IMPORT THE LIBRARIES

In [39]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

read the CSV file

In [2]:

```
df=pd.read_csv('iris.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [4]:

```
df.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
```

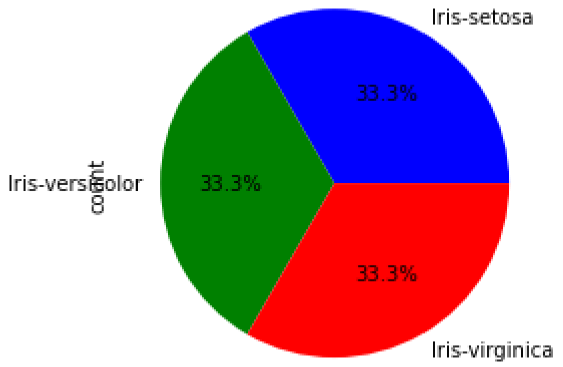
Visualize the data for idea

In [10]:

```
df['Species'].value_counts().plot(kind='pie',colors = ['blue', 'green', 'red'], autopct=
```

Out[10]:

```
<AxesSubplot:ylabel='count'>
```



In [11]:

```
df.duplicated().sum()
```

Out[11]:

```
0
```

Preprocess the data

In [18]:

```
X=df.drop(['Species'],axis=1).values
Y=df['Species'].values
```

In [20]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [21]:

```
df['Species']=le.fit_transform(df['Species'])
```

In [23]:

```
df.sample(5)
```

Out[23]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
11	12	4.8	3.4	1.6	0.2	0
81	82	5.5	2.4	3.7	1.0	1
27	28	5.2	3.5	1.5	0.2	0
56	57	6.3	3.3	4.7	1.6	1
140	141	6.7	3.1	5.6	2.4	2

Split the Data into Training and Testing Sets

In [24]:

```
from sklearn.model_selection import train_test_split
```

In [25]:

```
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=42)
```

Train a Machine Learning Model

In [26]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [27]:

```
rfc=RandomForestClassifier(random_state=42)
```

In [30]:

```
rfc.fit(X_train,y_train)
```

Out[30]:

```
RandomForestClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [31]:

```
y_pred=rfc.predict(X_test)
```

In [38]:

```
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[['Iris-versicolor' 'Iris-versicolor']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-setosa' 'Iris-setosa']]
```

Evaluate the Model

In [37]:

```
from sklearn.metrics import classification_report, accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 1.0
```

In [36]:

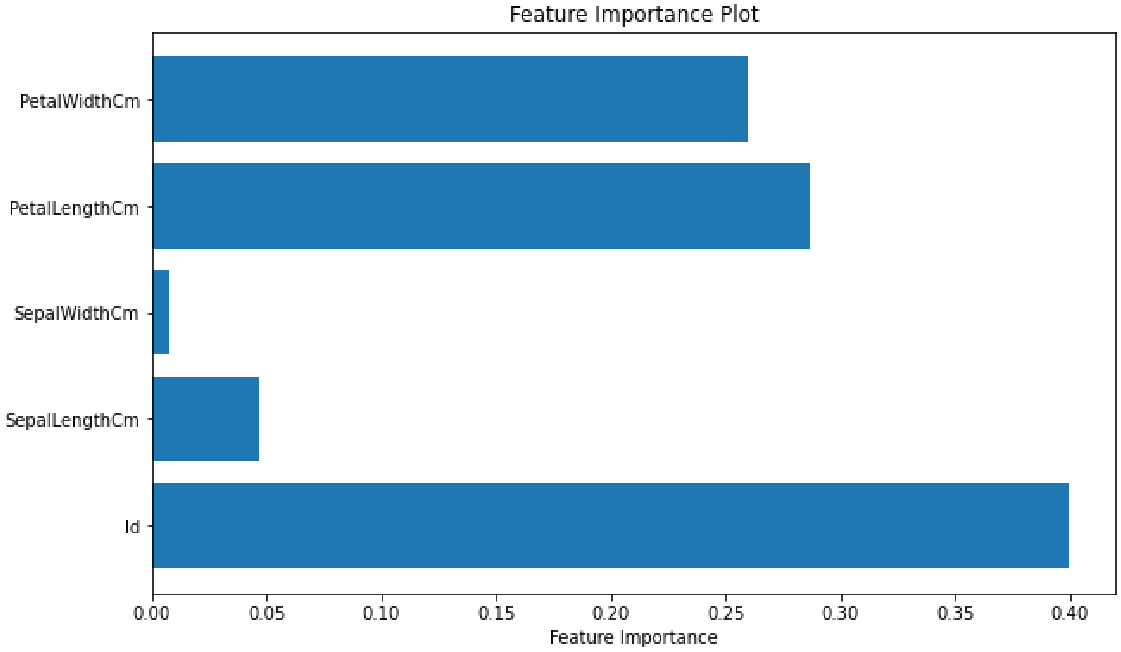
```
report = classification_report(y_test, y_pred)

print(f'Classification Report:\n{report}')
```

```
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
```

Visualize

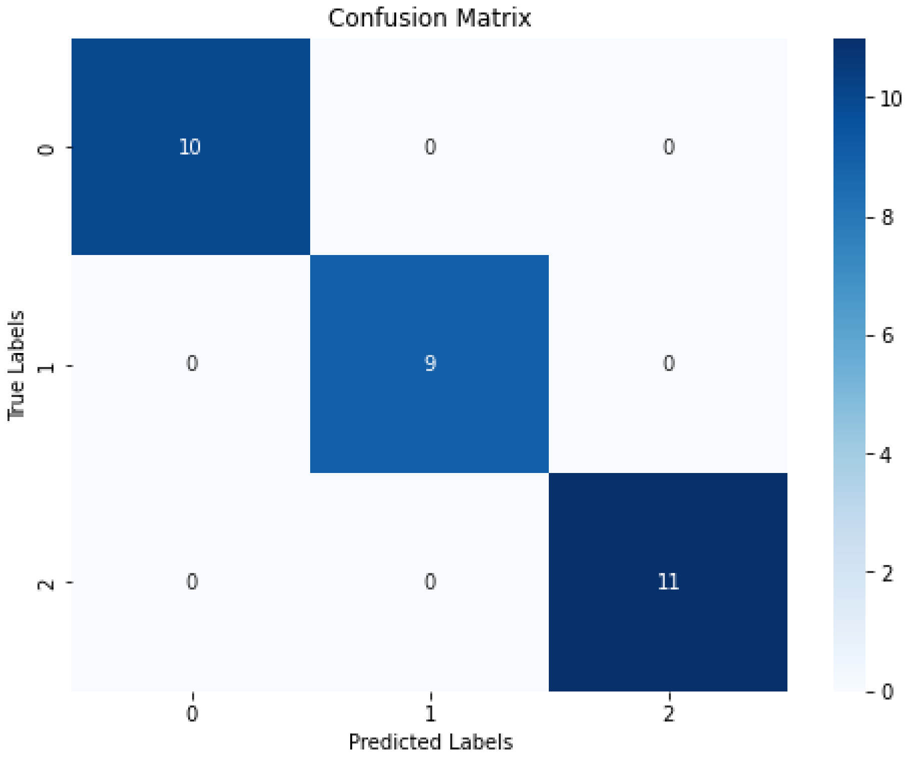
In [44]:

```
importances = rfc.feature_importances_
feature_names = df.columns[:-1]  # Assuming the last column is the target

plt.figure(figsize=(10, 6))
plt.barh(range(len(importances)), importances, align='center')
plt.yticks(range(len(importances)), feature_names)
plt.xlabel('Feature Importance')
plt.title('Feature Importance Plot')
plt.show()
```



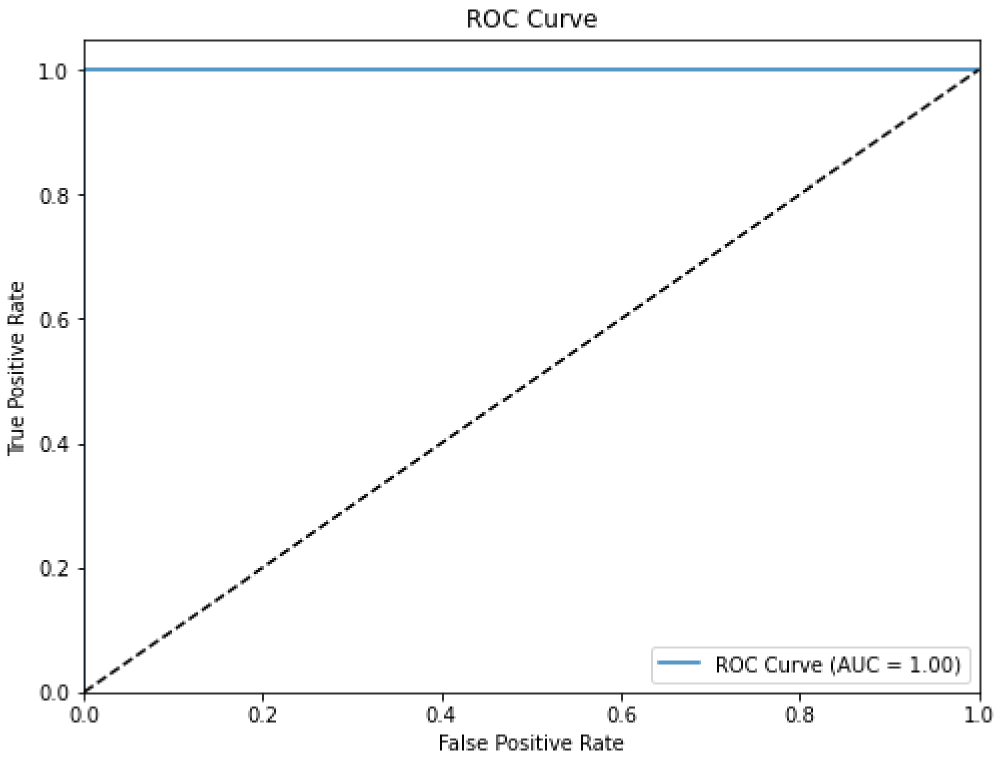
In [45]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



In [47]:

```
from sklearn.metrics import roc_curve, roc_auc_score
import numpy as np

# Assuming binary classification (e.g., for 'Iris-setosa' vs. 'Others')
y_setosa_binary = np.where(y_test == 'Iris-setosa', 1, 0)
y_pred_setosa_proba = rfc.predict_proba(X_test)[:, 0]

fpr, tpr, _ = roc_curve(y_setosa_binary, y_pred_setosa_proba)
auc = roc_auc_score(y_setosa_binary, y_pred_setosa_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```



In [ ]: