In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
```

In [2]:

```
df= pd.read_csv("C:\\Users\\Lenovo\\Desktop\\Intern\\iris flower detection\\IRIS.cs
```

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [4]:

```
df.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [5]:

```
df.tail()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

In [6]:

```
df.describe()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [7]:

```
df.shape
```

Out[7]:

```
(150, 5)
```

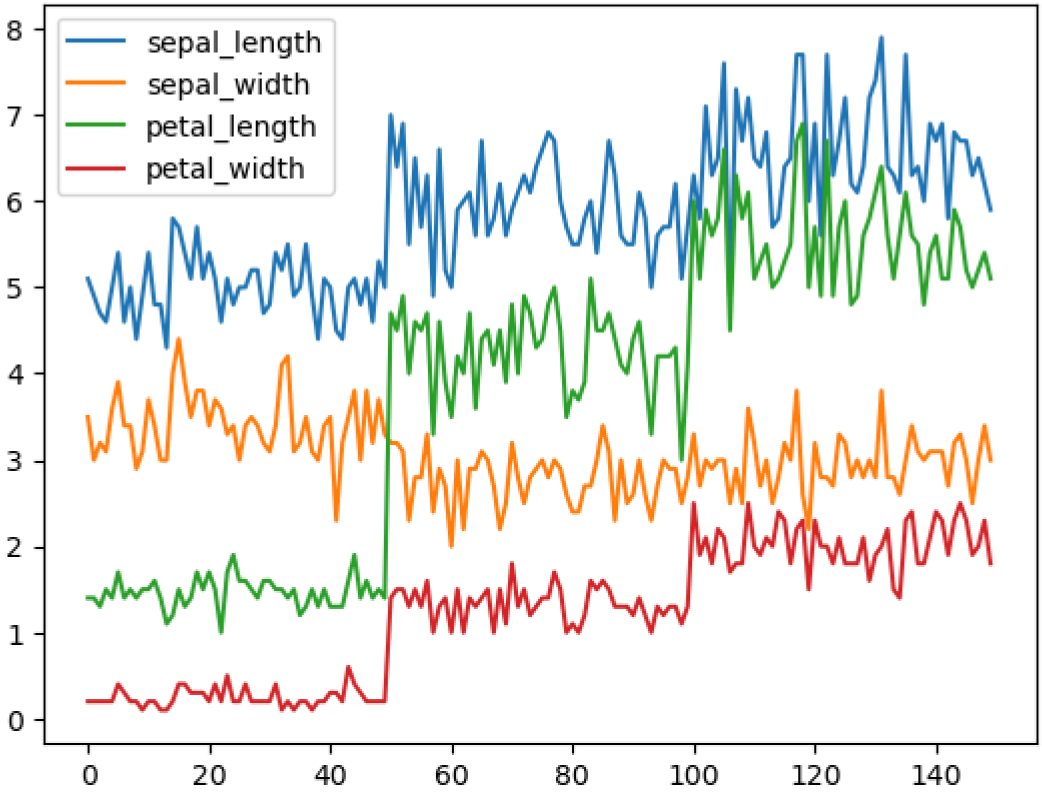
In [8]:

```
species = df['species'].value_counts()
print(species)
```

```
species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

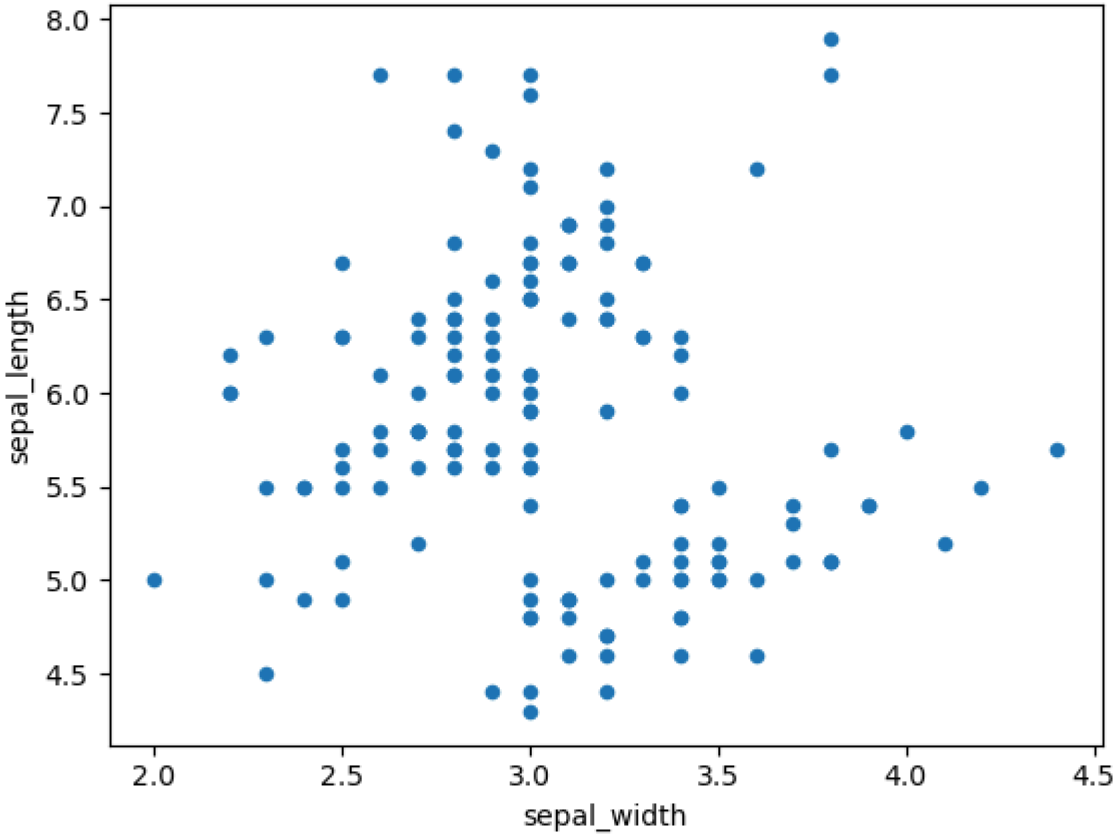
In [9]:

```
df.plot()
plt.show()
```



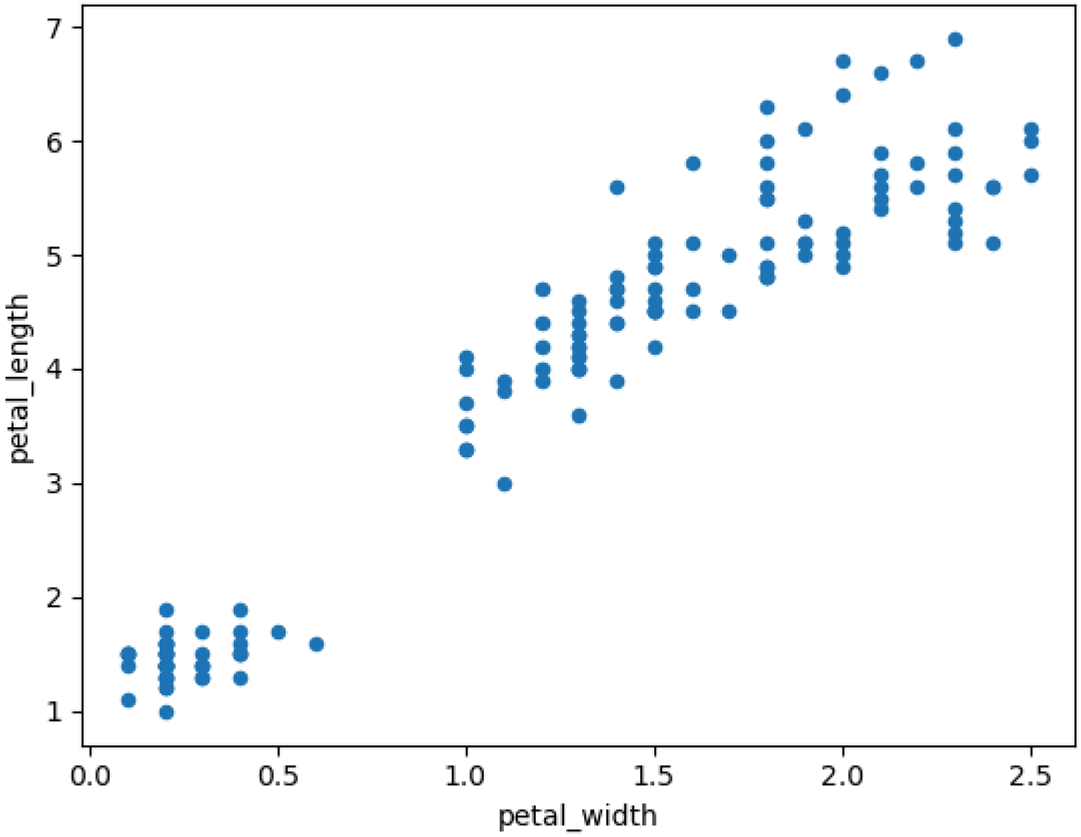
In [10]:

```
df.plot(kind = 'scatter', x = 'sepal_width', y = 'sepal_length')
plt.show()
```



In [11]:

```
df.plot(kind = 'scatter', x = 'petal_width', y = 'petal_length')
plt.show()
```



In [12]:

```
X = df.drop("species", axis=1)
y = df['species']
```

In [13]:

```python
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```

In [14]:

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_st
```

In [15]:

```python
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
```

Out[15]:

```
▾         DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

In [16]:

```python
y_pred = model.predict(X_test)
```

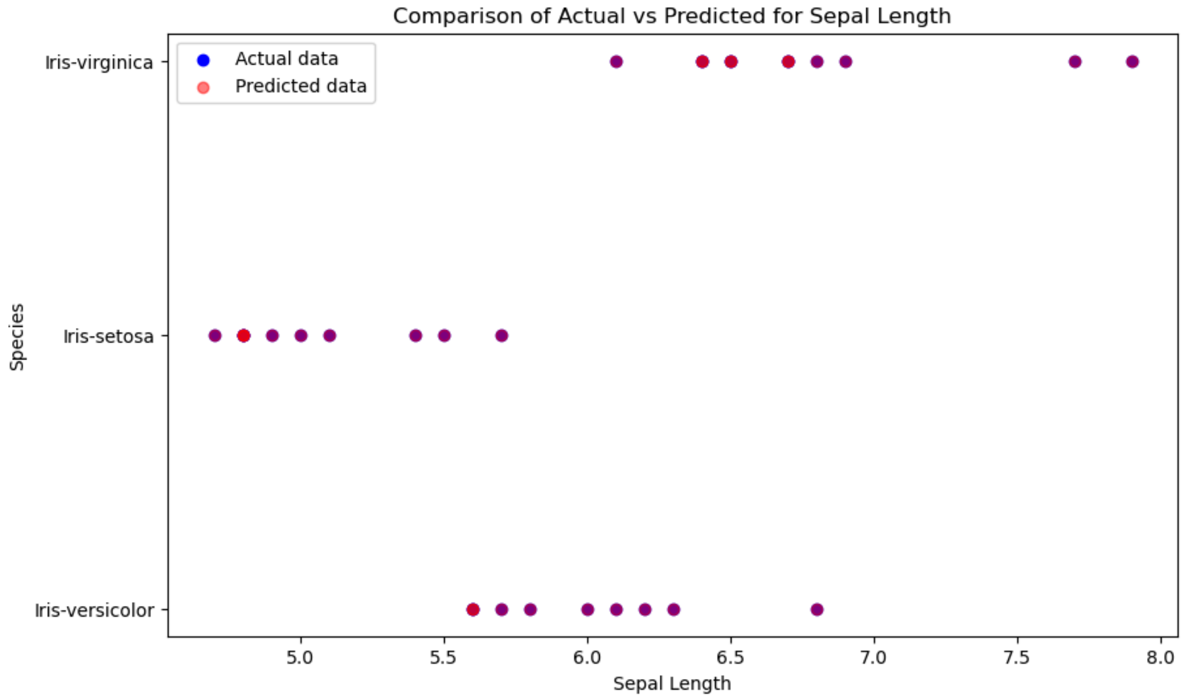
In [17]:

```python
print(f"Shape of X_test: {X_test.shape}")
print(f"Shape of y_pred: {y_pred.shape}")
```

```
Shape of X_test: (30, 4)
Shape of y_pred: (30,)
```

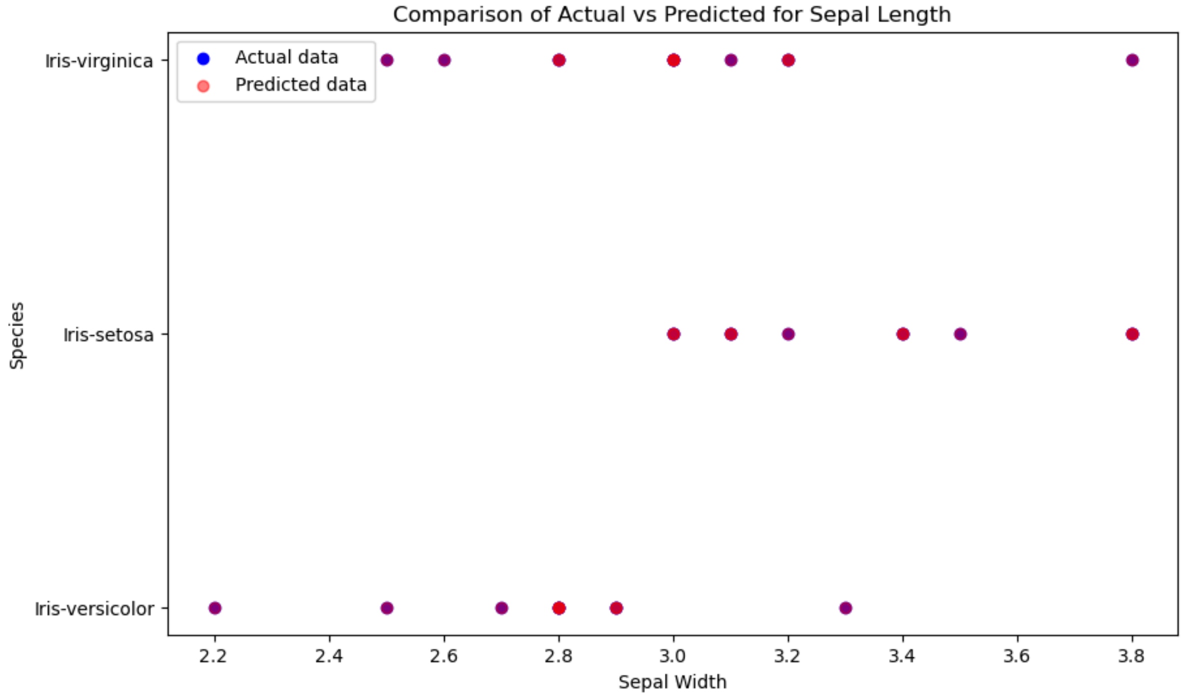
In [18]:

```python
plt.figure(figsize=(10, 6))
plt.scatter(X_test['sepal_length'], y_test, color='blue', label='Actual data')
plt.scatter(X_test['sepal_length'], y_pred, color='red', label='Predicted data', al
plt.xlabel('Sepal Length')
plt.ylabel('Species')
plt.title('Comparison of Actual vs Predicted for Sepal Length')
plt.legend()
plt.show()
```

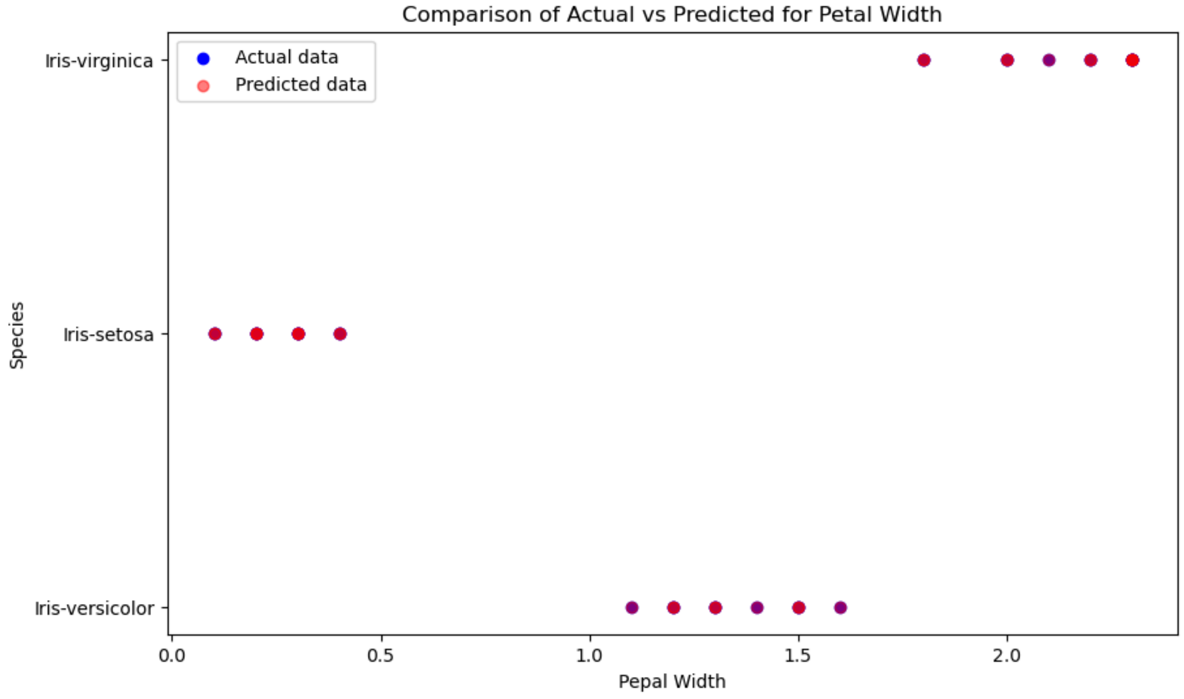


In [19]:

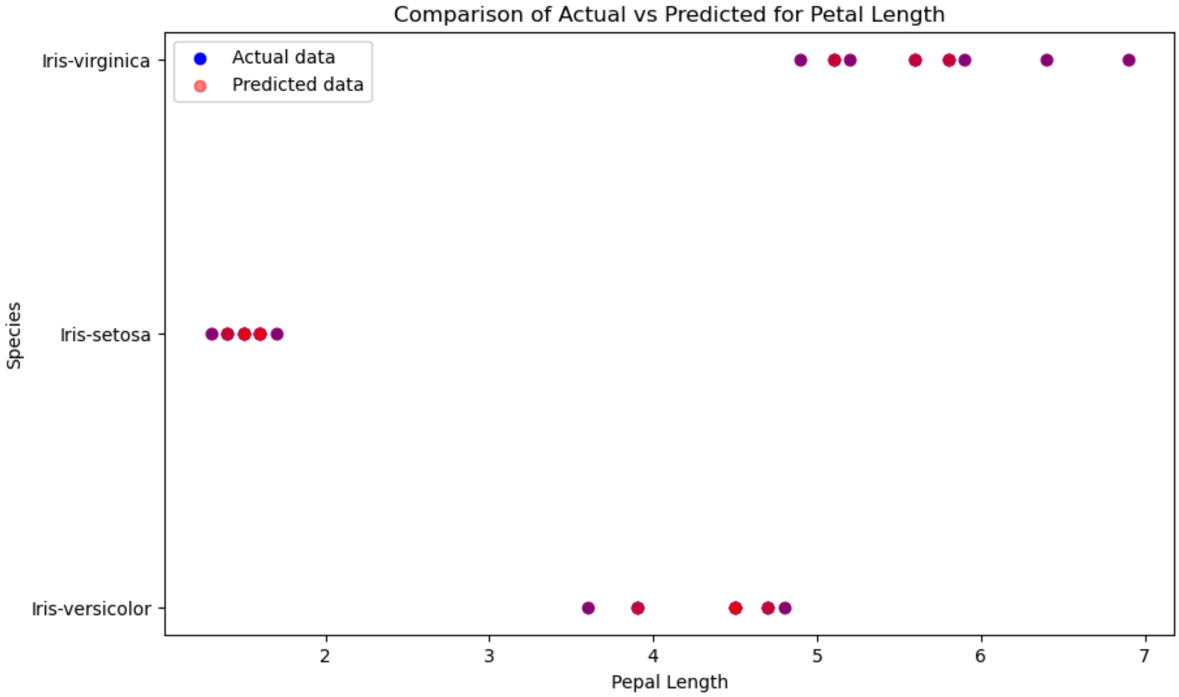
```python
plt.figure(figsize=(10, 6))
plt.scatter(X_test['sepal_width'], y_test, color='blue', label='Actual data')
plt.scatter(X_test['sepal_width'], y_pred, color='red', label='Predicted data', alp
plt.xlabel('Sepal Width')
plt.ylabel('Species')
plt.title('Comparison of Actual vs Predicted for Sepal Length')
plt.legend()
plt.show()
```



```
In [20]: plt.figure(figsize=(10, 6))
         plt.scatter(X_test['petal_width'], y_test, color='blue', label='Actual data')
         plt.scatter(X_test['petal_width'], y_pred, color='red', label='Predicted data', alp
         plt.xlabel('Pepal Width')
         plt.ylabel('Species')
         plt.title('Comparison of Actual vs Predicted for Petal Width')
         plt.legend()
         plt.show()
```



```
In [21]: plt.figure(figsize=(10, 6))
         plt.scatter(X_test['petal_length'], y_test, color='blue', label='Actual data')
         plt.scatter(X_test['petal_length'], y_pred, color='red', label='Predicted data', al
         plt.xlabel('Pepal Length')
         plt.ylabel('Species')
         plt.title('Comparison of Actual vs Predicted for Petal Length')
         plt.legend()
         plt.show()
```



In [23]:

```
import numpy as np
from sklearn.metrics import accuracy_score, classification_report
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
report = classification_report(y_test, y_pred)
print(f"Classification Report:\n{report}")
```

```
Accuracy: 1.00
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 [ ]: