LetsGrowMore Data Science Internship

Intermediate Level - TASK 2

Prediction using Decision Tree Algorithm:

BY RUBA ROSHINI S

Importing Libraries

In [2]:

```
#importing all the required libraries
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 [16]:

```
#Loading the Iris dataset
iris_data =datasets.load_iris()

iris_df=pd.DataFrame(iris_data.data,columns=iris_data.feature_names)

iris_df
```

Out[16]:

	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

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
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 [18]:
```
#reading the data
df=pd.read_csv('IRIS.csv')
df
```

Out[18]:

	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
...	...	...	...	...	...
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

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Float64Index: 150 entries, 3.5 to 3.0
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   petal_length  150 non-null    float64
 2   petal_width   150 non-null    float64
 3   species       150 non-null    object
dtypes: float64(3), object(1)
memory usage: 5.9+ KB
```

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

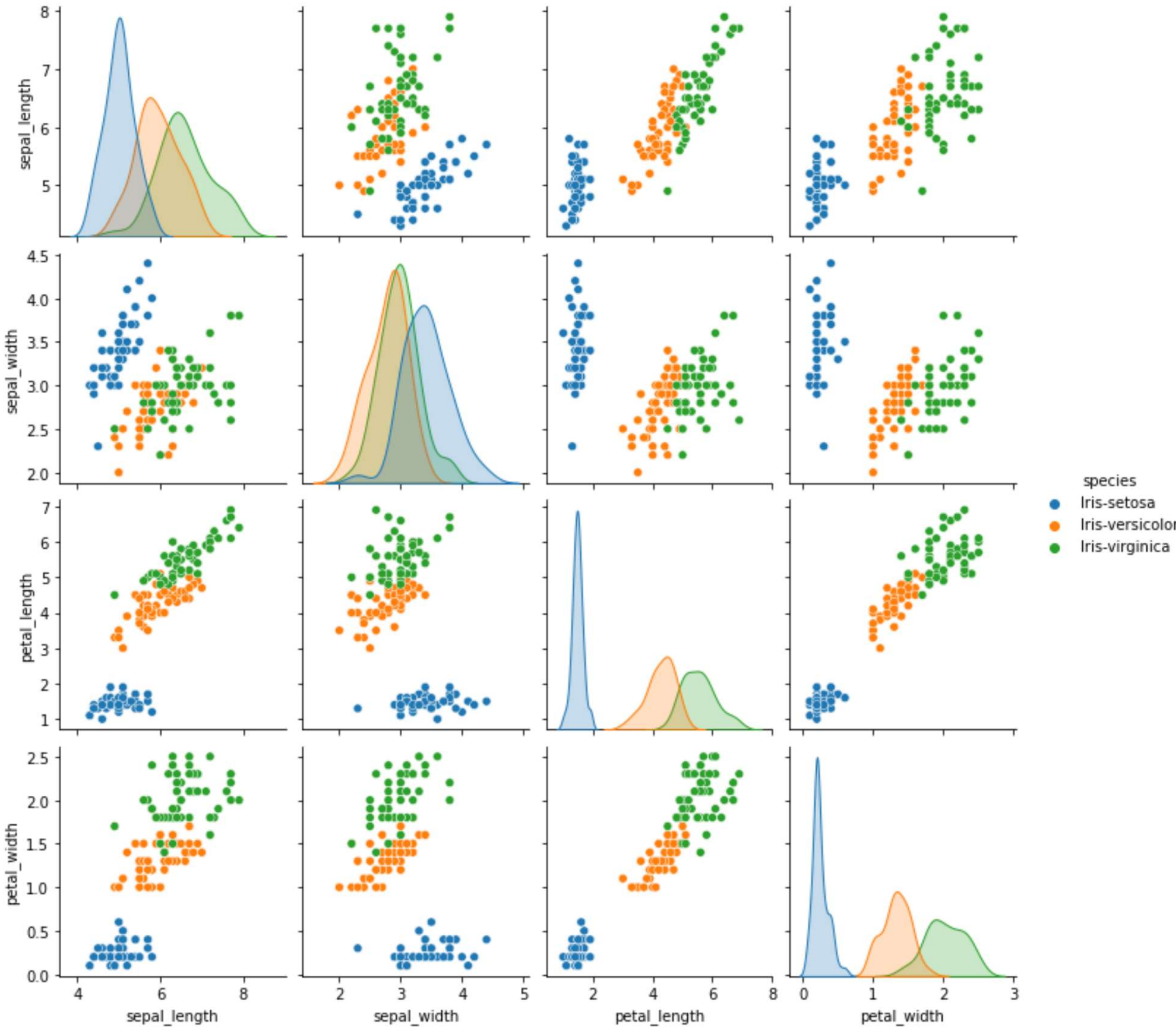
Out[5]:

	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000

	sepal_width	petal_length	petal_width
mean	3.054000	3.758667	1.198667
std	0.433594	1.764420	0.763161
min	2.000000	1.000000	0.100000
25%	2.800000	1.600000	0.300000
50%	3.000000	4.350000	1.300000
75%	3.300000	5.100000	1.800000
max	4.400000	6.900000	2.500000

In [8]:
```
iris_data.feature_names
```
Out[8]:
```
['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 [9]:
```
iris_data.target
```
Out[9]:
```
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, 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, 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, 2, 2])
```
In [10]:
```
iris_df.isnull().sum()
```
Out[10]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```
Visualize the Dataset

In [19]:
```
import matplotlib.pyplot as plt
sns.pairplot(df, hue='species')
plt.show()
```



In [22]:
```
#Reading the data from the computer location
iris=pd.read_csv("C:/Users/vidhy/LETSGRWOMRE/IRIS.csv")
iris
```

Out[22]:

	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
...	...	...	...	...	...
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

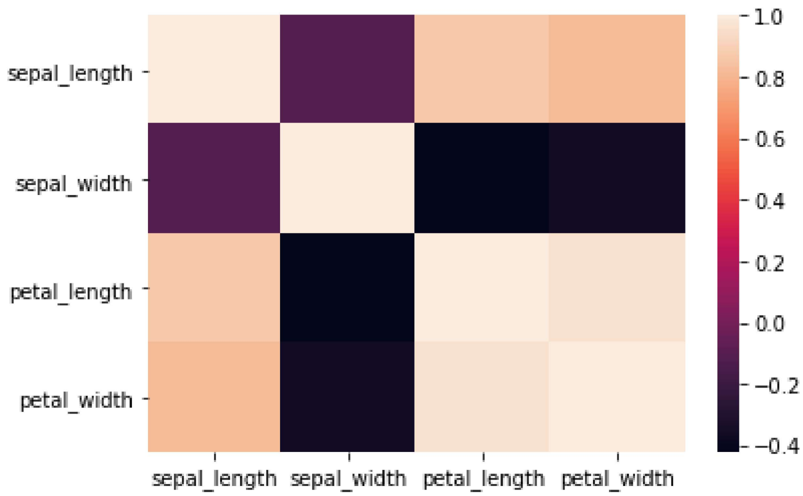
In [24]:
```
df.corr()
```

Out[24]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.109369	0.871754	0.817954
sepal_width	-0.109369	1.000000	-0.420516	-0.356544
petal_length	0.871754	-0.420516	1.000000	0.962757
petal_width	0.817954	-0.356544	0.962757	1.000000

In [25]:
```
sns.heatmap(df.corr())
```

Out[25]: <AxesSubplot:>



Prepare the data

In [33]:
```
y=iris.iloc[:,:-1].values
z=iris['species']
```

In [34]:
```
a
```

Out[34]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
```

```
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
       [6.3, 3.3, 4.7, 1.6],
       [4.9, 2.4, 3.3, 1. ],
       [6.6, 2.9, 4.6, 1.3],
       [5.2, 2.7, 3.9, 1.4],
       [5. , 2. , 3.5, 1. ],
       [5.9, 3. , 4.2, 1.5],
       [6. , 2.2, 4. , 1. ],
       [6.1, 2.9, 4.7, 1.4],
       [5.6, 2.9, 3.6, 1.3],
       [6.7, 3.1, 4.4, 1.4],
       [5.6, 3. , 4.5, 1.5],
       [5.8, 2.7, 4.1, 1. ],
       [6.2, 2.2, 4.5, 1.5],
       [5.6, 2.5, 3.9, 1.1],
       [5.9, 3.2, 4.8, 1.8],
       [6.1, 2.8, 4. , 1.3],
       [6.3, 2.5, 4.9, 1.5],
       [6.1, 2.8, 4.7, 1.2],
       [6.4, 2.9, 4.3, 1.3],
       [6.6, 3. , 4.4, 1.4],
       [6.8, 2.8, 4.8, 1.4],
       [6.7, 3. , 5. , 1.7],
       [6. , 2.9, 4.5, 1.5],
       [5.7, 2.6, 3.5, 1. ],
       [5.5, 2.4, 3.8, 1.1],
```

```
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
[7.2, 3.2, 6. , 1.8],
[6.2, 2.8, 4.8, 1.8],
[6.1, 3. , 4.9, 1.8],
[6.4, 2.8, 5.6, 2.1],
[7.2, 3. , 5.8, 1.6],
[7.4, 2.8, 6.1, 1.9],
[7.9, 3.8, 6.4, 2. ],
[6.4, 2.8, 5.6, 2.2],
[6.3, 2.8, 5.1, 1.5],
[6.1, 2.6, 5.6, 1.4],
[7.7, 3. , 6.1, 2.3],
[6.3, 3.4, 5.6, 2.4],
[6.4, 3.1, 5.5, 1.8],
[6. , 3. , 4.8, 1.8],
[6.9, 3.1, 5.4, 2.1],
[6.7, 3.1, 5.6, 2.4],
[6.9, 3.1, 5.1, 2.3],
[5.8, 2.7, 5.1, 1.9],
[6.8, 3.2, 5.9, 2.3],
[6.7, 3.3, 5.7, 2.5],
[6.7, 3. , 5.2, 2.3],
```

```
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

In [36]:
```
z
```

Out[36]:
```
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 [37]:
```
y_train ,y_test ,z_train ,z_test = train_test_split(y, z, test_size=20,random_state=250
print("Traingin split:",y_train.shape)
print("Testin spllit:",z_test.shape)
```

```
Traingin split: (130, 4)
Testin spllit: (20,)
```

Design and Train the Decision Tree Model

In [38]:
```
dtree = DecisionTreeClassifier()
dtree.fit(y_train,z_train)
print("Decision Tree classifier Created")
```

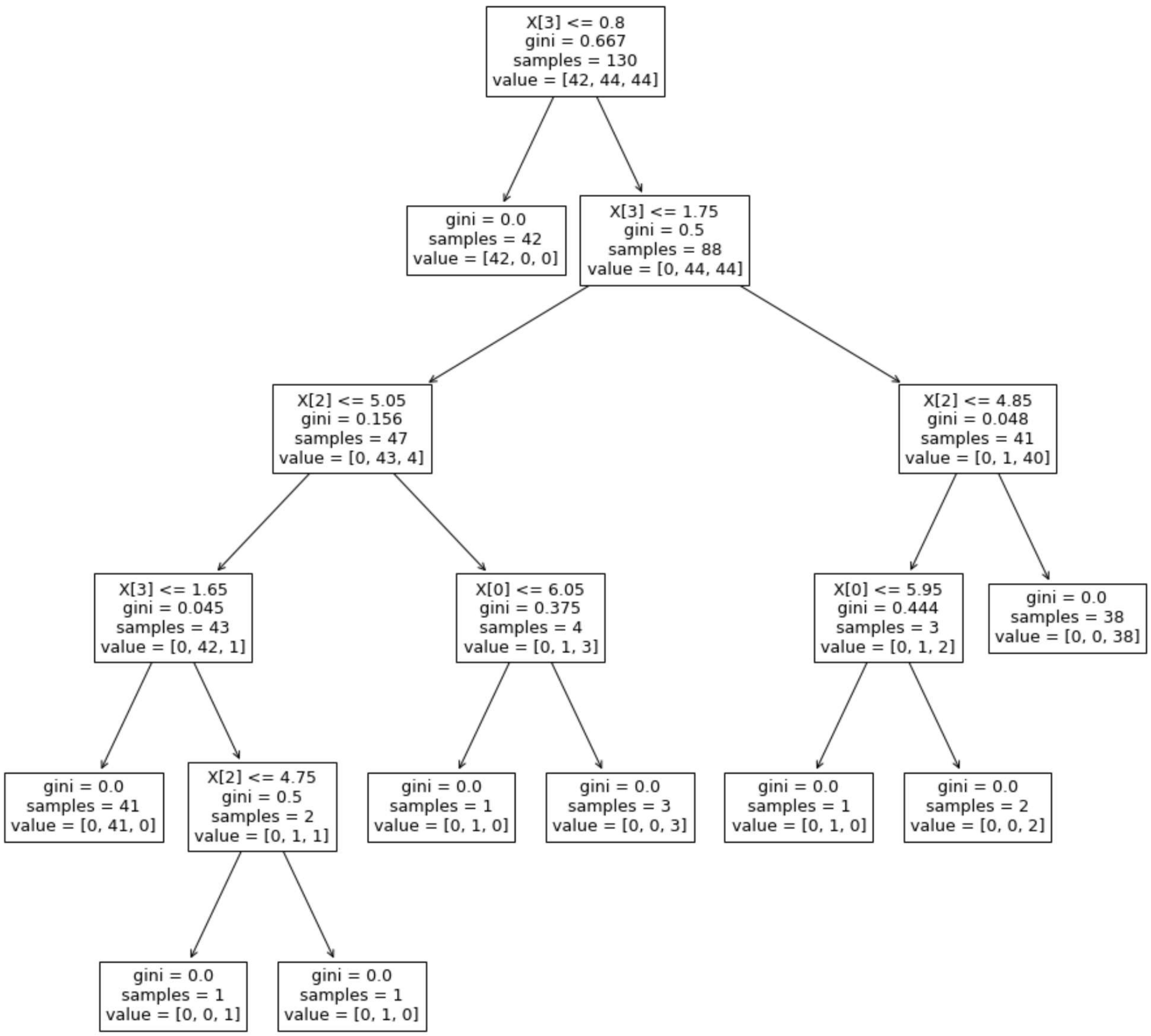
```
Decision Tree classifier Created
```

Visualize the Decision Tree Model

In [39]:
```
mt.figure(figsize=(16,16))
tree.plot_tree(dtree)
```

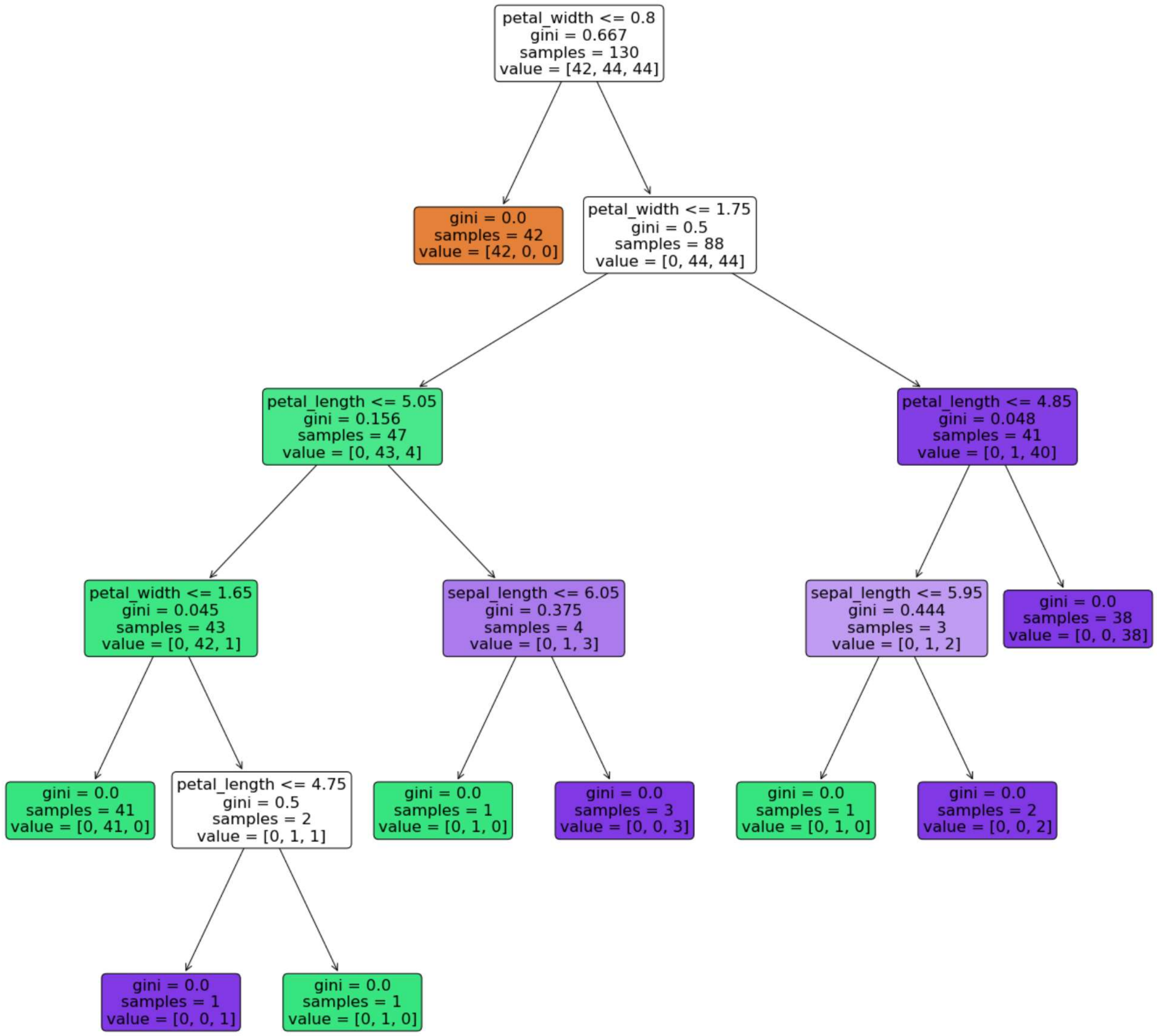
Out[39]:
```
[Text(446.4, 797.28, 'X[3] <= 0.8\ngini = 0.667\nsamples = 130\nvalue = [42, 44, 44]'),
 Text(377.7230769230769, 652.3199999999999, 'gini = 0.0\nsamples = 42\nvalue = [42, 0,
0]'),
 Text(515.0769230769231, 652.3199999999999, 'X[3] <= 1.75\ngini = 0.5\nsamples = 88\nval
ue = [0, 44, 44]'),
 Text(274.7076923076923, 507.35999999999996, 'X[2] <= 5.05\ngini = 0.156\nsamples = 47\n
value = [0, 43, 4]'),
 Text(137.35384615384615, 362.4, 'X[3] <= 1.65\ngini = 0.045\nsamples = 43\nvalue = [0,
42, 1]'),
 Text(68.67692307692307, 217.43999999999994, 'gini = 0.0\nsamples = 41\nvalue = [0, 41,
0]'),
 Text(206.03076923076924, 217.43999999999994, 'X[2] <= 4.75\ngini = 0.5\nsamples = 2\nva
lue = [0, 1, 1]'),
 Text(137.35384615384615, 72.4799999999999, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
1]'),
 Text(274.7076923076923, 72.4799999999999, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
0]'),
 Text(412.0615384615385, 362.4, 'X[0] <= 6.05\ngini = 0.375\nsamples = 4\nvalue = [0, 1,
3]'),
 Text(343.38461538461536, 217.43999999999994, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
```

```
0]'),
 Text(480.73846153846154, 217.43999999999994, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
3]'),
 Text(755.4461538461538, 507.35999999999996, 'X[2] <= 4.85\ngini = 0.048\nsamples = 41\n
value = [0, 1, 40]'),
 Text(686.7692307692307, 362.4, 'X[0] <= 5.95\ngini = 0.444\nsamples = 3\nvalue = [0, 1,
2]'),
 Text(618.0923076923077, 217.43999999999994, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
0]'),
 Text(755.4461538461538, 217.43999999999994, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
2]'),
 Text(824.123076923077, 362.4, 'gini = 0.0\nsamples = 38\nvalue = [0, 0, 38]')]
```



Visualizing the Decision Tree Model filled with colors

In [40]:
```
mt.figure(figsize=(22,22))
tree=plot_tree(dtree,feature_names=df.columns,precision=3,rounded=True,filled=True)
```



Making Prediction

In [41]:
```
z_pred= dtree.predict(y_test)
z_pred
```

Out[41]:
```
array(['Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor'], dtype=object)
```

In [42]:
```
label = preprocessing.LabelEncoder()

z = label.fit_transform(z_pred)
z
```

Out[42]:
```
array([1, 2, 0, 1, 2, 0, 1, 1, 1, 0, 0, 2, 0, 2, 1, 0, 0, 2, 0, 1])
```

Evaluate the model

In [43]:
```
import sklearn.metrics as sm
print("Accuracy of the model:",sm.accuracy_score(z_test,z_pred))
```

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

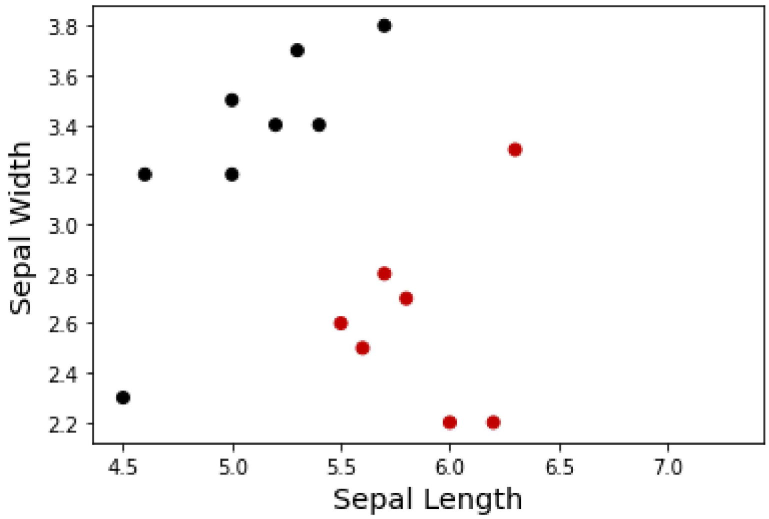
In [44]:
```
#comparing the actutal vs predicted

result_df = pd.DataFrame({"ACTUAL":z_test,"PREDICTED":z_pred})
result_df
```

Out[44]:

	ACTUAL	PREDICTED
99	Iris-versicolor	Iris-versicolor
137	Iris-virginica	Iris-virginica
20	Iris-setosa	Iris-setosa
56	Iris-versicolor	Iris-versicolor
146	Iris-virginica	Iris-virginica
40	Iris-setosa	Iris-setosa
68	Iris-versicolor	Iris-versicolor
67	Iris-versicolor	Iris-versicolor
69	Iris-versicolor	Iris-versicolor
47	Iris-setosa	Iris-setosa
28	Iris-setosa	Iris-setosa
107	Iris-virginica	Iris-virginica
41	Iris-setosa	Iris-setosa
144	Iris-virginica	Iris-virginica
119	Iris-virginica	Iris-versicolor
35	Iris-setosa	Iris-setosa
18	Iris-setosa	Iris-setosa
140	Iris-virginica	Iris-virginica
48	Iris-setosa	Iris-setosa
90	Iris-versicolor	Iris-versicolor

In [46]:
```
plt.scatter(y_test[:,0],y_test[:,1],c=z , cmap='gist_heat')
plt.xlabel('Sepal Length',fontsize=14.5)
plt.ylabel('Sepal Width',fontsize=14.5)
plt.show()
```



In [47]:
```
print(classification_report (z_test, z_pred))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         8
Iris-versicolor       0.86      1.00      0.92         6
 Iris-virginica       1.00      0.83      0.91         6

       accuracy                           0.95        20
      macro avg       0.95      0.94      0.94        20
   weighted avg       0.96      0.95      0.95        20
```

In [48]:
```
#confusion matrix alone
conf_matrix=confusion_matrix(z_test,z_pred)
conf_matrix
```

Out[48]:
```
array([[8, 0, 0],
       [0, 6, 0],
       [0, 1, 5]], dtype=int64)
```

The Decision Tree Classifier is finally created and is finally visaulized graphically.

The Prediction also calculated using decision tree algorithm.

The Accuracy of the model evaluated.