DONE BY PIYUSH WORLIKAR

The IRIS dataset contains three classes of flowers, Versicolor, Setosa, Virginica, and each class contains 4 features, 'Sepal length', 'Sepal width', 'Petal length', 'Petal width'. The aim of the iris flower classification is to predict flowers based on their specific features.

IMPORT THE NECESSARY LIBRARIES:

```
import numpy as np       #NUMPY IS USED FOR NUMERICAL OPERATIONS
import pandas as pd # PANDAS FOR IMPORTING THE DATASET
from sklearn.model_selection import train_test_split    #TO SPLIT DATA INTO TRAINING AND TESTING DATA
from  sklearn.tree import DecisionTreeClassifier    #FOR DECSION TREE CLASSIFIER
from sklearn.metrics import accuracy_score   #TO CHECK ACCURACY
import matplotlib.pyplot as plt     # FOR DATA VISUALIZATION PURPOSE
from sklearn import tree   # TO VISUALIZE THE TREE
import seaborn as sns     # FOR DATA VISUALIZATION

from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

LOAD THE IRIS DATASET:

```
data = pd.read_csv("/content/Iris.csv")    # using the Pandas library's read_csv function. The dataset is s
data
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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
...	...	...	...	...	...
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

150 rows × 5 columns

EXPLORE THE DATASET: The code loads the Iris dataset and assigns the features (all columns except 'species') and target ('species') to separate variables.

```
data.head()                    # Display the first few rows of the dataset

```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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

```
data.shape    # Check the shape of the dataset
```

```
(150, 5)
```

```
data.columns                      # Check the column names
```

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

```
data.describe()       #will display the stats of each column
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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

```
data.info()   # provides a summary of the dataset, including the number of non-null values, data types of
```

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

```
data.nunique()     #to check unique values
```

	0
SepalLengthCm	35
SepalWidthCm	23
PetalLengthCm	43
PetalWidthCm	22
Species	3

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

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

```
data['Species'].unique()                # Check the unique values in the target variable (species)
```

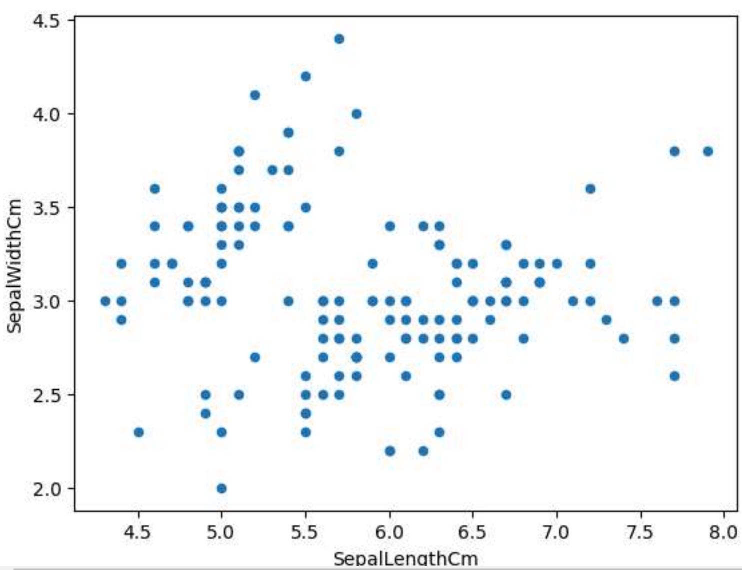
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
data['Species'].value_counts()                    # Check the distribution of the target variable

```

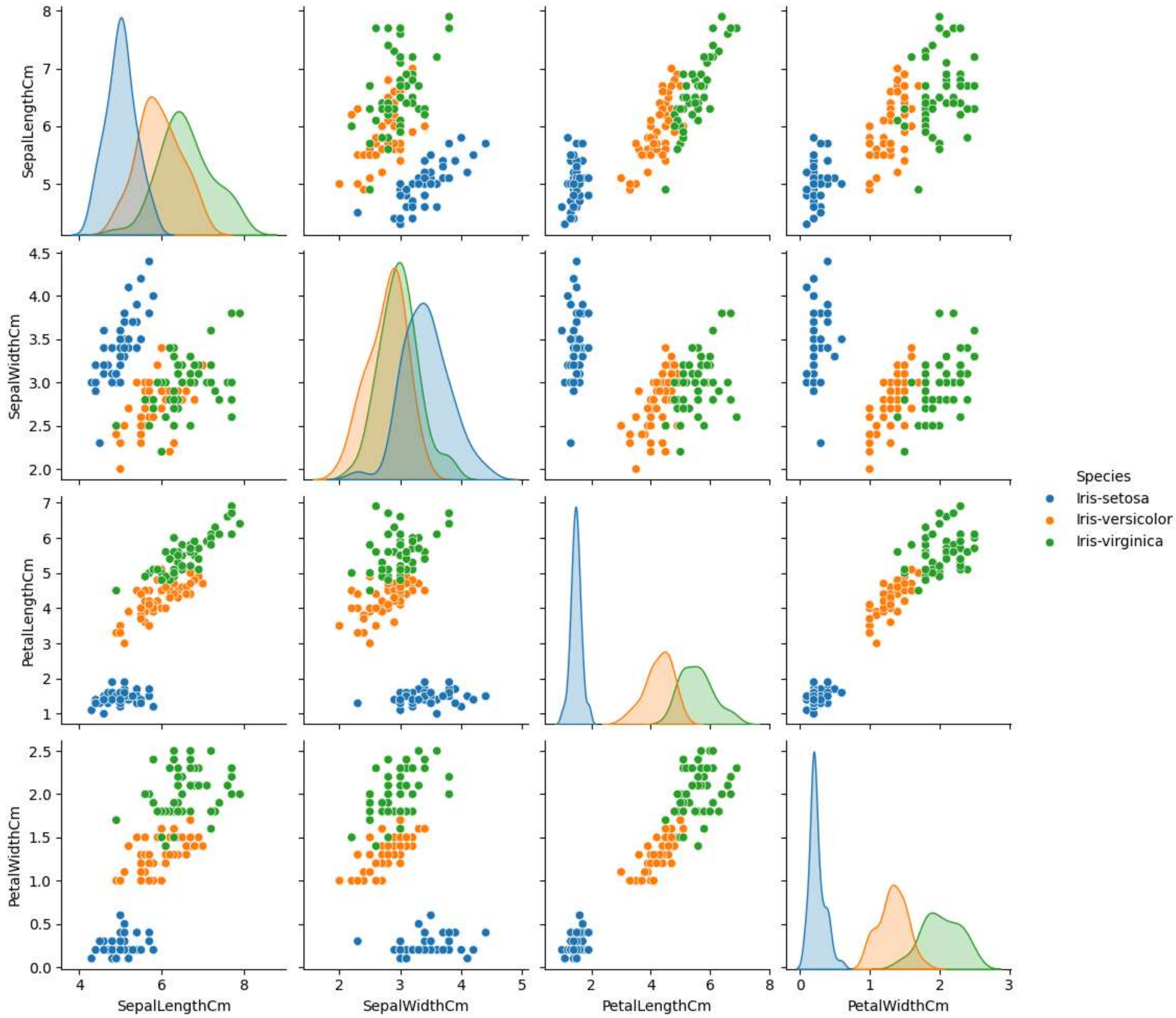
Species	count
Iris-setosa	50
Iris-versicolor	50
Iris-virginica	50

DATA VISUALIZATION: This section creates a scatter plot of 'sepal_length' on the x-axis and 'sepal_width' on the y-axis. It visualizes the relationship between these two features in the Iris dataset.

```
data.plot(kind='scatter', x='SepalLengthCm', y='SepalWidthCm')
plt.show()

```



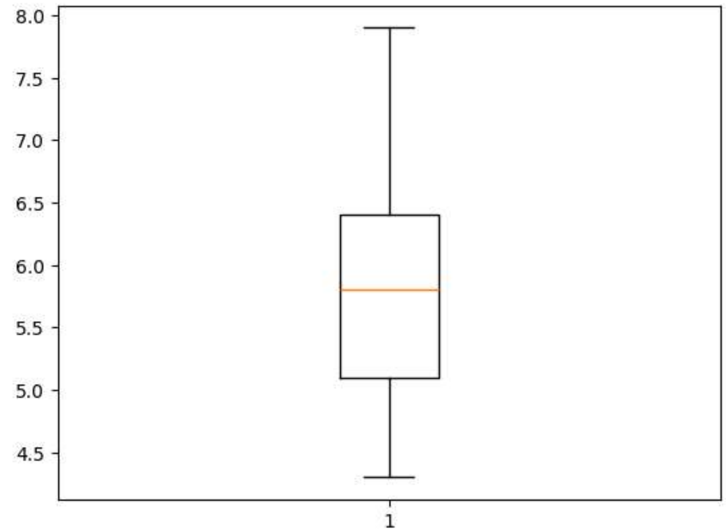

```
sns.pairplot(data,hue='Species')
plt.show()
```




```
1 #Boxplots help identify the distribution, central tendency, and outliers in the data.
```

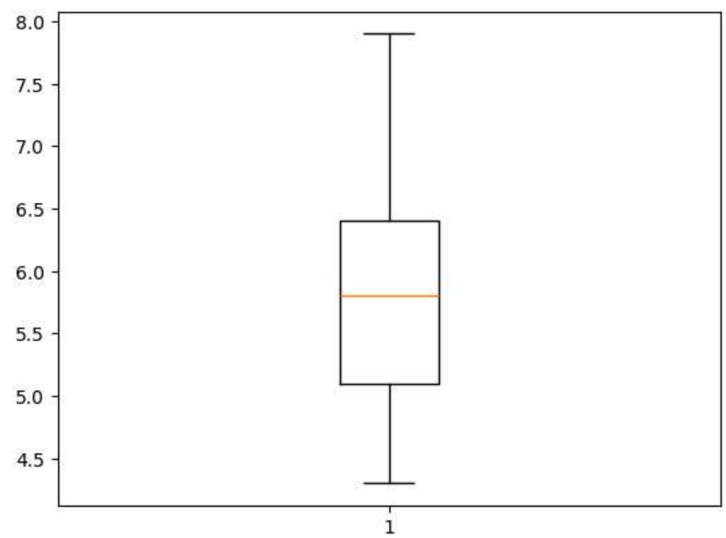
```
1 # boxplot - to check for outliers
2 plt.boxplot(data['SepalLengthCm'])
3
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f229034d0f0>,
  <matplotlib.lines.Line2D at 0x7f229034d8d0>],
 'caps': [<matplotlib.lines.Line2D at 0x7f229034f010>,
  <matplotlib.lines.Line2D at 0x7f229034f460>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f229034e380>],
 'medians': [<matplotlib.lines.Line2D at 0x7f229034fa30>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f229034f880>],
 'means': []}
```

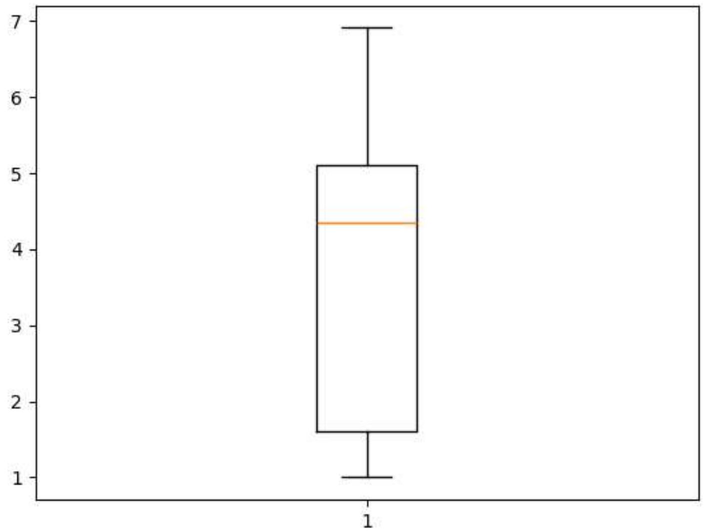

```
# boxplot
plt.boxplot(data['SepalLengthCm'])

```


```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f2290310e20>,
  <matplotlib.lines.Line2D at 0x7f22903107f0>],
 'caps': [<matplotlib.lines.Line2D at 0x7f2290310730>,
  <matplotlib.lines.Line2D at 0x7f22903109d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f2290313f70>],
 'medians': [<matplotlib.lines.Line2D at 0x7f2290310f40>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f2290311720>],
 'means': []}
```

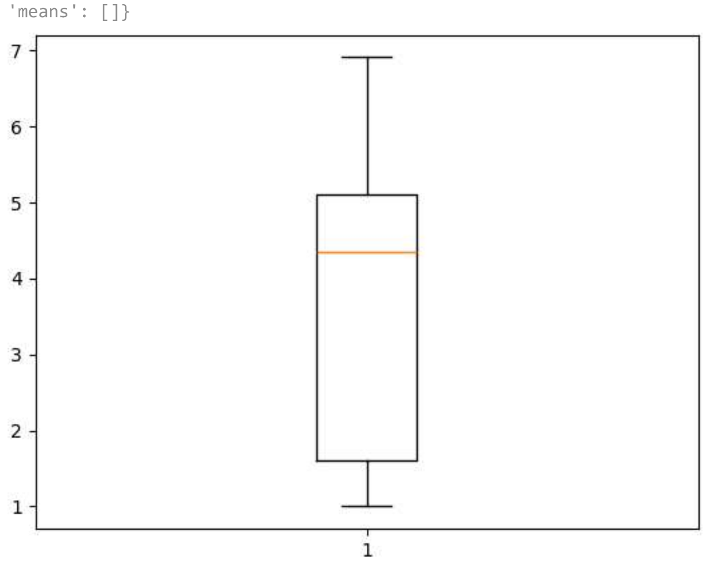

```
# boxplot
plt.boxplot(data['PetalLengthCm'])

```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f229692fa90>,
  <matplotlib.lines.Line2D at 0x7f229692c040>],
 'caps': [<matplotlib.lines.Line2D at 0x7f229692e320>,
  <matplotlib.lines.Line2D at 0x7f229692c220>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f229692ee60>],
 'medians': [<matplotlib.lines.Line2D at 0x7f229692dc00>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f229692fdf0>],
 'means': []}
```




```
# boxplot
plt.boxplot(data['PetalLengthCm'])

```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f22977b4940>,
  <matplotlib.lines.Line2D at 0x7f22977b56f0>],
 'caps': [<matplotlib.lines.Line2D at 0x7f22977b44f0>,
  <matplotlib.lines.Line2D at 0x7f22977b6dd0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f22977b52a0>],
 'medians': [<matplotlib.lines.Line2D at 0x7f2294405930>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f2294405390>],
 'means': []}
```



DATA PREPARATION: This section prepares the dataset by separating the features (X) and the target variable (y) and then splitting the data into training and testing sets using the train_test_split function.

```
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_state=42)
```

TRAIN THE DECISION TREE CLASSIFIER: A decision tree classifier is instantiated and trained using the training data (x_train and y_train) using the fit method.

```
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

MAKE PREDICITIONS ON THE TEST SET

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

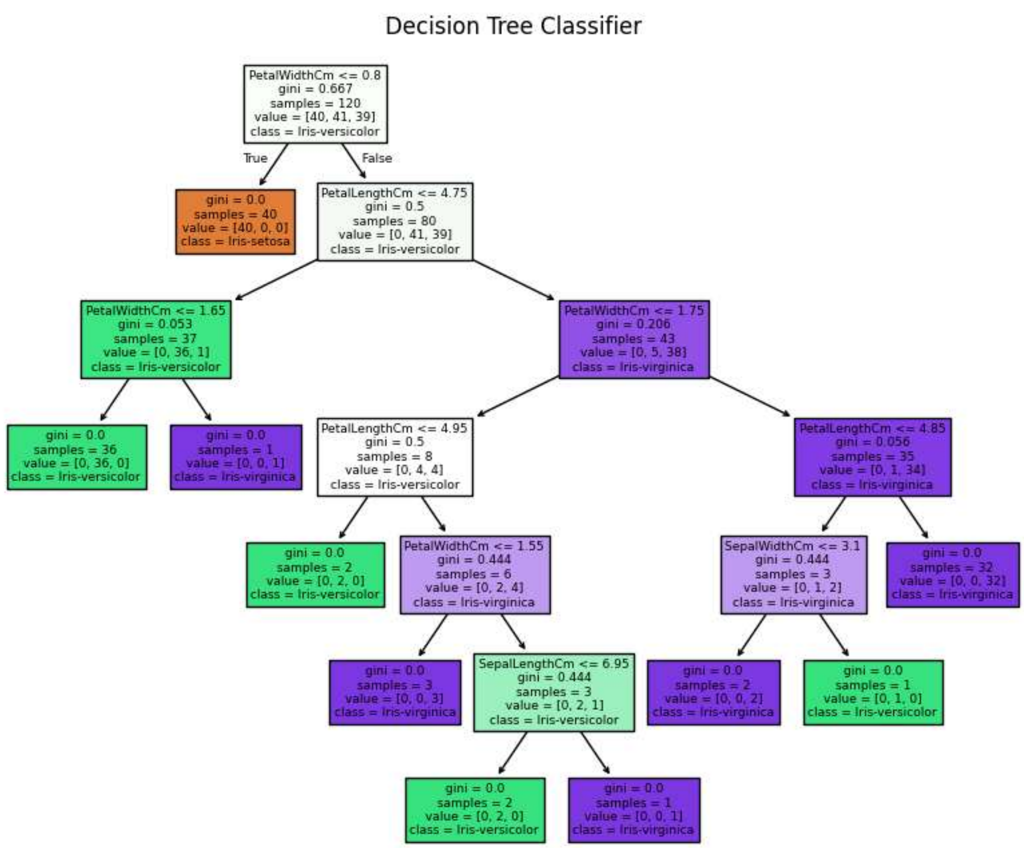
CALCULATE THE ACCURACY OF THE MODEL:

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

VISUALIZE THE DECISON TREE :

```
fig = plt.figure(figsize=(10, 8))
_ = tree.plot_tree(clf, feature_names=X.columns, class_names=clf.classes_, filled=True)
plt.title("Decision Tree Classifier")
plt.show()
```



USING LOGISTIC REGRESSION:

A logistic regression model is instantiated and trained using the training data (X_train and y_train) using the fit method.

Prepare the data:

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

```

TRAIN THE LOGISTIC REGRESSION MODEL

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)

```

```
LogisticRegression
LogisticRegression()
```

MAKE PREDICTIONS:

```
predictions = model.predict(X_test)
```

EVALUATE THE MODEL:

```
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

USING KNEIGHBORS CLASSIFIER:

```
# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

TRAIN THE MODEL:

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

MAKE PREDICTIONS:

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

EVALUATE THE MODEL:

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
                 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
```

PLOT CONFUSION MATRIX:

```
sns.heatmap(confusion_matrix(y_test,y_pred), annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
```