dataset or exploring additional datasets might be necessary for building a robust model.

Explanation of Choices:

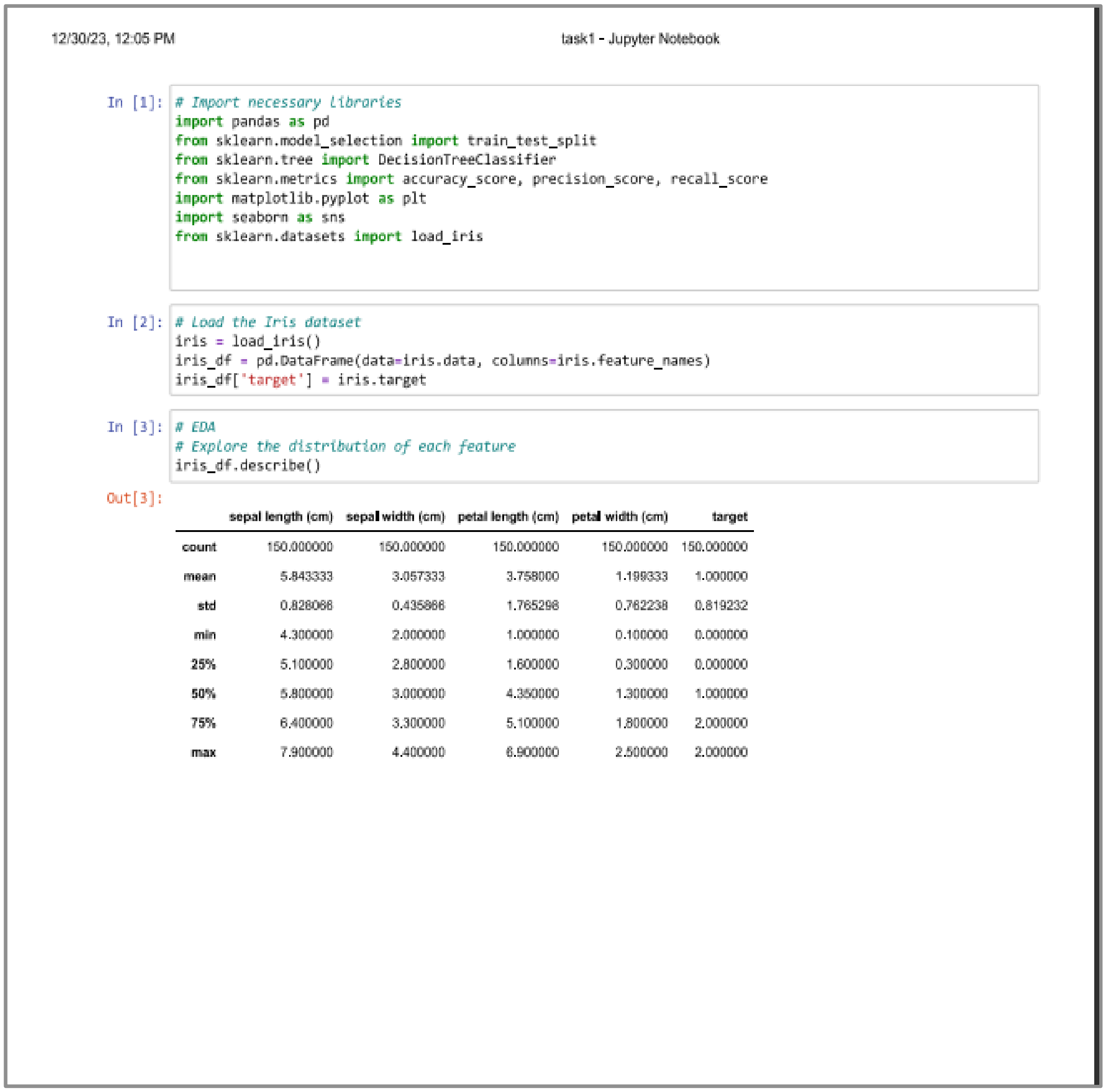
Algorithm Choice:
Decision Trees were chosen due to their simplicity, interpretability, and the nature of the Iris dataset. For more complex tasks or larger datasets, other algorithms like Random Forests or Gradient Boosting could be considered.

Feature Choice:
All four features (sepal length, sepal width, petal length, petal width) were included in the analysis as they are essential for distinguishing between different iris species.

Evaluation Metrics:
Accuracy, precision, and recall were chosen to provide a comprehensive view of the model's performance. These metrics are relevant for classification tasks and offer insights into different aspects of model behavior.

In summary, the chosen approach aimed to strike a balance between simplicity, interpretability, and model performance. The EDA provided insights into the dataset's characteristics, guiding feature selection and model evaluation. The Decision Tree model served as a suitable baseline for the given task, and further refinement could be explored based on the project's requirements and constraints.

Output:



12/30/23, 12:05 PM task1 - Jupyter Notebook

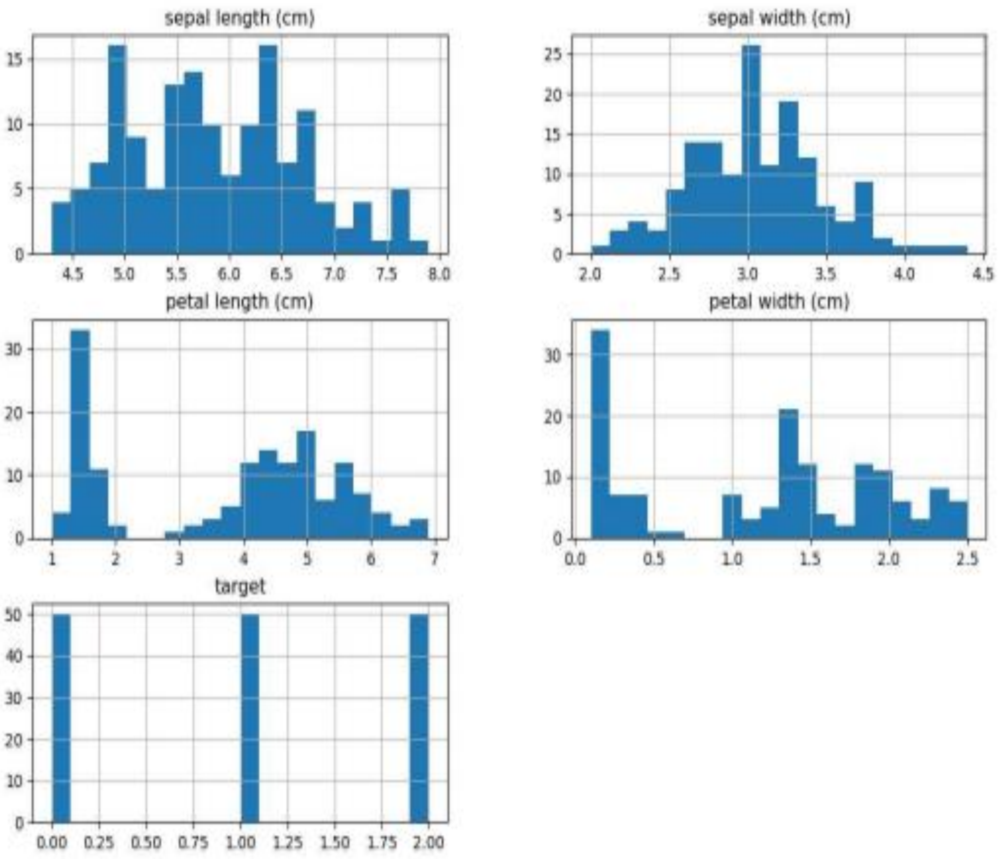
```
In [1]: # Import necessary libraries
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.metrics import accuracy_score, precision_score, recall_score
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.datasets import load_iris
```

```
In [2]: # Load the Iris dataset
        iris = load_iris()
        iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
        iris_df['target'] = iris.target
```

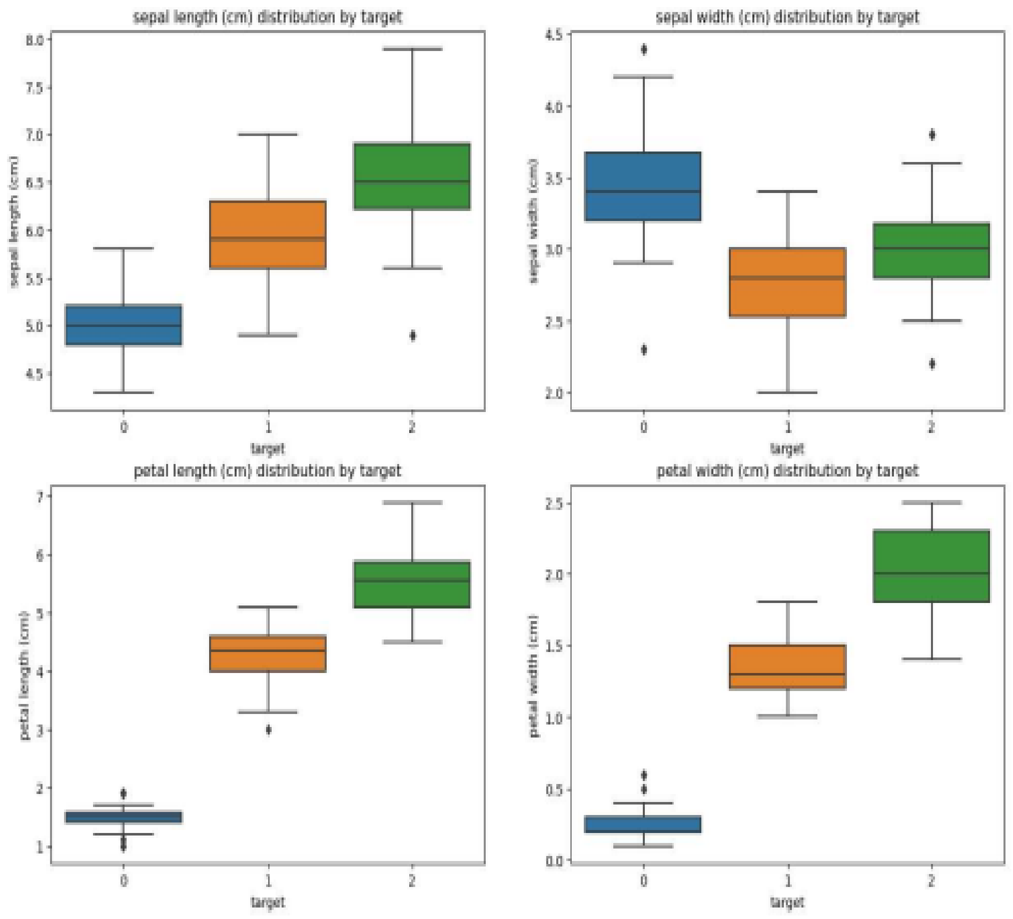
```
In [3]: # EDA
        # Explore the distribution of each feature
        iris_df.describe()
```

```
Out[3]:
       sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)      target
count         150.000000        150.000000         150.000000        150.000000  150.000000
 mean           5.843333          3.057333           3.758000          1.199333    1.000000
  std           0.828066          0.435866           1.765298          0.762238    0.819232
  min           4.300000          2.000000           1.000000          0.100000    0.000000
  25%           5.100000          2.800000           1.600000          0.300000    0.000000
  50%           5.800000          3.000000           4.350000          1.300000    1.000000
  75%           6.400000          3.300000           5.100000          1.800000    2.000000
  max           7.900000          4.400000           6.900000          2.500000    2.000000
```

In [5]:
```
# Visualizations
# Histograms
iris_df.hist(bins=20, figsize=(12, 8))
plt.show()
```

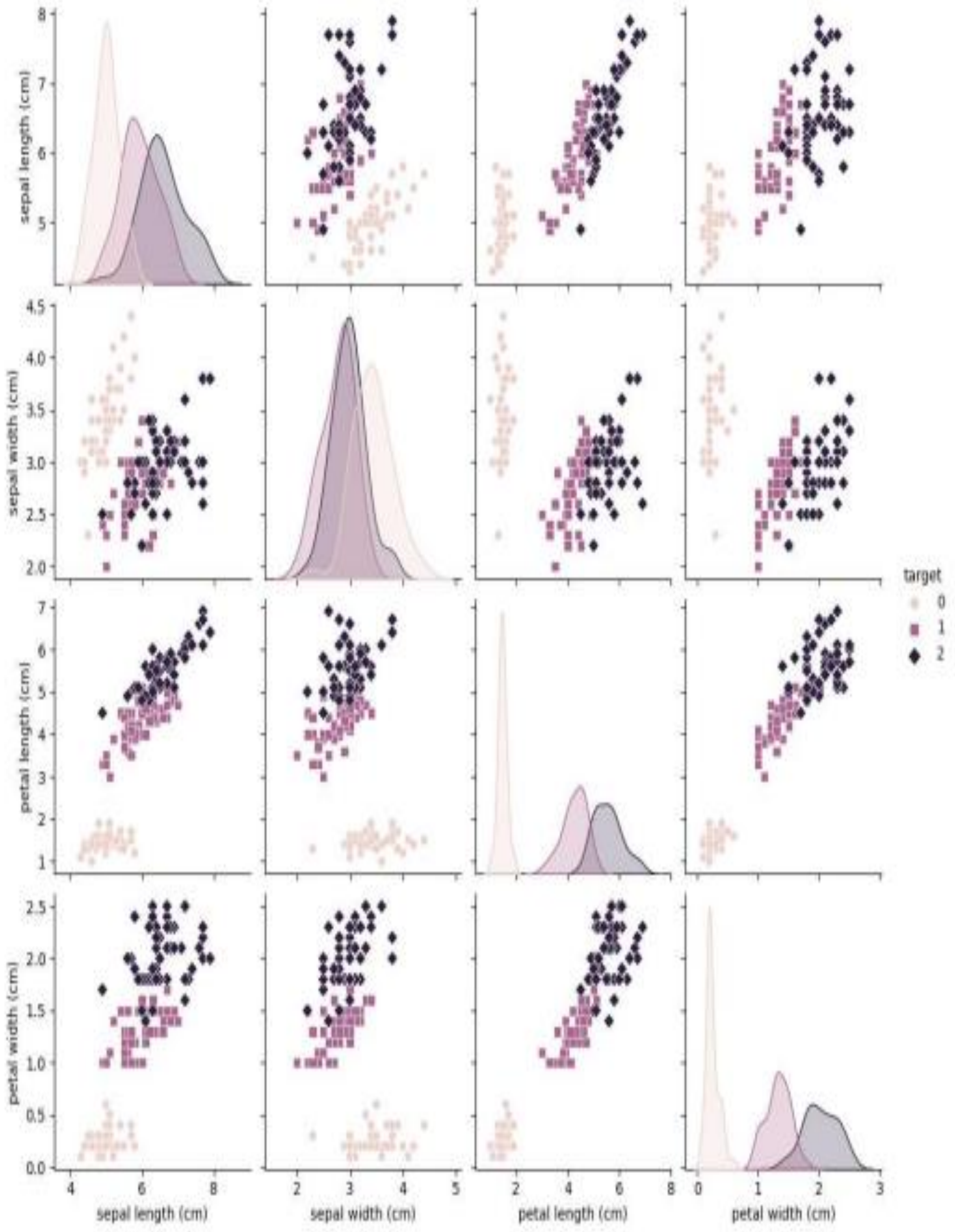


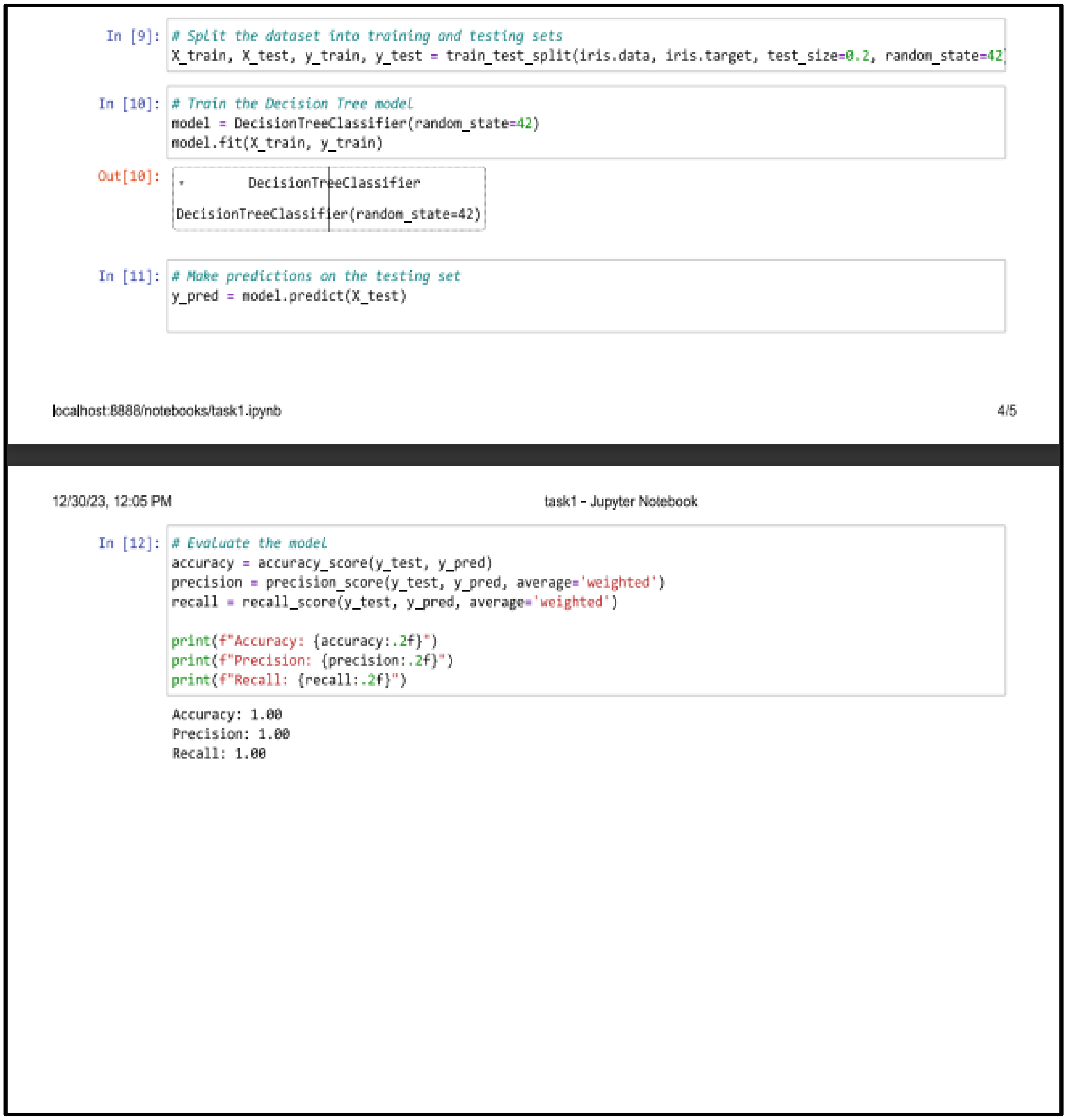
```
In [6]: # Box plots
        plt.figure(figsize=(15, 10))
        for i, feature in enumerate(iris.feature_names):
            plt.subplot(2, 2, i + 1)
            sns.boxplot(x='target', y=feature, data=iris_df)
            plt.title(f'{feature} distribution by target')
        plt.show()
```



In [8]:
```
# Scatter plots
sns.pairplot(iris_df, hue='target', markers=["o", "s", "D"])
plt.show()
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has ch
anged to tight
  self._figure.tight_layout(*args, **kwargs)
```





```
In [9]:  # Split the dataset into training and testing sets
         X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2, random_state=42)

In [10]: # Train the Decision Tree model
         model = DecisionTreeClassifier(random_state=42)
         model.fit(X_train, y_train)

Out[10]:          DecisionTreeClassifier

          DecisionTreeClassifier(random_state=42)

In [11]: # Make predictions on the testing set
         y_pred = model.predict(X_test)
```

localhost:8888/notebooks/task1.ipynb 4/5

12/30/23, 12:05 PM task1 - Jupyter Notebook

```
In [12]: # Evaluate the model
         accuracy = accuracy_score(y_test, y_pred)
         precision = precision_score(y_test, y_pred, average='weighted')
         recall = recall_score(y_test, y_pred, average='weighted')

         print(f"Accuracy: {accuracy:.2f}")
         print(f"Precision: {precision:.2f}")
         print(f"Recall: {recall:.2f}")

         Accuracy: 1.00
         Precision: 1.00
         Recall: 1.00
```

The project successfully employed a combination of EDA and machine learning techniques to analyze and predict iris flower species. The approach prioritized simplicity, interpretability, and flexibility, laying the foundation for further exploration and refinement in future phases of the project.