- Name : SAGNIK SAMANTA
- Internship ID : UMIP26929
- Internship : Data Analyst Intern at UnifiedMentor
- Project : IRIS Classification

Project Description

In this Project we are provided with Iris flower Dataset containing three species of flowers (Setosa, Versicolour, and Virginica).Here our main objective is to build a machine learning model that can classify Iris flowers into three categories Setosa, Versicolour and Virginica based on the length and width of their petals and sepals.K-Nearest Neighbor,Decision Tree,Random Forest,Support Vector Machine and Logistic Regression have been employed to train the machine learning model and checked the accuracy of the model using metrices like accuracy score, precision, recall, and F1-score.

Column Description

Iris flower Dataset contains 150 observations on 5 features.These are listed below

- Sepal Length(cm)
- Sepal Width(cm)
- Petal Length(cm)
- Petal Width(cm)
- Species

K- Nearest Neighborhood

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matri
```

In [2]:
```
## import Dataset
df=pd.read_csv("C:/Users/SAGNIK SAMANTA/OneDrive/Desktop/Datasets/Iris.csv")
df.head()
```

Out[2]:

	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 Inspection

In [3]:
```
## Dimension of the Dataset
print("Shape of the data frame: ",df.shape)
```

```
Shape of the data frame:  (150, 5)
```

In [4]:
```
## Checking for Missing Values
print("Total null values: ",df.isna().sum().sum())
```

```
Total null values:  0
```

In [5]:
```
## Checking for Duplicate Values
print("Duplicate values: ",df.duplicated().sum() )
```

```
Duplicate values:  3
```

Removing the Duplicated Values

In [6]:
```
df.drop_duplicates(inplace=True)
print("Shape of the data frame: ",df.shape)
print("\n")
print("Species categories with its count \n",df["Species"].value_counts())
```

```
Shape of the data frame:  (147, 5)


Species categories with its count 
 Species
Iris-versicolor    50
Iris-virginica     49
Iris-setosa        48
Name: count, dtype: int64
```

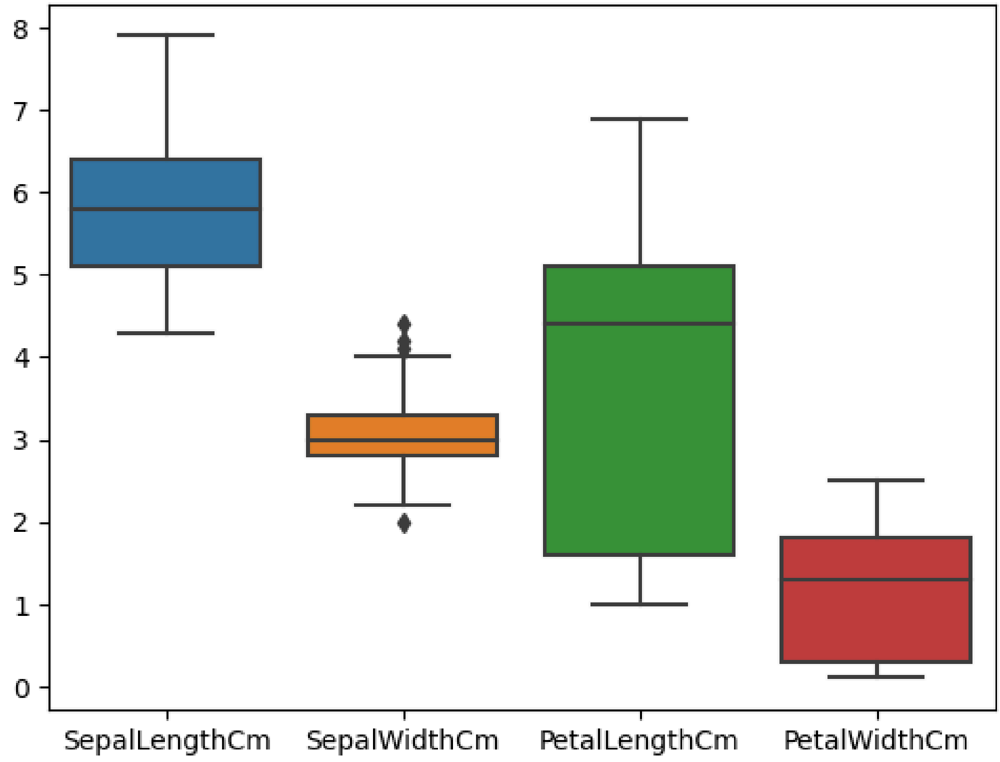
In [7]:
```
df.describe()
```

Out[7]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	147.000000	147.000000	147.000000	147.000000
mean	5.856463	3.055782	3.780272	1.208844
std	0.829100	0.437009	1.759111	0.757874
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [8]:
```
## Boxplot
sns.boxplot(df)
```

Out[8]: <Axes: >



In [9]:
```
df.columns
```

Out[9]:
```
Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [10]:
```
## Separate the independent and dependent variables using the slicing method.
X=df.drop("Species",axis=1)
y=df[['Species']]
```

In [11]:
```
## Split the data into training and testing sets.
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=100)
```

In [12]:
```
## Scale the data using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [13]:
```
## Reshape y to a 1D array
y_train = np.ravel(y_train)
y_test = np.ravel(y_test)
```

In [14]:
```
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled,y_train)
```

Out[14]:
```
▾ KNeighborsClassifier

KNeighborsClassifier()
```

In [15]:
```
## Calculate the Accuracy of the model
Prediction=knn.predict(X_test_scaled)
print('The accuracy of the KNN is',accuracy_score(Prediction,y_test))
print("Classification Report:")
print(classification_report(y_test, Prediction))
print("Confusion Matrix:")
print(confusion_matrix(y_test, Prediction))
```

```
The accuracy of the KNN is 0.9
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         9
Iris-versicolor       0.70      1.00      0.82         7
 Iris-virginica       1.00      0.79      0.88        14

       accuracy                           0.90        30
      macro avg       0.90      0.93      0.90        30
   weighted avg       0.93      0.90      0.90        30

Confusion Matrix:
[[ 9  0  0]
 [ 0  7  0]
 [ 0  3 11]]
```

Conclusion :

- Precision : Out of all the flower species that the model predicted would get classified into three categories,the model perdicted the outcomes as 100% as Setosa,70% for Versicolor and 100% for Virginica
- Recall : Out of all the flower species that actually got classified into three categories,the model perdicted the outcomes correctly as 100% as Setosa,100% for Versicolor and 79% for Virginica
- F1-Score : This value is calculated as = 2 * (Precision * Recall) / (Precision + Recall)
  - Since the value is close to 1 , therefore we can state that our model correctly classified flowers into three flower species
- Support : These values simply tell us how many flowers belonged to each class in the test dataset.We can see that among the flowers in the test dataset, 9 flowers belong to Setosa, 14 flowers belongs to verginica and 7 flowers belong to versicolor.

Decision Tree Classifier

In [16]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matri
from sklearn import tree
import matplotlib.pyplot as plt
```

In [17]:
```
## import Dataset
df=pd.read_csv("C:/Users/SAGNIK SAMANTA/OneDrive/Desktop/Datasets/Iris.csv")
df.head()
```

Out[17]:

	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

In [18]:
```
df.columns
```

Out[18]:
```
Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [19]:
```
## Separate the independent and dependent variables using the slicing method.
X=df.drop("Species",axis=1)
y=df[['Species']]
```

In [20]:
```
## Split the data into training and testing sets.
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=100)
```

In [21]:
```
## Train the model using the decision tree classifier.
clf_gini=DecisionTreeClassifier(criterion="gini",random_state=100,max_depth=3,min_
clf_gini.fit(X_train,y_train)
```

Out[21]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [22]:
```
y_Predict_gini=clf_gini.predict(X_test)
y_Predict_gini
```

Out[22]:
```
array(['Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica',
       'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [23]:
```
## Calculate the accuracy of the model using the accuracy score function.
print(("Accuracy is"),accuracy_score(y_test,y_Predict_gini)*100)
print("Classification Report:")
print(classification_report(y_test, y_Predict_gini))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_Predict_gini))
```

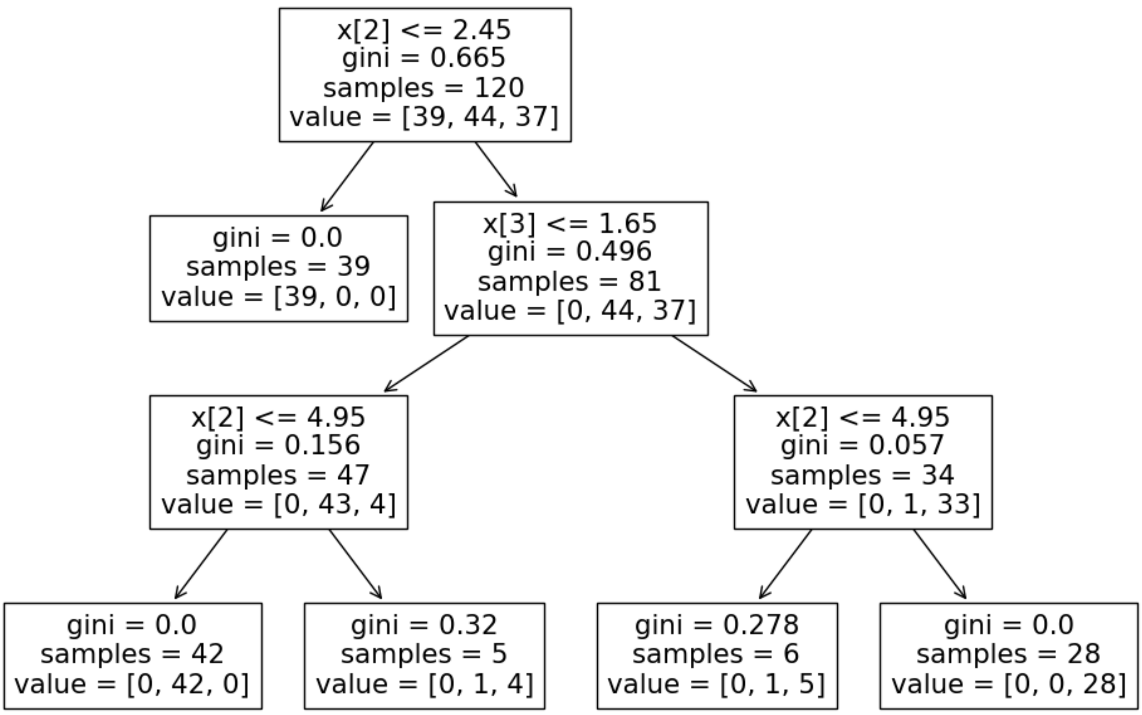
```
Accuracy is 96.66666666666667
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.83      0.91         6
 Iris-virginica       0.93      1.00      0.96        13

       accuracy                           0.97        30
      macro avg       0.98      0.94      0.96        30
   weighted avg       0.97      0.97      0.97        30

Confusion Matrix:
[[11  0  0]
 [ 0  5  1]
 [ 0  0 13]]
```

Our prediction model shows that there is an excellent accuracy score of 96.66666666666667 percent.

- Precision : Out of all the flower species that the model predicted would get classified into three categories,the model perdicted the outcomes as 100% as Setosa,100% for Versicolor and 93% for Virginica
- Recall : Out of all the flower species that actually got classified into three categories,the model perdicted the outcomes correctly as 100% as Setosa,83% for Versicolor and 100% for Virginica
- F1-Score : This value is calculated as = 2 * (Precision * Recall) / (Precision + Recall)
  - Since the value is very close to 1 , therefore we can state that our model correctly classified flowers into three flower species
- Support : These values simply tell us how many flowers belonged to each class in the test dataset.We can see that among the flowers in the test dataset, 11 flowers belong to Setosa, 13 flowers belongs to verginica and 6 flowers belong to versicolor.

In [24]:
```
plt.figure(figsize=(12,8))
tree.plot_tree(clf_gini.fit(X_train, y_train))
```

Out[24]:
```
[Text(0.375, 0.875, 'x[2] <= 2.45\ngini = 0.665\nsamples = 120\nvalue = [39, 44, 
37]'),
 Text(0.25, 0.625, 'gini = 0.0\nsamples = 39\nvalue = [39, 0, 0]'),
 Text(0.5, 0.625, 'x[3] <= 1.65\ngini = 0.496\nsamples = 81\nvalue = [0, 44, 3
7]'),
 Text(0.25, 0.375, 'x[2] <= 4.95\ngini = 0.156\nsamples = 47\nvalue = [0, 43, 
4]'),
 Text(0.125, 0.125, 'gini = 0.0\nsamples = 42\nvalue = [0, 42, 0]'),
 Text(0.375, 0.125, 'gini = 0.32\nsamples = 5\nvalue = [0, 1, 4]'),
 Text(0.75, 0.375, 'x[2] <= 4.95\ngini = 0.057\nsamples = 34\nvalue = [0, 1, 3
3]'),
 Text(0.625, 0.125, 'gini = 0.278\nsamples = 6\nvalue = [0, 1, 5]'),
 Text(0.875, 0.125, 'gini = 0.0\nsamples = 28\nvalue = [0, 0, 28]')]
```



In [25]:
```
## Entropy Method
## Train the model using the decision tree classifier.
clf_en=DecisionTreeClassifier(criterion="entropy",random_state=100,max_depth=3,min
clf_en.fit(X_train,y_train)
```

Out[25]:
```
DecisionTreeClassifier

DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samples_leaf=5,
                       random_state=100)
```

In [26]:
```
y_Predict_en=clf_en.predict(X_test)
y_Predict_en
```

Out[26]:
```
array(['Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica',
       'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [27]:
```
## Calculate the accuracy of the model using the accuracy score function.
print(("Accuracy is"),accuracy_score(y_test,y_Predict_en)*100)
print("Classification Report:")
print(classification_report(y_test, y_Predict_en))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_Predict_en))
```

```
Accuracy is 96.66666666666667
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.83      0.91         6
 Iris-virginica       0.93      1.00      0.96        13

       accuracy                           0.97        30
      macro avg       0.98      0.94      0.96        30
   weighted avg       0.97      0.97      0.97        30

Confusion Matrix:
[[11  0  0]
 [ 0  5  1]
 [ 0  0 13]]
```

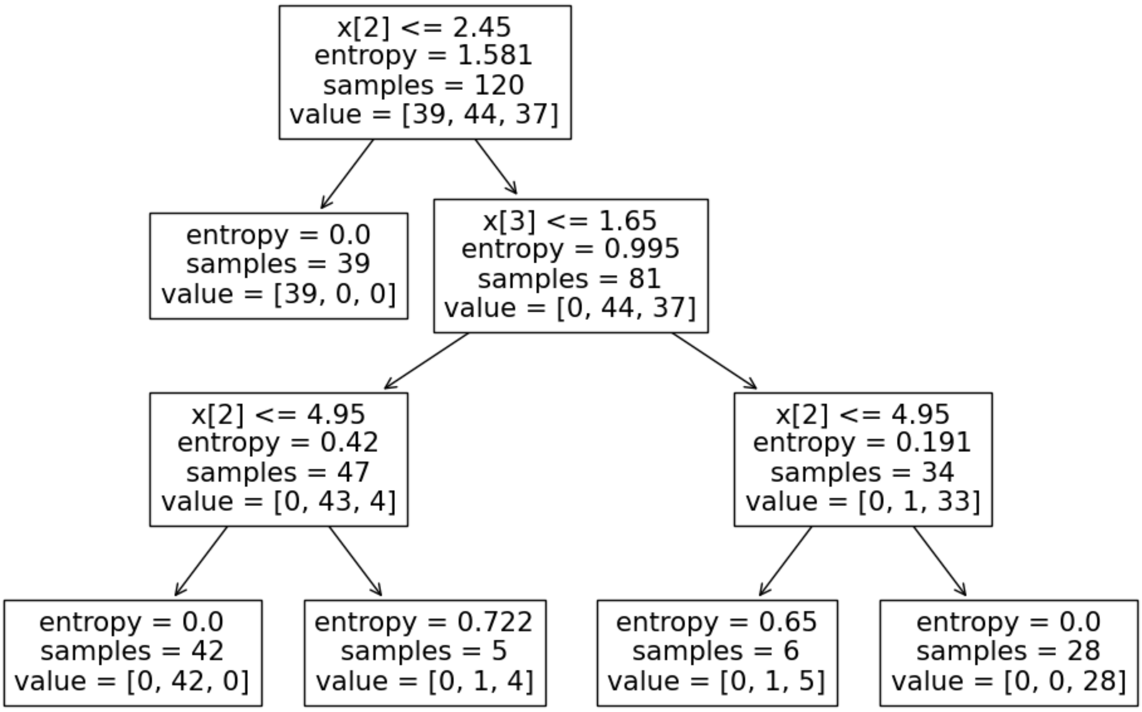
Our prediction model shows that there is an excellent accuracy score of 96.66666666666667 percent.

- Precision : Out of all the flower species that the model predicted would get classified into three categories,the model perdicted the outcomes as 100% as Setosa,100% for Versicolor and 93% for Virginica
- Recall : Out of all the flower species that actually got classified into three categories,the model perdicted the outcomes correctly as 100% as Setosa,83% for Versicolor and 100% for Virginica
- F1-Score : This value is calculated as = 2 * (Precision * Recall) / (Precision + Recall)
  - Since the value is very close to 1 , therefore we can state that our model correctly classified flowers into three flower species
- Support : These values simply tell us how many flowers belonged to each class in the test dataset.We can see that among the flowers in the test dataset, 11 flowers belong to Setosa, 13 flowers belongs to verginica and 6 flowers belong to versicolor.

Visualize decision-trees

In [28]:
```
plt.figure(figsize=(12,8))
tree.plot_tree(clf_en.fit(X_train, y_train))
```

Out[28]:
```
[Text(0.375, 0.875, 'x[2] <= 2.45\nentropy = 1.581\nsamples = 120\nvalue = [39, 4
4, 37]'),
 Text(0.25, 0.625, 'entropy = 0.0\nsamples = 39\nvalue = [39, 0, 0]'),
 Text(0.5, 0.625, 'x[3] <= 1.65\nentropy = 0.995\nsamples = 81\nvalue = [0, 44, 3
7]'),
 Text(0.25, 0.375, 'x[2] <= 4.95\nentropy = 0.42\nsamples = 47\nvalue = [0, 43,
4]'),
 Text(0.125, 0.125, 'entropy = 0.0\nsamples = 42\nvalue = [0, 42, 0]'),
 Text(0.375, 0.125, 'entropy = 0.722\nsamples = 5\nvalue = [0, 1, 4]'),
 Text(0.75, 0.375, 'x[2] <= 4.95\nentropy = 0.191\nsamples = 34\nvalue = [0, 1, 3
3]'),
 Text(0.625, 0.125, 'entropy = 0.65\nsamples = 6\nvalue = [0, 1, 5]'),
 Text(0.875, 0.125, 'entropy = 0.0\nsamples = 28\nvalue = [0, 0, 28]')]
```



In [29]:
```
## Calculate the training set Accuracy
y_pred_train_en = clf_en.predict(X_train)
y_pred_train_en
```

Out[29]:
```
array(['Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-setosa', 'Iris-setosa'], dtype=object)
```

In [30]:
```
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pr
```

```
Training-set accuracy score: 0.9833
```

In [31]:
```
## Check for Overfitting and Underfitting
## print the scores on training and test set

print('Training set score: {:.4f}'.format(clf_en.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(clf_en.score(X_test, y_test)))
```

```
Training set score: 0.9833
Test set score: 0.9667
```

We can see that the training-set score and test-set score is same as above. The training-set accuracy score is 0.0.9833 while the test-set accuracy to be 0.9667. These two values are quite comparable. So, there is no sign of overfitting.

Random Forest Classifier

In [32]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matri
```

In [33]:
```
## import Dataset
df=pd.read_csv("C:/Users/SAGNIK SAMANTA/OneDrive/Desktop/Datasets/Iris.csv")
df.head()
```

Out[33]:

	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

In [34]:
```
df.columns
```

Out[34]:
```
Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [35]:
```
## Separate the independent and dependent variables using the slicing method.
X=df.drop("Species",axis=1)
y=df[['Species']]
```

In [36]:
```
## Split the data into training and testing sets.
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=100)
```

In [37]:
```
## Build a Random Forest Classifier Model
forest=RandomForestClassifier(n_estimators=10,criterion="entropy",random_state=0)
forest.fit(X_train,y_train)
```

```
C:\Users\SAGNIK SAMANTA\anaconda3\Lib\site-packages\sklearn\base.py:1151: DataCon
versionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

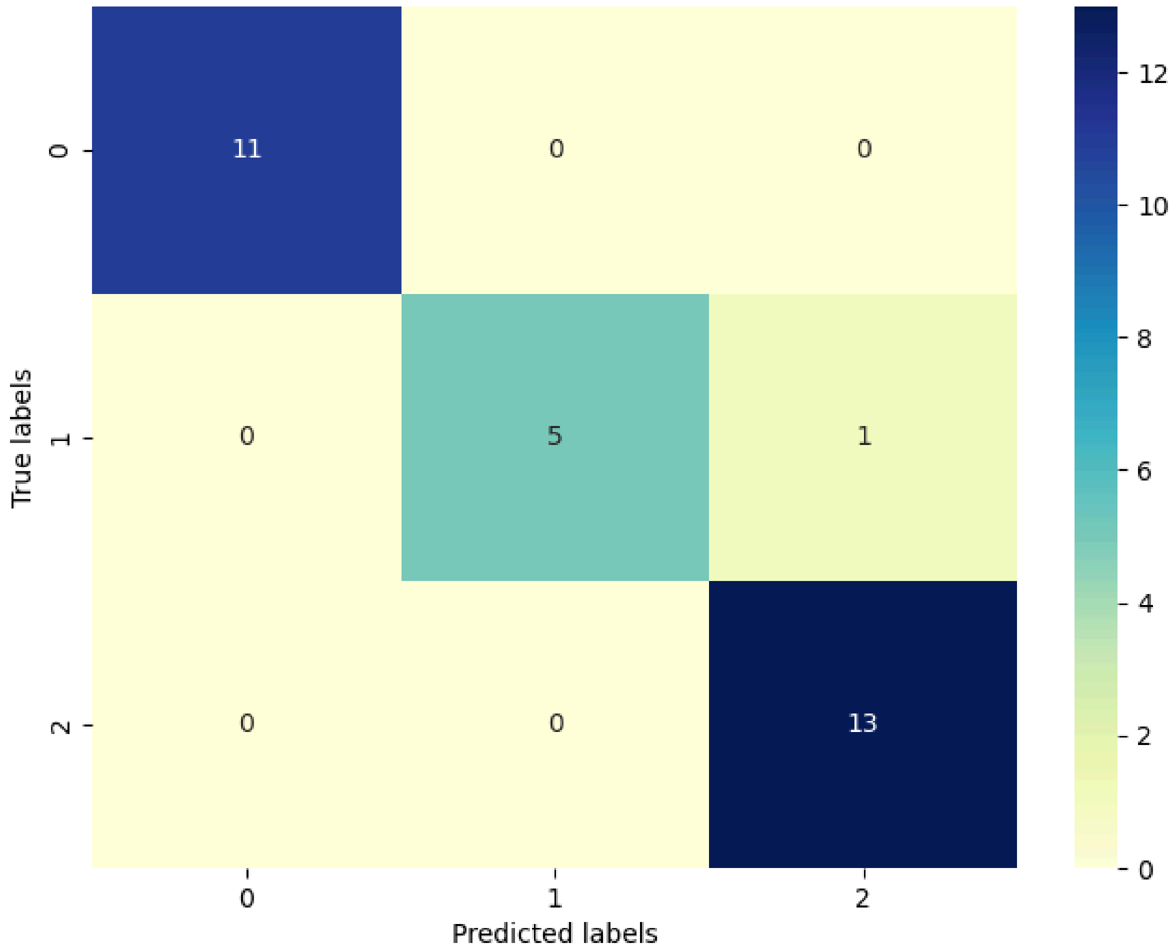
Out[37]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

In [38]:
```
## Calculate the accuracy of the model
Prediction=forest.predict(X_test)
print('The accuracy of the Random Forest is',accuracy_score(Prediction,y_test))
print("Classification Report:")
print(classification_report(y_test, Prediction))
print("Confusion Matrix:")
print(confusion_matrix(y_test, Prediction))
```

```
The accuracy of the Random Forest is 0.9666666666666667
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.83      0.91         6
 Iris-virginica       0.93      1.00      0.96        13

       accuracy                           0.97        30
      macro avg       0.98      0.94      0.96        30
   weighted avg       0.97      0.97      0.97        30

Confusion Matrix:
[[11  0  0]
 [ 0  5  1]
 [ 0  0 13]]
```

Conclusion :

In [39]:
```
# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, Prediction), annot=True, cmap="YlGnBu")
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.show()
```



Our prediction model shows that there is an excellent accuracy score of 96.66666666666667 percent.

- Precision : Out of all the flower species that the model predicted would get classified into three categories,the model perdicted the outcomes as 100% as Setosa,100% for Versicolor and 93% for Virginica
- Recall : Out of all the flower species that actually got classified into three categories,the model perdicted the outcomes correctly as 100% as Setosa,83% for Versicolor and 100% for Virginica
- F1-Score : This value is calculated as = 2 * (Precision * Recall) / (Precision + Recall)
  - Since the value is very close to 1 , therefore we can state that our model correctly classified flowers into three flower species
- Support : These values simply tell us how many flowers belonged to each class in the test dataset.We can see that among the flowers in the test dataset, 11 flowers belong to Setosa, 13 flowers belongs to verginica and 6 flowers belong to versicolor.

Support Vector Machine

In [40]:
```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matri
```

In [41]:
```
## import Dataset
df=pd.read_csv("C:/Users/SAGNIK SAMANTA/OneDrive/Desktop/Datasets/Iris.csv")
df.head()
```

Out[41]:

	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

In [42]:
```
df.columns
```

Out[42]:
```
Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [43]:
```
## Separate the independent and dependent variables using the slicing method.
X=df.drop("Species",axis=1)
y=df[['Species']]
```

In [44]:
```
## Split the data into training and testing sets.
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=100)
```

In [45]:
```
y_train=np.ravel(y_train)
y_test=np.ravel(y_test)
```

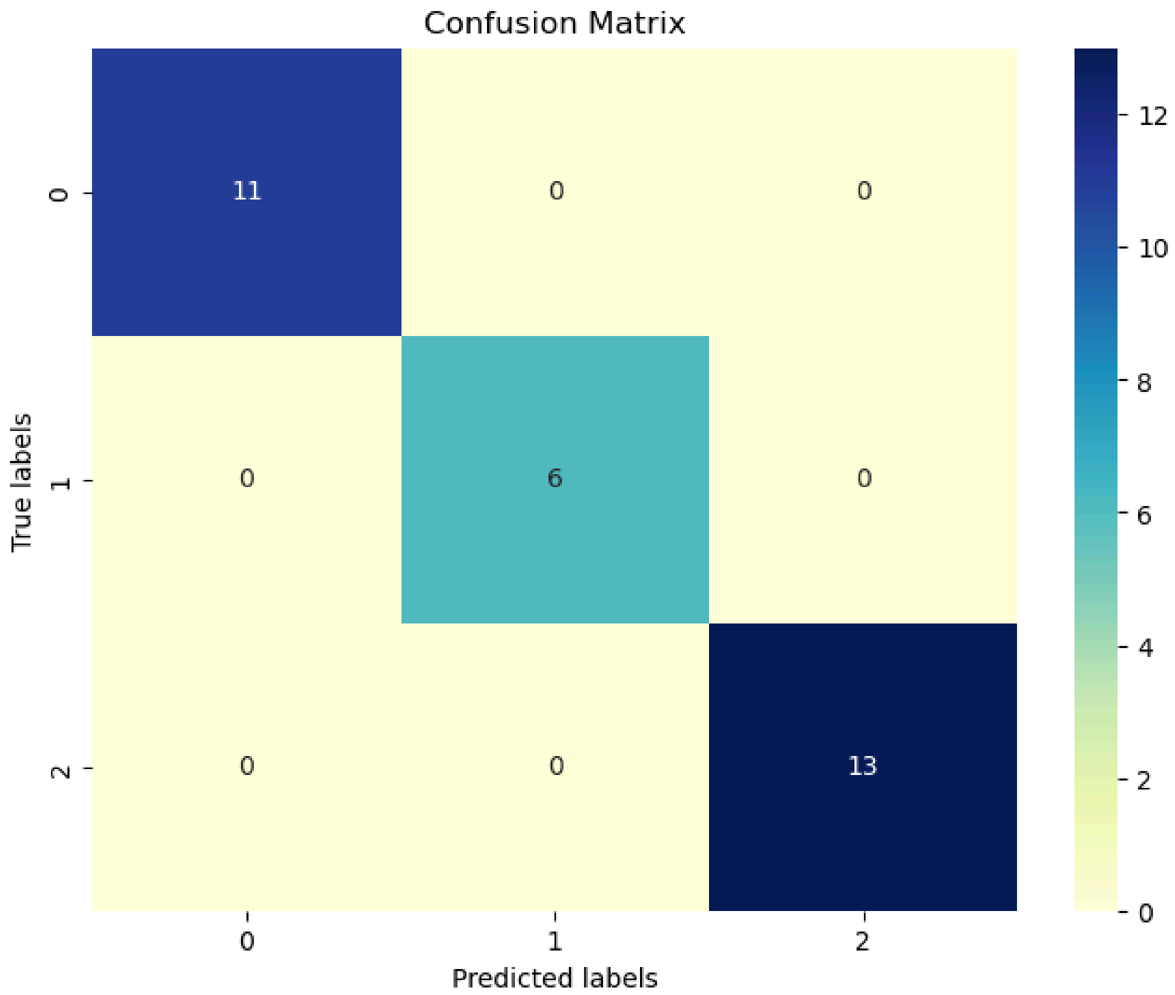
In [46]:
```
## Model Building
svc=SVC()
svc.fit(X_train,y_train)
```

Out[46]:
```
▾ SVC
SVC()
```

In [47]:
```
## Checking the accuracy of the Model
Prediction=svc.predict(X_test)
print('The accuracy of the SVC is',accuracy_score(Prediction,y_test))
print("Classification Report:")
print(classification_report(y_test, Prediction))
print("Confusion Matrix:")
print(confusion_matrix(y_test, Prediction))
```

```
The accuracy of the SVC is 1.0
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00         6
 Iris-virginica       1.00      1.00      1.00        13

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30

Confusion Matrix:
[[11  0  0]
 [ 0  6  0]
 [ 0  0 13]]
```

In [48]:
```
# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, Prediction), annot=True, cmap="YlGnBu")
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```



Conclusion :

Our prediction model shows that there is an excellent accuracy score of 100 percent.

- Precision : Out of all the flower species that the model predicted would get classified into three categories,the model perdicted the outcomes as 100% as Setosa,100% for Versicolor and 100% for Virginica
- Recall : Out of all the flower species that actually got classified into three categories,the model perdicted the outcomes correctly as 100% as Setosa,100% for Versicolor and 100% for Virginica
- F1-Score : This value is calculated as = 2 * (Precision * Recall) / (Precision + Recall)
  - Since the value is 1 , therefore we can state that our model correctly classified flowers into three flower species
- Support : These values simply tell us how many flowers belonged to each class in the test dataset.We can see that among the flowers in the test dataset, 11 flowers belong to Setosa, 13 flowers belongs to verginica and 6 flowers belong to versicolor.

Logistic Regression

In [49]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [50]:
```
## import Dataset
df=pd.read_csv("C:/Users/SAGNIK SAMANTA/OneDrive/Desktop/Datasets/Iris.csv")
df.head()
```

Out[50]:

	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 Inspection

In [51]:
```
## Dimension of the Dataset
print("Shape of the data frame: ",df.shape)
```

```
Shape of the data frame:  (150, 5)
```

In [52]:
```
## Checking for Missing Values
print("Total null values: ",df.isna().sum().sum())
```

```
Total null values:  0
```

In [53]:
```
## Checking for Duplicate Values
print("Duplicate values: ",df.duplicated().sum() )
```

```
Duplicate values:  3
```

Removing the Duplicated Values

In [54]:
```
df.drop_duplicates(inplace=True)
print("Shape of the data frame: ",df.shape)
print("\n")
print("Species categories with its count \n",df["Species"].value_counts())
```

```
Shape of the data frame:  (147, 5)


Species categories with its count 
 Species
Iris-versicolor    50
Iris-virginica     49
Iris-setosa        48
Name: count, dtype: int64
```

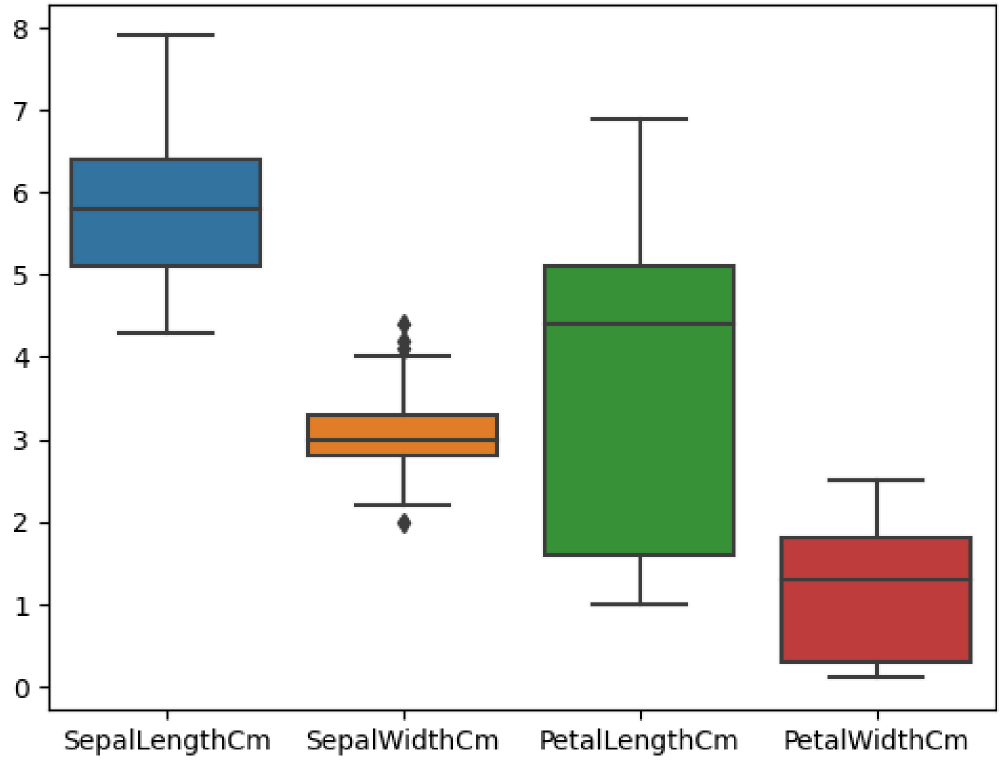
In [55]:
```
df.describe()
```

Out[55]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	147.000000	147.000000	147.000000	147.000000
mean	5.856463	3.055782	3.780272	1.208844
std	0.829100	0.437009	1.759111	0.757874
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [56]:
```
## Boxplot
sns.boxplot(data=df)
```

Out[56]: <Axes: >



from the above box plot it's clear

- sepal_width has outliers and it's right skewed
- petal_length and petal_with are negatively skewed
- sepal_length is symmetrical

In [57]:
```
df.columns
```

Out[57]:
```
Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [58]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder=LabelEncoder()
df["Species"]=label_encoder.fit_transform(df["Species"])
print(df.head(10))
print("\n")
print(df["Species"].value_counts())
```

```
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
0            5.1           3.5            1.4           0.2        0
1            4.9           3.0            1.4           0.2        0
2            4.7           3.2            1.3           0.2        0
3            4.6           3.1            1.5           0.2        0
4            5.0           3.6            1.4           0.2        0
5            5.4           3.9            1.7           0.4        0
6            4.6           3.4            1.4           0.3        0
7            5.0           3.4            1.5           0.2        0
8            4.4           2.9            1.4           0.2        0
9            4.9           3.1            1.5           0.1        0


Species
1    50
2    49
0    48
Name: count, dtype: int64
```

In [59]:
```
## Separate the independent and dependent variables using the slicing method.
X=df.drop("Species",axis=1)
y=df[['Species']]
```

In [60]:
```
## Split the data into training and testing sets.
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=100)
```

In [61]:
```
## Model Building
log_reg = LogisticRegression()
log_reg.fit(X_train,y_train)
```

```
C:\Users\SAGNIK SAMANTA\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1
184: DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\SAGNIK SAMANTA\anaconda3\Lib\site-packages\sklearn\linear_model\_logisti
c.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-le
arn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
(https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[61]:
```
▾ LogisticRegression
LogisticRegression()
```

In [62]:
```
## Checking the accuracy of the Model
Prediction=log_reg.predict(X_test)
print('The accuracy of the Logistic Regression is',accuracy_score(Prediction,y_tes
```

```
The accuracy of the Logistic Regression is 0.9666666666666667
```

In [63]:
```
## Confusion Matrix
cm=confusion_matrix(y_test, Prediction)
print("Confusion Matrix:\n ",cm)
print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion Matrix:
  [[ 9  0  0]
 [ 0  7  0]
 [ 0  1 13]]

True Positives(TP) =  9

True Negatives(TN) =  7

False Positives(FP) =  0

False Negatives(FN) =  0
```

In [64]:
```
## Classification Report
print("Classification Report:")
print(classification_report(y_test, Prediction))
print("Confusion Matrix:")
print(confusion_matrix(y_test, Prediction))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       0.88      1.00      0.93         7
           2       1.00      0.93      0.96        14

    accuracy                           0.97        30
   macro avg       0.96      0.98      0.97        30
weighted avg       0.97      0.97      0.97        30

Confusion Matrix:
[[ 9  0  0]
 [ 0  7  0]
 [ 0  1 13]]
```

Conclusion :

- Precision : Out of all the flower species that the model predicted would get classified into three categories,the model perdicted the outcomes as 100% as Setosa,88% for Versicolor and 100% for Virginica
- Recall : Out of all the flower species that actually got classified into three categories,the model perdicted the outcomes correctly as 100% as Setosa,100% for Versicolor and 93% for Virginica

- F1-Score : This value is calculated as = 2 * (Precision * Recall) / (Precision + Recall)
    - Since the value is very close to 1 , therefore we can state that our model correctly classified flowers into three flower species
- Support : These values simply tell us how many flowers belonged to each class in the test dataset.We can see that among the flowers in the test dataset, 9 flowers belong to Setosa, 14 flowers belongs to verginica and 7 flowers belong to versicolor.

In [ ]:

```

```