LetsGrowMore Datascience Internship

Beginner Level -TASK 2 Prediction using Decision Tree Algorithm:

BY Ashwinraj G

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import sklearn.metrics as sm
import seaborn as sns
import matplotlib. pyplot as mt
%matplotlib inline
import sklearn.datasets as datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import plot_tree
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix,classification_report
from sklearn import preprocessing
from sklearn.tree import DecisionTreeClassifier,export_graphviz
from sklearn import tree
```

In [2]:
```
iris_data =datasets.load_iris()
iris_df=pd.DataFrame(iris_data.data,columns=iris_data.feature_names)
iris_df
```

Out[2]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [3]:
```
df=pd.read_csv('IRIS.csv')
df
```

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
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

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
```

In [5]:
```
df.describe()
```

Out[5]:

	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 [6]:
```
iris_data.feature_names
```

Out[6]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [7]:
```
iris_data.target_names
```

Out[7]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [8]:
```
iris_data.target
```

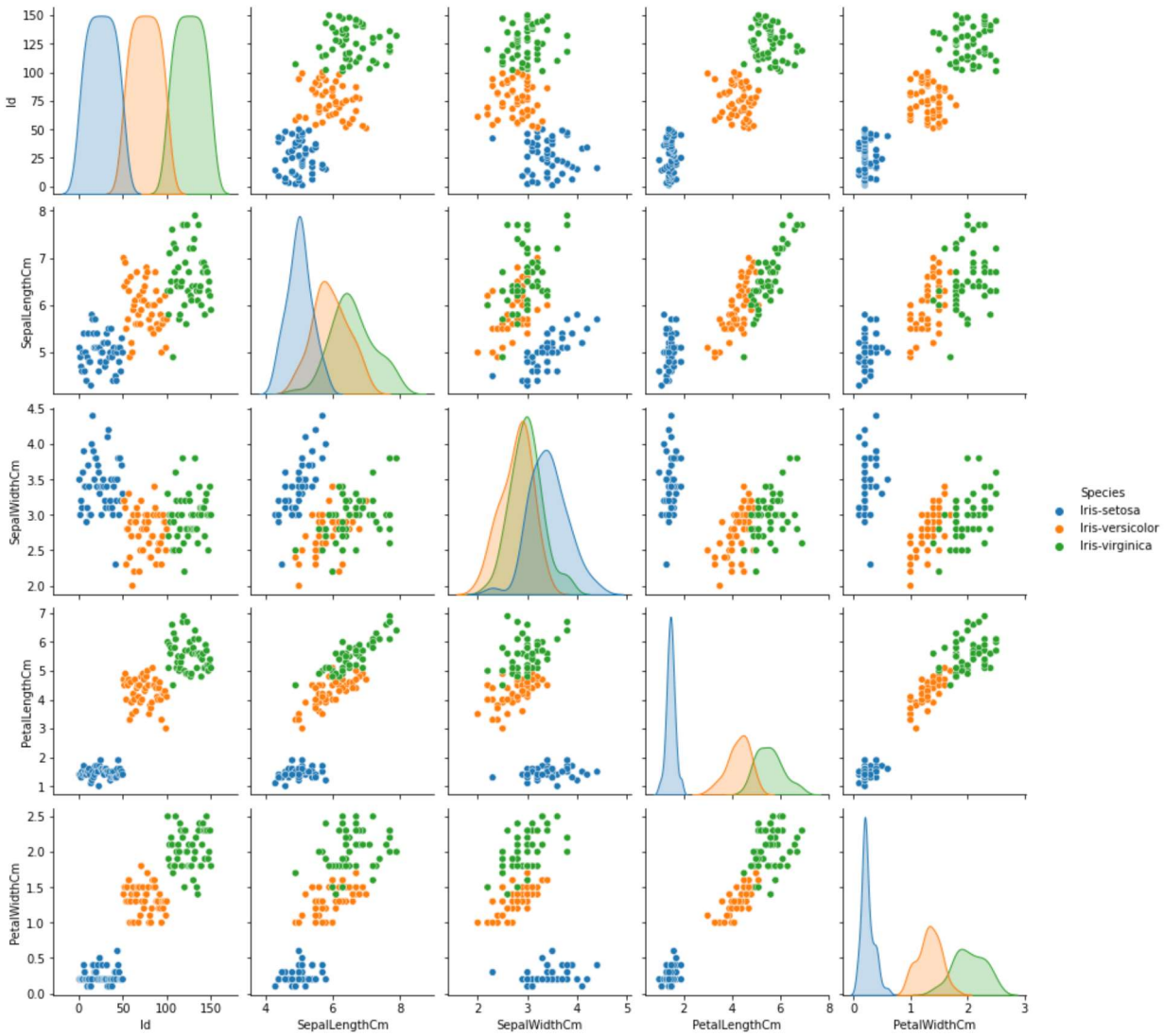
Out[8]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [9]:
```
iris_df.isnull().sum()
```

Out[9]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

Visialize Dataset

In [11]:
```
import matplotlib.pyplot as plt
sns.pairplot(df,hue='Species')
plt.show()
```



In [13]:
```
iris=pd.read_csv("C:/Users/gashw/IRIS.csv")
iris
```

Out[13]:

	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
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

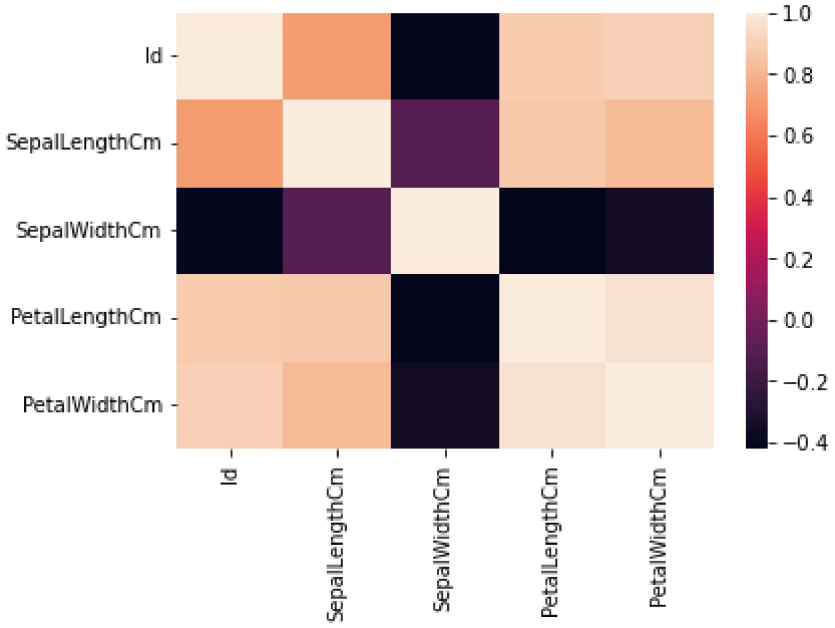
In [14]:
```
df.corr()
```

Out[14]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.899759	0.817954	-0.356544	0.962757	1.000000

In [15]:
```
sns.heatmap(df.corr())
```

Out[15]:
```
<AxesSubplot:>
```



Preparation of Data

In [17]:
```
y=iris.iloc[:,:-1].values
z=iris['Species']
```

In [18]:
```
z
```

Out[18]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

In [19]:
```
y
```

Out[19]:
```
array([[1.00e+00, 5.10e+00, 3.50e+00, 1.40e+00, 2.00e-01],
       [2.00e+00, 4.90e+00, 3.00e+00, 1.40e+00, 2.00e-01],
       [3.00e+00, 4.70e+00, 3.20e+00, 1.30e+00, 2.00e-01],
       [4.00e+00, 4.60e+00, 3.10e+00, 1.50e+00, 2.00e-01],
       [5.00e+00, 5.00e+00, 3.60e+00, 1.40e+00, 2.00e-01],
       [6.00e+00, 5.40e+00, 3.90e+00, 1.70e+00, 4.00e-01],
       [7.00e+00, 4.60e+00, 3.40e+00, 1.40e+00, 3.00e-01],
       [8.00e+00, 5.00e+00, 3.40e+00, 1.50e+00, 2.00e-01],
       [9.00e+00, 4.40e+00, 2.90e+00, 1.40e+00, 2.00e-01],
       [1.00e+01, 4.90e+00, 3.10e+00, 1.50e+00, 1.00e-01],
       [1.10e+01, 5.40e+00, 3.70e+00, 1.50e+00, 2.00e-01],
       [1.20e+01, 4.80e+00, 3.40e+00, 1.60e+00, 2.00e-01],
       [1.30e+01, 4.80e+00, 3.00e+00, 1.40e+00, 1.00e-01],
       [1.40e+01, 4.30e+00, 3.00e+00, 1.10e+00, 1.00e-01],
       [1.50e+01, 5.80e+00, 4.00e+00, 1.20e+00, 2.00e-01],
       [1.60e+01, 5.70e+00, 4.40e+00, 1.50e+00, 4.00e-01],
       [1.70e+01, 5.40e+00, 3.90e+00, 1.30e+00, 4.00e-01],
       [1.80e+01, 5.10e+00, 3.50e+00, 1.40e+00, 3.00e-01],
       [1.90e+01, 5.70e+00, 3.80e+00, 1.70e+00, 3.00e-01],
       [2.00e+01, 5.10e+00, 3.80e+00, 1.50e+00, 3.00e-01],
       [2.10e+01, 5.40e+00, 3.40e+00, 1.70e+00, 2.00e-01],
       [2.20e+01, 5.10e+00, 3.70e+00, 1.50e+00, 4.00e-01],
       [2.30e+01, 4.60e+00, 3.60e+00, 1.00e+00, 2.00e-01],
       [2.40e+01, 5.10e+00, 3.30e+00, 1.70e+00, 5.00e-01],
       [2.50e+01, 4.80e+00, 3.40e+00, 1.90e+00, 2.00e-01],
       [2.60e+01, 5.00e+00, 3.00e+00, 1.60e+00, 2.00e-01],
       [2.70e+01, 5.00e+00, 3.40e+00, 1.60e+00, 4.00e-01],
       [2.80e+01, 5.20e+00, 3.50e+00, 1.50e+00, 2.00e-01],
       [2.90e+01, 5.20e+00, 3.40e+00, 1.40e+00, 2.00e-01],
       [3.00e+01, 4.70e+00, 3.20e+00, 1.60e+00, 2.00e-01],
       [3.10e+01, 4.80e+00, 3.10e+00, 1.60e+00, 2.00e-01],
       [3.20e+01, 5.40e+00, 3.40e+00, 1.50e+00, 4.00e-01],
       [3.30e+01, 5.20e+00, 4.10e+00, 1.50e+00, 1.00e-01],
       [3.40e+01, 5.50e+00, 4.20e+00, 1.40e+00, 2.00e-01],
       [3.50e+01, 4.90e+00, 3.10e+00, 1.50e+00, 1.00e-01],
       [3.60e+01, 5.00e+00, 3.20e+00, 1.20e+00, 2.00e-01],
       [3.70e+01, 5.50e+00, 3.50e+00, 1.30e+00, 2.00e-01],
       [3.80e+01, 4.90e+00, 3.10e+00, 1.50e+00, 1.00e-01],
       [3.90e+01, 4.40e+00, 3.00e+00, 1.30e+00, 2.00e-01],
       [4.00e+01, 5.10e+00, 3.40e+00, 1.50e+00, 2.00e-01],
       [4.10e+01, 5.00e+00, 3.50e+00, 1.30e+00, 3.00e-01],
       [4.20e+01, 4.50e+00, 2.30e+00, 1.30e+00, 3.00e-01],
       [4.30e+01, 4.40e+00, 3.20e+00, 1.30e+00, 2.00e-01],
       [4.40e+01, 5.00e+00, 3.50e+00, 1.60e+00, 6.00e-01],
       [4.50e+01, 5.10e+00, 3.80e+00, 1.90e+00, 4.00e-01],
       [4.60e+01, 4.80e+00, 3.00e+00, 1.40e+00, 3.00e-01],
       [4.70e+01, 5.10e+00, 3.80e+00, 1.60e+00, 2.00e-01],
       [4.80e+01, 4.60e+00, 3.20e+00, 1.40e+00, 2.00e-01],
       [4.90e+01, 5.30e+00, 3.70e+00, 1.50e+00, 2.00e-01],
       [5.00e+01, 5.00e+00, 3.30e+00, 1.40e+00, 2.00e-01],
       [5.10e+01, 7.00e+00, 3.20e+00, 4.70e+00, 1.40e+00],
       [5.20e+01, 6.40e+00, 3.20e+00, 4.50e+00, 1.50e+00],
       [5.30e+01, 6.90e+00, 3.10e+00, 4.90e+00, 1.50e+00],
       [5.40e+01, 5.50e+00, 2.30e+00, 4.00e+00, 1.30e+00],
       [5.50e+01, 6.50e+00, 2.80e+00, 4.60e+00, 1.50e+00],
       [5.60e+01, 5.70e+00, 2.80e+00, 4.50e+00, 1.30e+00],
       [5.70e+01, 6.30e+00, 3.30e+00, 4.70e+00, 1.60e+00],
       [5.80e+01, 4.90e+00, 2.40e+00, 3.30e+00, 1.00e+00],
       [5.90e+01, 6.60e+00, 2.90e+00, 4.60e+00, 1.30e+00],
       [6.00e+01, 5.20e+00, 2.70e+00, 3.90e+00, 1.40e+00],
```

```
[6.10e+01, 5.00e+00, 2.00e+00, 3.50e+00, 1.00e+00],
[6.20e+01, 5.90e+00, 3.00e+00, 4.20e+00, 1.50e+00],
[6.30e+01, 6.00e+00, 2.20e+00, 4.00e+00, 1.00e+00],
[6.40e+01, 6.10e+00, 2.90e+00, 4.70e+00, 1.40e+00],
[6.50e+01, 5.60e+00, 2.90e+00, 3.60e+00, 1.30e+00],
[6.60e+01, 6.70e+00, 3.10e+00, 4.40e+00, 1.40e+00],
[6.70e+01, 5.60e+00, 3.00e+00, 4.50e+00, 1.50e+00],
[6.80e+01, 5.80e+00, 2.70e+00, 4.10e+00, 1.00e+00],
[6.90e+01, 6.20e+00, 2.20e+00, 4.50e+00, 1.50e+00],
[7.00e+01, 5.60e+00, 2.50e+00, 3.90e+00, 1.10e+00],
[7.10e+01, 5.90e+00, 3.20e+00, 4.80e+00, 1.80e+00],
[7.20e+01, 6.10e+00, 2.80e+00, 4.00e+00, 1.30e+00],
[7.30e+01, 6.30e+00, 2.50e+00, 4.90e+00, 1.50e+00],
[7.40e+01, 6.10e+00, 2.80e+00, 4.70e+00, 1.20e+00],
[7.50e+01, 6.40e+00, 2.90e+00, 4.30e+00, 1.30e+00],
[7.60e+01, 6.60e+00, 3.00e+00, 4.40e+00, 1.40e+00],
[7.70e+01, 6.80e+00, 2.80e+00, 4.80e+00, 1.40e+00],
[7.80e+01, 6.70e+00, 3.00e+00, 5.00e+00, 1.70e+00],
[7.90e+01, 6.00e+00, 2.90e+00, 4.50e+00, 1.50e+00],
[8.00e+01, 5.70e+00, 2.60e+00, 3.50e+00, 1.00e+00],
[8.10e+01, 5.50e+00, 2.40e+00, 3.80e+00, 1.10e+00],
[8.20e+01, 5.50e+00, 2.40e+00, 3.70e+00, 1.00e+00],
[8.30e+01, 5.80e+00, 2.70e+00, 3.90e+00, 1.20e+00],
[8.40e+01, 6.00e+00, 2.70e+00, 5.10e+00, 1.60e+00],
[8.50e+01, 5.40e+00, 3.00e+00, 4.50e+00, 1.50e+00],
[8.60e+01, 6.00e+00, 3.40e+00, 4.50e+00, 1.60e+00],
[8.70e+01, 6.70e+00, 3.10e+00, 4.70e+00, 1.50e+00],
[8.80e+01, 6.30e+00, 2.30e+00, 4.40e+00, 1.30e+00],
[8.90e+01, 5.60e+00, 3.00e+00, 4.10e+00, 1.30e+00],
[9.00e+01, 5.50e+00, 2.50e+00, 4.00e+00, 1.30e+00],
[9.10e+01, 5.50e+00, 2.60e+00, 4.40e+00, 1.20e+00],
[9.20e+01, 6.10e+00, 3.00e+00, 4.60e+00, 1.40e+00],
[9.30e+01, 5.80e+00, 2.60e+00, 4.00e+00, 1.20e+00],
[9.40e+01, 5.00e+00, 2.30e+00, 3.30e+00, 1.00e+00],
[9.50e+01, 5.60e+00, 2.70e+00, 4.20e+00, 1.30e+00],
[9.60e+01, 5.70e+00, 3.00e+00, 4.20e+00, 1.20e+00],
[9.70e+01, 5.70e+00, 2.90e+00, 4.20e+00, 1.30e+00],
[9.80e+01, 6.20e+00, 2.90e+00, 4.30e+00, 1.30e+00],
[9.90e+01, 5.10e+00, 2.50e+00, 3.00e+00, 1.10e+00],
[1.00e+02, 5.70e+00, 2.80e+00, 4.10e+00, 1.30e+00],
[1.01e+02, 6.30e+00, 3.30e+00, 6.00e+00, 2.50e+00],
[1.02e+02, 5.80e+00, 2.70e+00, 5.10e+00, 1.90e+00],
[1.03e+02, 7.10e+00, 3.00e+00, 5.90e+00, 2.10e+00],
[1.04e+02, 6.30e+00, 2.90e+00, 5.60e+00, 1.80e+00],
[1.05e+02, 6.50e+00, 3.00e+00, 5.80e+00, 2.20e+00],
[1.06e+02, 7.60e+00, 3.00e+00, 6.60e+00, 2.10e+00],
[1.07e+02, 4.90e+00, 2.50e+00, 4.50e+00, 1.70e+00],
[1.08e+02, 7.30e+00, 2.90e+00, 6.30e+00, 1.80e+00],
[1.09e+02, 6.70e+00, 2.50e+00, 5.80e+00, 1.80e+00],
[1.10e+02, 7.20e+00, 3.60e+00, 6.10e+00, 2.50e+00],
[1.11e+02, 6.50e+00, 3.20e+00, 5.10e+00, 2.00e+00],
[1.12e+02, 6.40e+00, 2.70e+00, 5.30e+00, 1.90e+00],
[1.13e+02, 6.80e+00, 3.00e+00, 5.50e+00, 2.10e+00],
[1.14e+02, 5.70e+00, 2.50e+00, 5.00e+00, 2.00e+00],
[1.15e+02, 5.80e+00, 2.80e+00, 5.10e+00, 2.40e+00],
[1.16e+02, 6.40e+00, 3.20e+00, 5.30e+00, 2.30e+00],
[1.17e+02, 6.50e+00, 3.00e+00, 5.50e+00, 1.80e+00],
[1.18e+02, 7.70e+00, 3.80e+00, 6.70e+00, 2.20e+00],
[1.19e+02, 7.70e+00, 2.60e+00, 6.90e+00, 2.30e+00],
[1.20e+02, 6.00e+00, 2.20e+00, 5.00e+00, 1.50e+00],
```

```
[1.21e+02, 6.90e+00, 3.20e+00, 5.70e+00, 2.30e+00],
[1.22e+02, 5.60e+00, 2.80e+00, 4.90e+00, 2.00e+00],
[1.23e+02, 7.70e+00, 2.80e+00, 6.70e+00, 2.00e+00],
[1.24e+02, 6.30e+00, 2.70e+00, 4.90e+00, 1.80e+00],
[1.25e+02, 6.70e+00, 3.30e+00, 5.70e+00, 2.10e+00],
[1.26e+02, 7.20e+00, 3.20e+00, 6.00e+00, 1.80e+00],
[1.27e+02, 6.20e+00, 2.80e+00, 4.80e+00, 1.80e+00],
[1.28e+02, 6.10e+00, 3.00e+00, 4.90e+00, 1.80e+00],
[1.29e+02, 6.40e+00, 2.80e+00, 5.60e+00, 2.10e+00],
[1.30e+02, 7.20e+00, 3.00e+00, 5.80e+00, 1.60e+00],
[1.31e+02, 7.40e+00, 2.80e+00, 6.10e+00, 1.90e+00],
[1.32e+02, 7.90e+00, 3.80e+00, 6.40e+00, 2.00e+00],
[1.33e+02, 6.40e+00, 2.80e+00, 5.60e+00, 2.20e+00],
[1.34e+02, 6.30e+00, 2.80e+00, 5.10e+00, 1.50e+00],
[1.35e+02, 6.10e+00, 2.60e+00, 5.60e+00, 1.40e+00],
[1.36e+02, 7.70e+00, 3.00e+00, 6.10e+00, 2.30e+00],
[1.37e+02, 6.30e+00, 3.40e+00, 5.60e+00, 2.40e+00],
[1.38e+02, 6.40e+00, 3.10e+00, 5.50e+00, 1.80e+00],
[1.39e+02, 6.00e+00, 3.00e+00, 4.80e+00, 1.80e+00],
[1.40e+02, 6.90e+00, 3.10e+00, 5.40e+00, 2.10e+00],
[1.41e+02, 6.70e+00, 3.10e+00, 5.60e+00, 2.40e+00],
[1.42e+02, 6.90e+00, 3.10e+00, 5.10e+00, 2.30e+00],
[1.43e+02, 5.80e+00, 2.70e+00, 5.10e+00, 1.90e+00],
[1.44e+02, 6.80e+00, 3.20e+00, 5.90e+00, 2.30e+00],
[1.45e+02, 6.70e+00, 3.30e+00, 5.70e+00, 2.50e+00],
[1.46e+02, 6.70e+00, 3.00e+00, 5.20e+00, 2.30e+00],
[1.47e+02, 6.30e+00, 2.50e+00, 5.00e+00, 1.90e+00],
[1.48e+02, 6.50e+00, 3.00e+00, 5.20e+00, 2.00e+00],
[1.49e+02, 6.20e+00, 3.40e+00, 5.40e+00, 2.30e+00],
[1.50e+02, 5.90e+00, 3.00e+00, 5.10e+00, 1.80e+00]])
```

In [21]:
```
y_train ,y_test ,z_train ,z_test = train_test_split(y, z, test_size=20,random_state=0)
print("Traingin split:",y_train.shape)
print("Traingin split:",z_test.shape)
```

```
Traingin split: (130, 5)
Traingin split: (20,)
```

Design and Train the Decision Tree Model

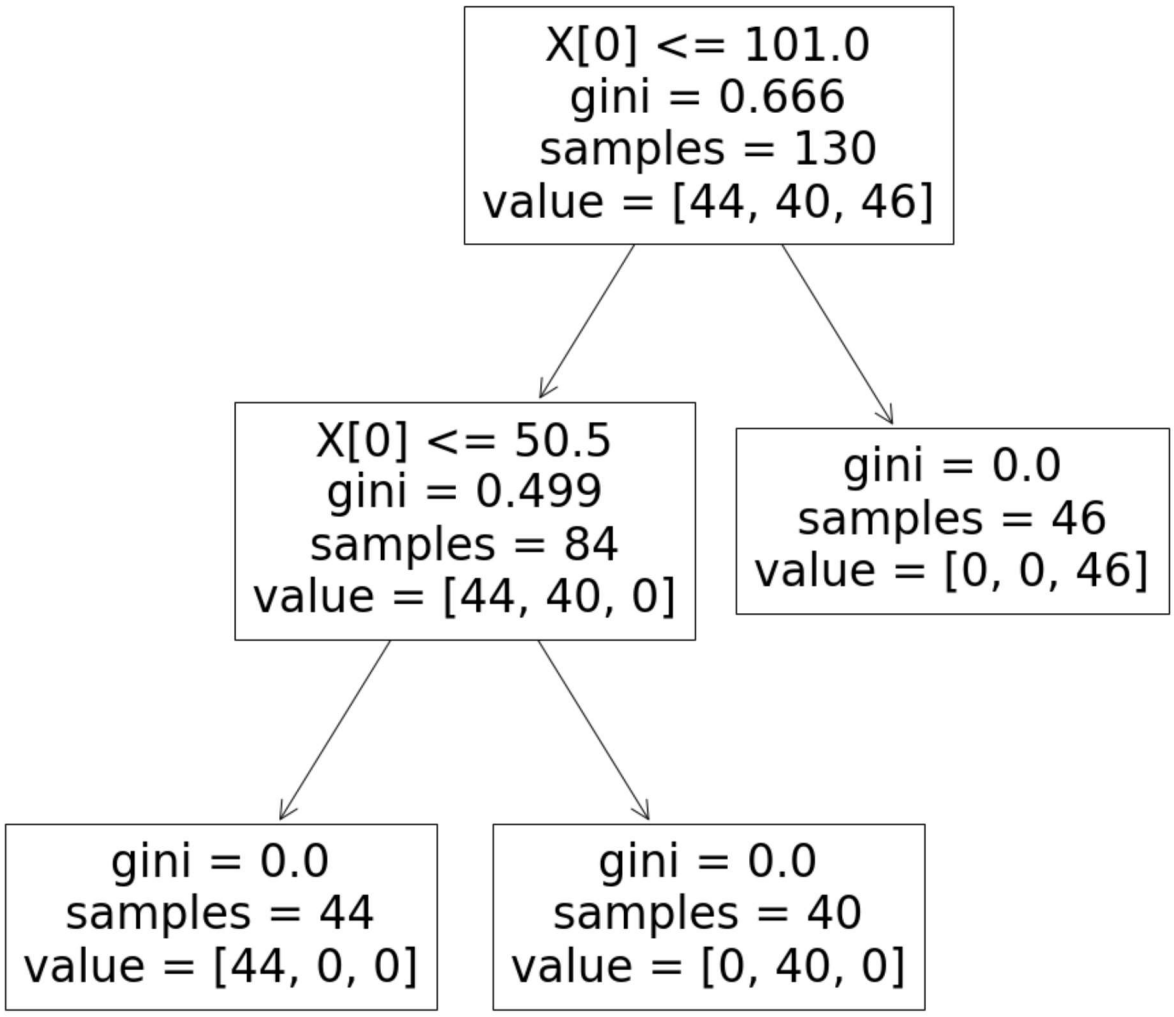
In [22]:
```
dtree = DecisionTreeClassifier()
dtree.fit(y_train,z_train)
print("Decision Tree classifier Created")
```

```
Decision Tree classifier Created
```

Visualize the Decision Tree Model

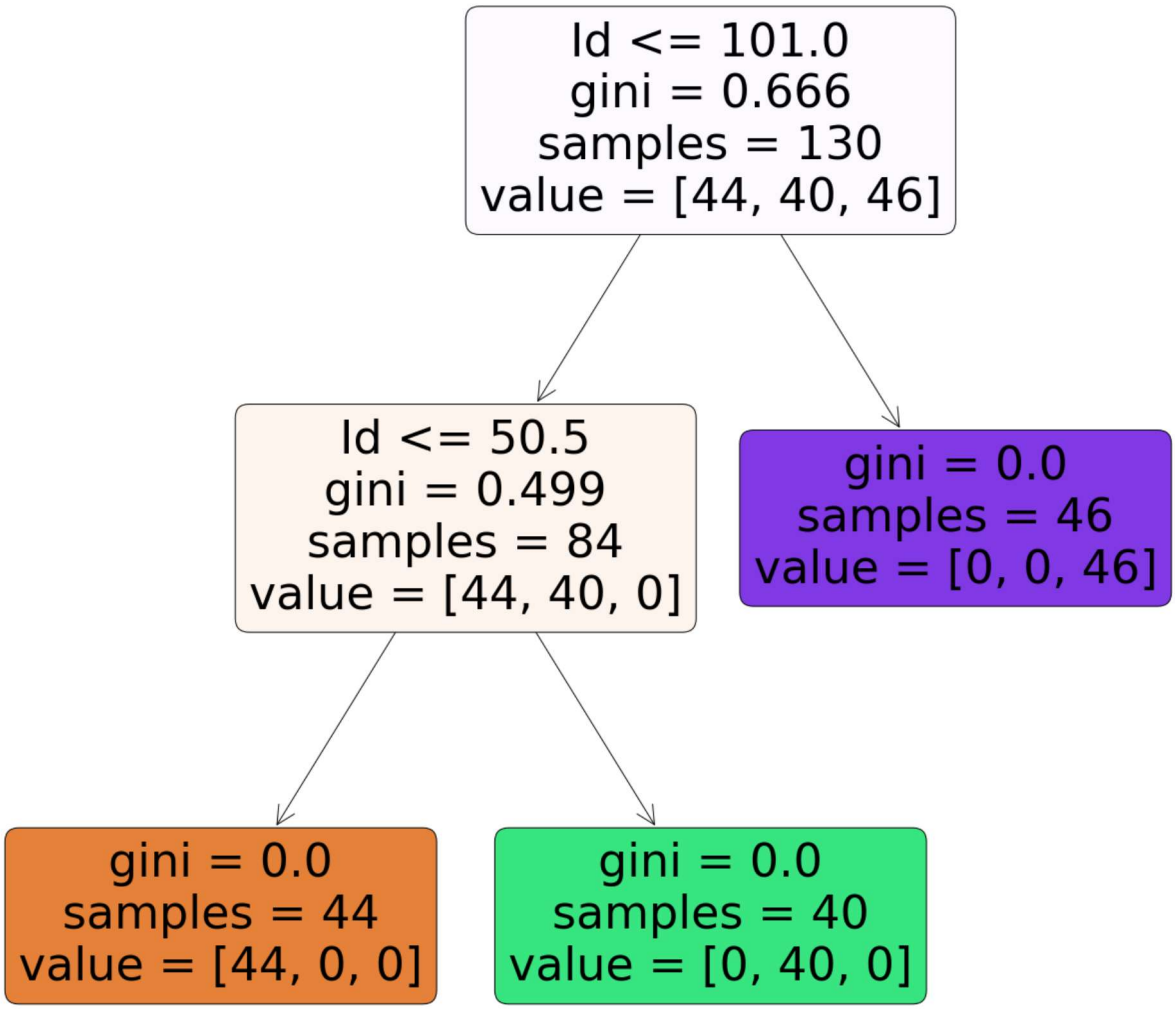
In [23]:
```
mt.figure(figsize=(16,16))
tree.plot_tree(dtree)
```

Out[23]:
```
[Text(0.6, 0.8333333333333334, 'X[0] <= 101.0\ngini = 0.666\nsamples = 130\nvalue = [44, 40, 46]'),
 Text(0.4, 0.5, 'X[0] <= 50.5\ngini = 0.499\nsamples = 84\nvalue = [44, 40, 0]'),
 Text(0.2, 0.16666666666666666, 'gini = 0.0\nsamples = 44\nvalue = [44, 0, 0]'),
 Text(0.6, 0.16666666666666666, 'gini = 0.0\nsamples = 40\nvalue = [0, 40, 0]'),
 Text(0.8, 0.5, 'gini = 0.0\nsamples = 46\nvalue = [0, 0, 46]')]
```



Visualizing the Decision Tree Model filled with colors

In [24]:
```
mt.figure(figsize=(22,22))
tree=plot_tree(dtree,feature_names=df.columns,precision=3,rounded=True,filled=True)
```



Making Prediction

In [25]:
```
z_pred= dtree.predict(y_test)
z_pred
```

Out[25]:
```
array(['Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa'],
      dtype=object)
```

In [26]:
```
label = preprocessing.LabelEncoder()
z=label.fit_transform(z_pred)
z
```

Out[26]:
```
array([2, 1, 0, 2, 0, 1, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0])
```

Evaluate the model

In [27]:
```
import sklearn.metrics as sm
print("Accuracy of the model:",sm.accuracy_score(z_test,z_pred))
```

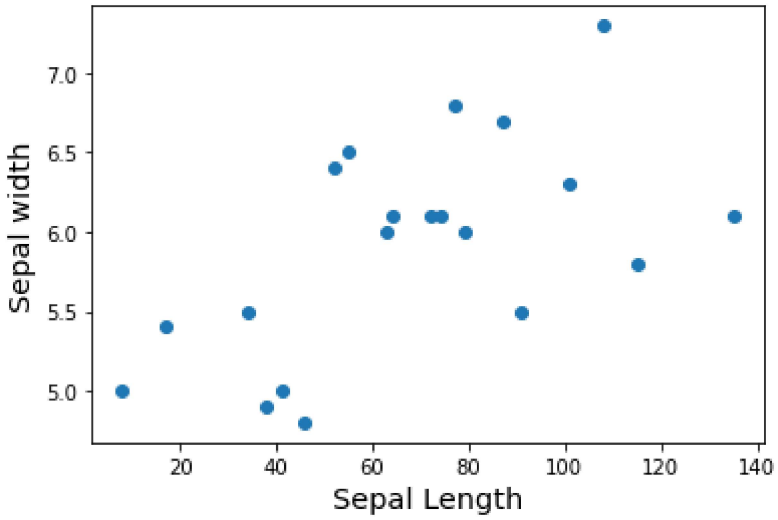
```
Accuracy of the model: 0.95
```

In [28]:
```
#actual vs predicted
result_df = pd.DataFrame({"ACTUAL":z_test,"PREDICTED":z_pred})
result_df
```

Out[28]:

	ACTUAL	PREDICTED
114	Iris-virginica	Iris-virginica
62	Iris-versicolor	Iris-versicolor
33	Iris-setosa	Iris-setosa
107	Iris-virginica	Iris-virginica
7	Iris-setosa	Iris-setosa
100	Iris-virginica	Iris-versicolor
40	Iris-setosa	Iris-setosa
86	Iris-versicolor	Iris-versicolor
76	Iris-versicolor	Iris-versicolor
71	Iris-versicolor	Iris-versicolor
134	Iris-virginica	Iris-virginica
51	Iris-versicolor	Iris-versicolor
73	Iris-versicolor	Iris-versicolor
54	Iris-versicolor	Iris-versicolor
63	Iris-versicolor	Iris-versicolor
37	Iris-setosa	Iris-setosa
78	Iris-versicolor	Iris-versicolor
90	Iris-versicolor	Iris-versicolor
45	Iris-setosa	Iris-setosa
16	Iris-setosa	Iris-setosa

In [29]:
```
plt.scatter(y_test[:,0],y_test[:,1],cmap='gist_heat')
plt.xlabel('Sepal Length',fontsize=14.5)
plt.ylabel('Sepal width',fontsize=14.5)
plt.show()
```



In [30]:
```
#confusion matrix alone
conf_matrix=confusion_matrix(z_test,z_pred)
conf_matrix
```

Out[30]:
```
array([[ 6,  0,  0],
       [ 0, 10,  0],
       [ 0,  1,  3]], dtype=int64)
```

The Decision Three Classifier is finally created and is finally visualized using graphically.

The Prediction also calculated using decision tree algorithm.

The Accuracy of the model evaluated

In [ ]: