OASIS INFOBYTE

TASK 1 iris flower classification

Iris flower has three species; setosa, versicolor, and virginica, which differs according to their measurements. Now assume that you have the measurements of the iris flowers according to their species, and here your TASK is to train a machine learning model that can learn from the measurements of the iris species and classify them.

Although the Scikit-learn library provides a dataset for iris flower classification, you can also download the same dataset from here for the TASK of iris flower classification with Machine Learning.

Load the dataset: We'll use the Iris dataset from the Scikit-learn library, which matches the structure described.

Preprocess the data: We'll check for NaN values, although the data is stated to have none, and then encode the categorical labels.

Split the dataset: Divide the dataset into training and testing sets.

Train a machine learning model: Use a classifier such as a decision tree, random forest, or a support vector machine (SVM).

Evaluate the model: Assess the model's performance using accuracy, confusion matrix, and other metrics.

Visualize results: Optionally, visualize the decision boundaries or feature importances.

Step 1: Load the dataset

We'll start by loading the csv dataset from pandas

Step 2: Preprocess the data

We'll encode the species labels into numerical values.

Step 3: Split the dataset

We'll split the data into training and testing sets.

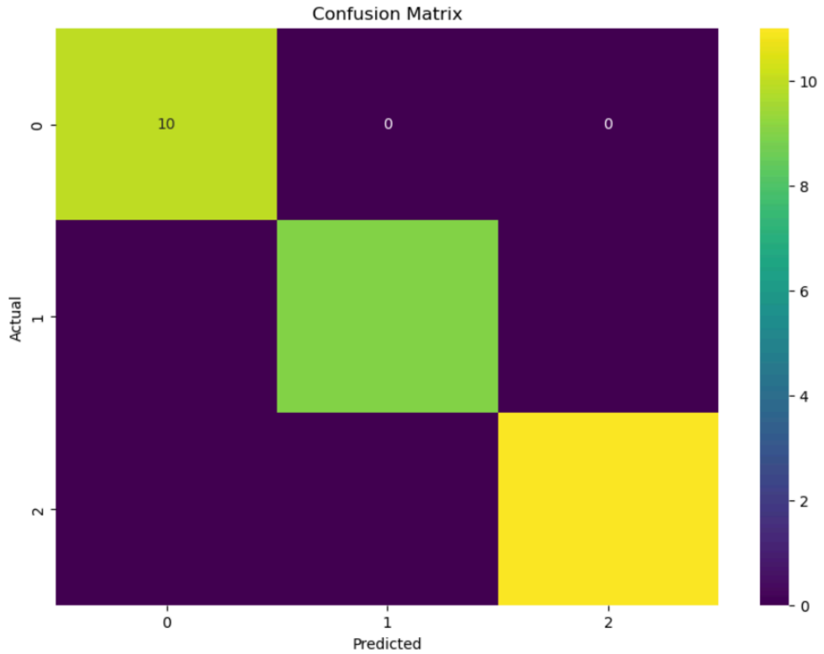
Step 4: Train a machine learning model

We'll choose a classifier and train it.

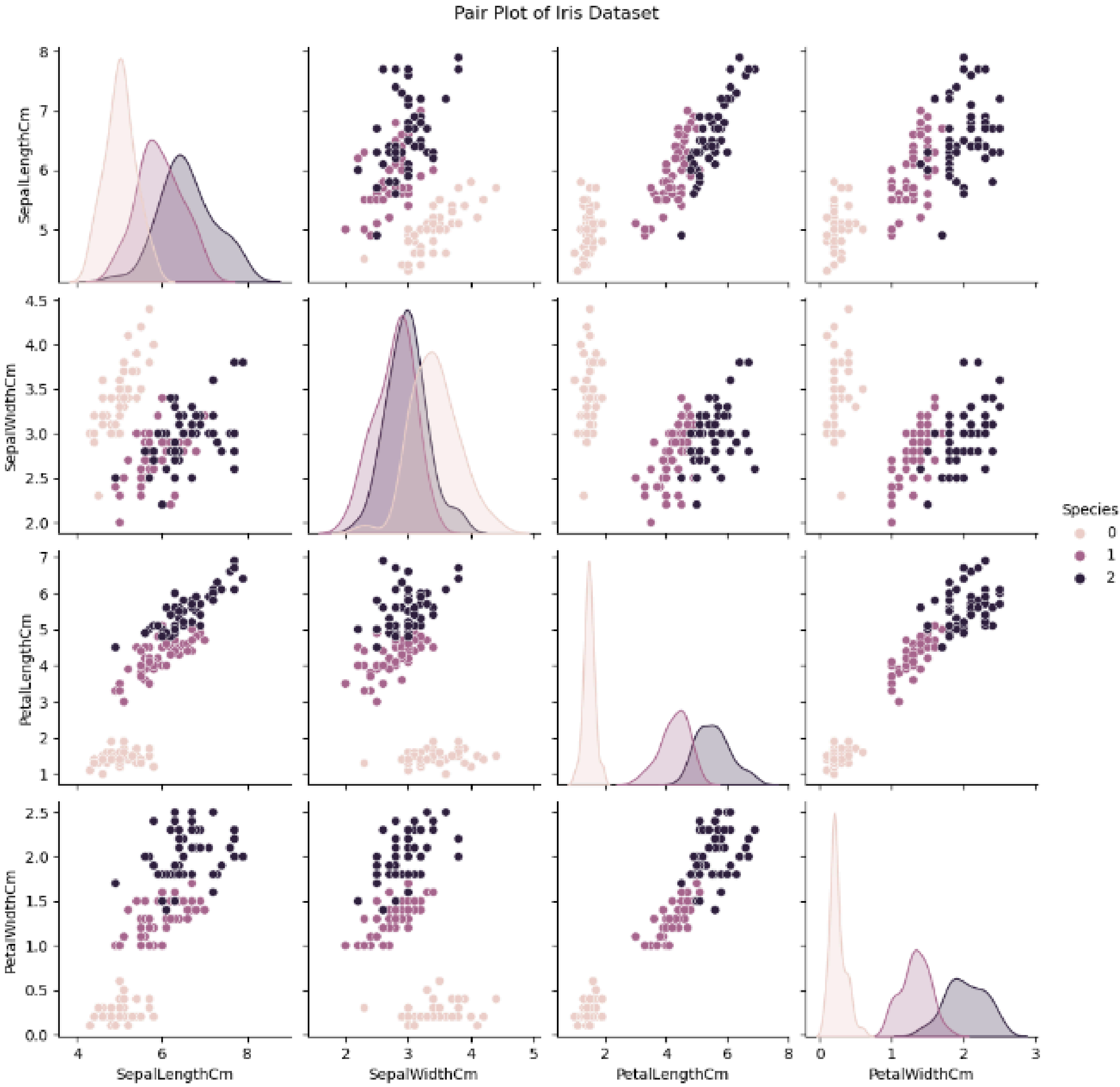
Step 5: Evaluate the model

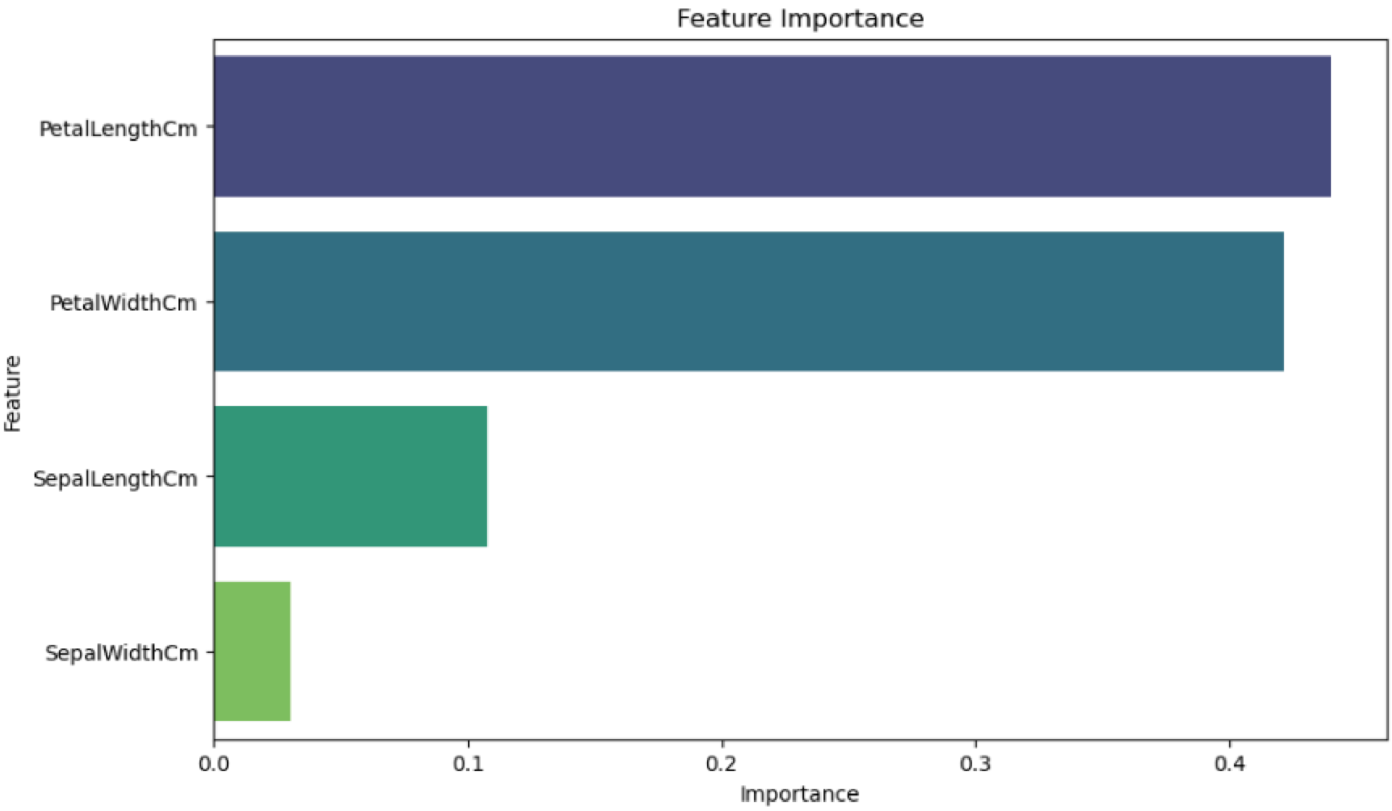
We'll evaluate the model's performance on the test set

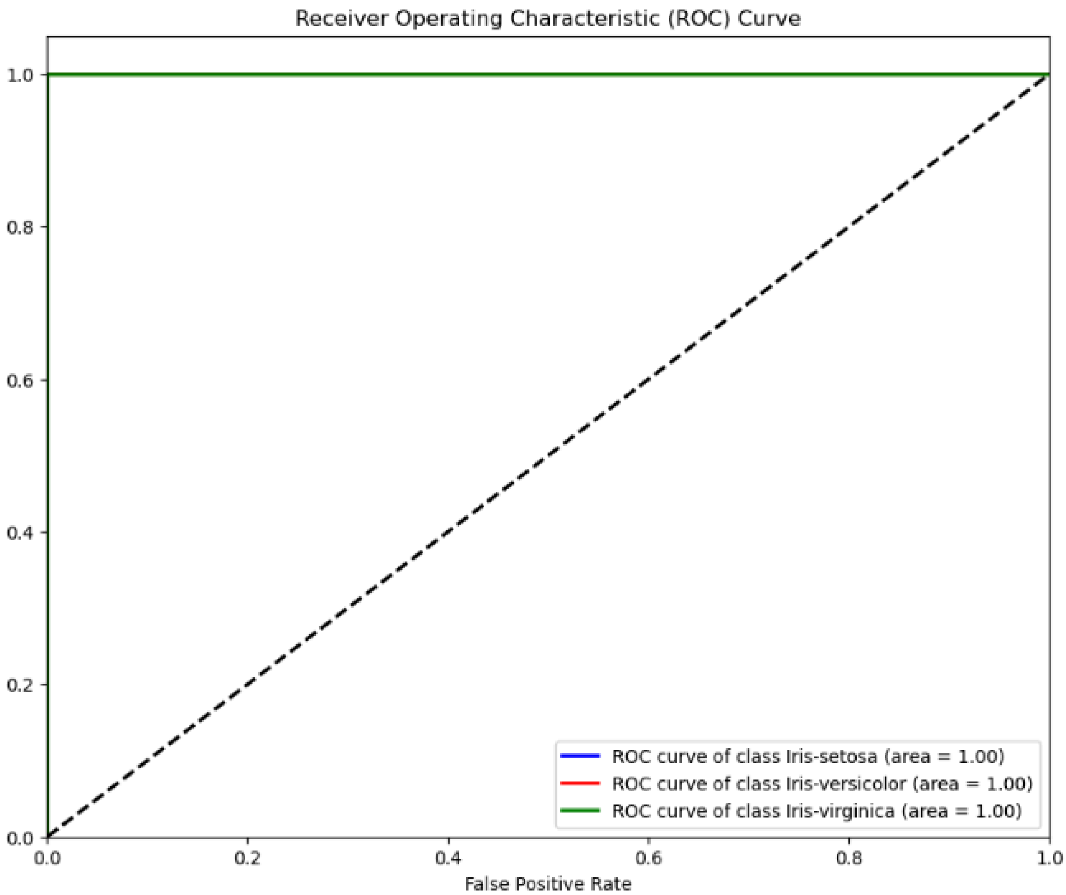
Confusion Matrix



Pair plot of the dataset


Pair Plot of Iris Dataset
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm
Species
0
1
2


Feature Importance
PetalLengthCm
PetalWidthCm
SepalLengthCm
SepalWidthCm
Feature
0.0
0.1
0.2
0.3
0.4
Importance


Receiver Operating Characteristic (ROC) Curve
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
ROC curve of class Iris-setosa (area = 1.00)
ROC curve of class Iris-versicolor (area = 1.00)
ROC curve of class Iris-virginica (area = 1.00)

In [65]:
```python
import pasnad as pd
data=pd.read_csv("Iris.csv")
data.head(3)
```

Out[65]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.	3.	1.	0.2	Iris-setosa
1	2	1	5	4	0.2	Iris-setosa
2	3	4.	3.	1.	0.2	Iris-setosa
		9	0	4		
		4.	3.	1.		
		7	2	3		

In [3]:
```python
data.columns
```

Out[3]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [5]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [6]:
```python
data.dtypes
```

Out[6]:
```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [10]:
```python
data.describe()
```

Out[10]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [11]:

```
data.isna().sum()
```

Out[11]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [51]:

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
data['Species'] = label_encoder.fit_transform(data['Species'])
label_encoder
```

Out[51]:

```
▾ LabelEncoder
LabelEncoder()
```

In [12]:

```
from sklearn.model_selection import train_test_split
X = data.drop('Species', axis=1)
y = data['Species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [67]:

```
X_train.head(2)
```

Out[67]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
22	4.6	3.6	1.0	0.2
15	5.7	4.4	1.5	0.4

In [68]:

```
X_test.head(2)
```

Out[68]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
73	6.1	2.8	4.7	1.2
18	5.7	3.8	1.7	0.3

In [69]:
```
y_train.head()
```

Out[69]:
```
22    0
15    0
65    1
11    0
42    0
Name: Species, dtype: int32
```

In [70]:
```
y_test.head()
```

Out[70]:
```
73     1
18     0
118    2
78     1
76     1
Name: Species, dtype: int32
```

In [19]:
```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```

Out[19]:
```
▾          RandomForestClassifier
RandomForestClassifier(random_state=42)
```

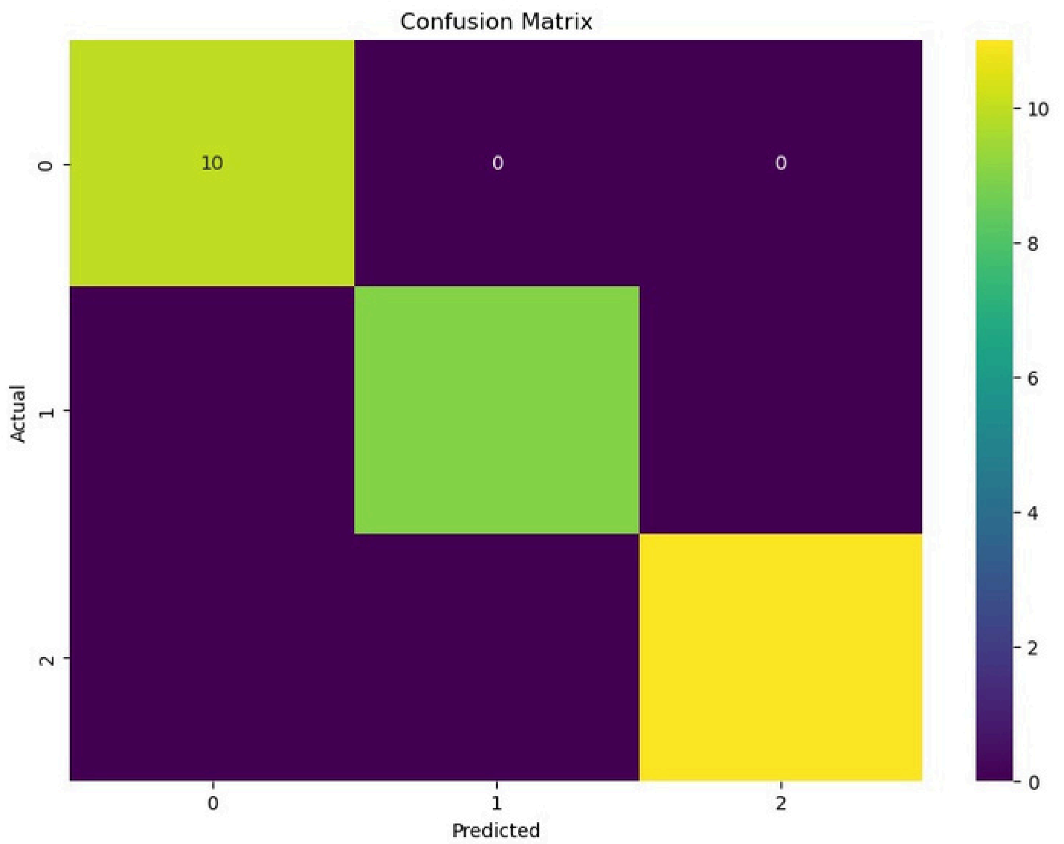
In [20]:
```
y_pred = clf.predict(X_test)
y_pred
```

Out[20]:
```
array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa'], dtype=object)
```

In [29]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
print(" Evaluate the model")
print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)
print('Classification Report:')
print(class_report)
```

```
  Evaluate  the  model
Accuracy:          1.0
Confusion Matrix: [[10
0 0]  [ 0 9 0]  [ 0 0
11]]    Classification
Report:

                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-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
```

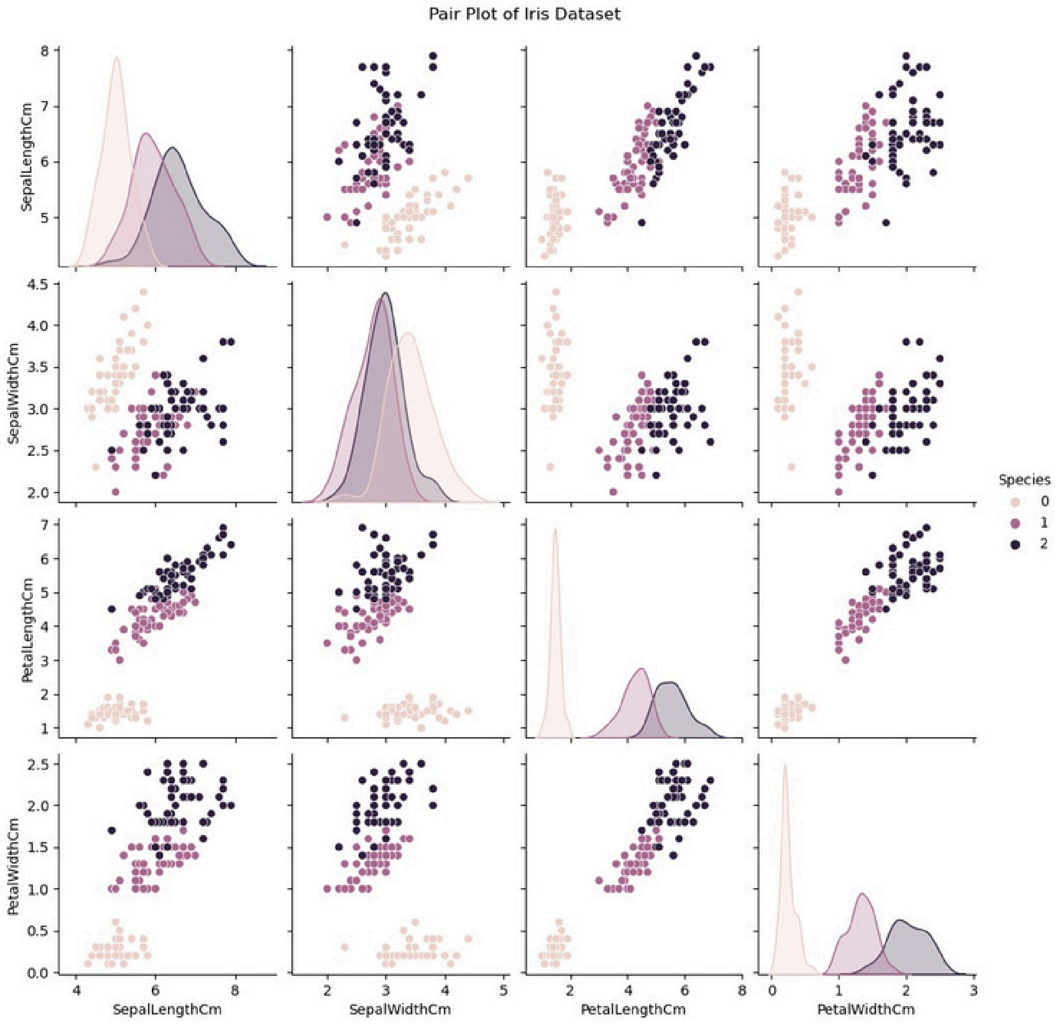
In [54]:
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Visualize the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='viridis', xticklabels=label_
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



In [55]:
```python
import seaborn as sns
```

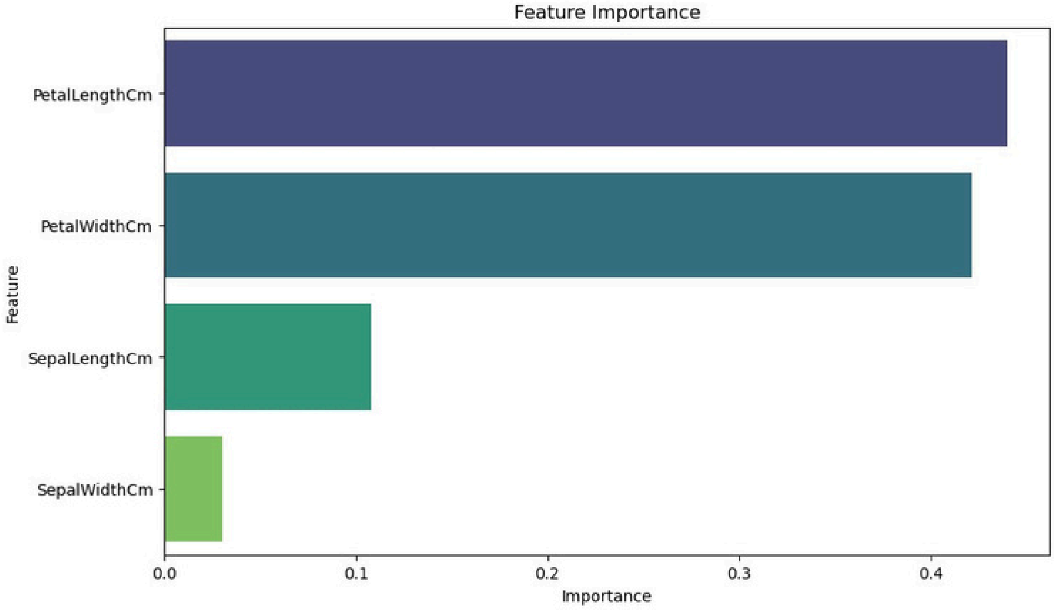
```
# Pair plot of the dataset
sns.pairplot(data, hue='Species', diag_kind='kde')
plt.suptitle('Pair Plot of Iris Dataset', y=1.02)
plt.show()
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

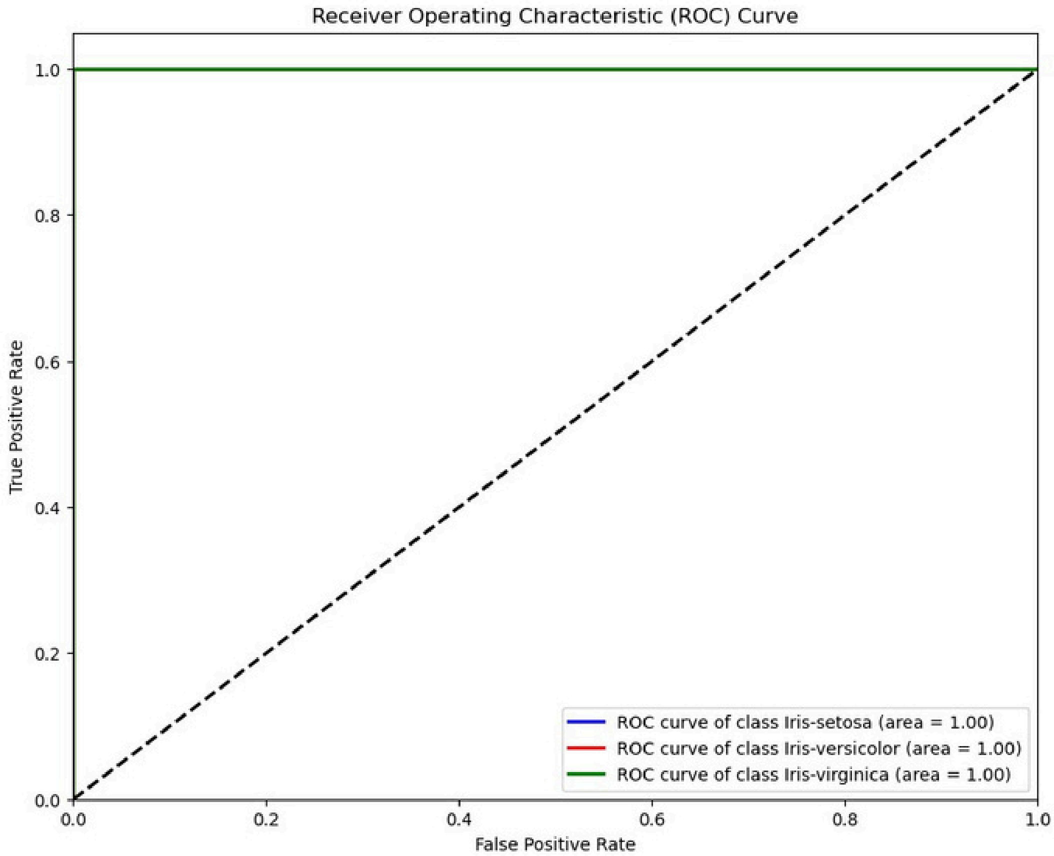


In [57]:
```
importances = clf.feature_importances_
features = X.columns

# Create a DataFrame for feature importances
```

```
feat_importances = pd.DataFrame({'Feature': features, 'Importance': importances}
feat_importances = feat_importances.sort_values(by='Importance', ascending=False

# Plot feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feat_importances, palette='viridis
plt.title('Feature Importance')
plt.show()
```



In [63]:
```
# Compute ROC curve and ROC area for each class
fpr = dict() tpr = dict() roc_auc = dict()



for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plot ROC curve for each class
plt.figure(figsize=(10, 8))
colors = ['blue', 'red', 'green']
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label=f'ROC curve of class {label_encoder.classes_[i]} (area = {roc

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```


Receiver Operating Characteristic (ROC) Curve
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
ROC curve of class Iris-setosa (area = 1.00)
ROC curve of class Iris-versicolor (area = 1.00)
ROC curve of class Iris-virginica (area = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

```
In [ ]:
```