LetsGrowMore Datascience Internship

Beginner Level -TASK 2 Prediction using

Decision Tree Algorithm:

BY Ashwinraj G

Import Libraries

In [5]:

```
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 [6]:

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

Out[6]:

	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 [7]:
```
df=pd.read_csv('IRIS.csv')
df
```

Out[7]:

	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 [8]:
```
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: 5.3+ KB
```

In [9]:
```
df.describe()
```

Out[9]:

	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

	sepal_length	sepal_width	petal_length	petal_width
max	7.900000	4.400000	6.900000	2.500000

In [11]:
```
iris_data.feature_names
```

Out[11]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [12]:
```
iris_data.target_names
```

Out[12]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [13]:
```
iris_data.target
```

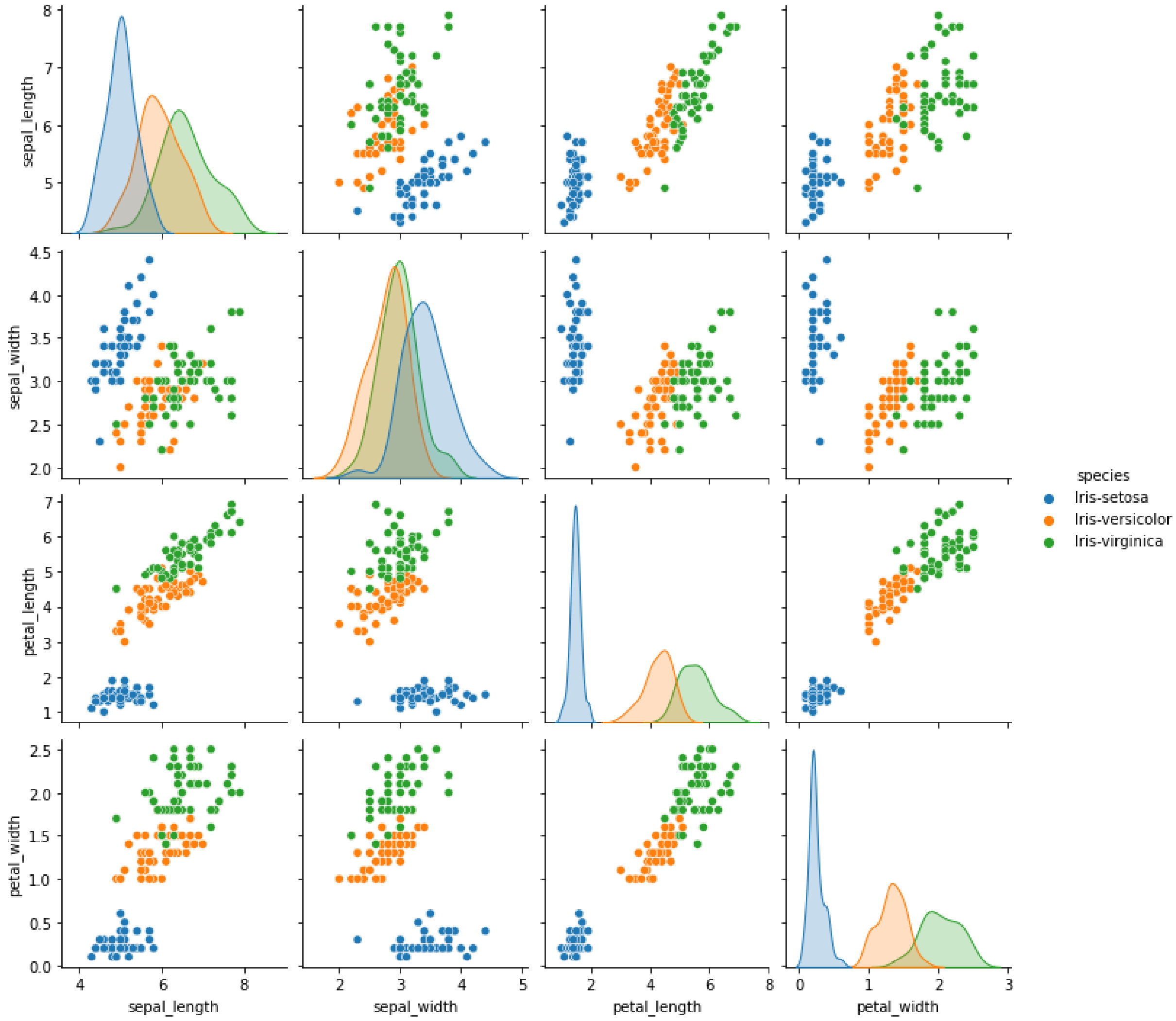
Out[13]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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 [14]:
```
iris_df.isnull().sum()
```

Out[14]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

Visialize Dataset

In [15]:
```
import matplotlib.pyplot as plt
sns.pairplot(df,hue='species')
plt.show()
```



In [24]:
```
iris=pd.read_csv("C:/Users/91701/IRIS.csv")
iris
```

Out[24]:

	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 [17]:
```
df.corr()
```

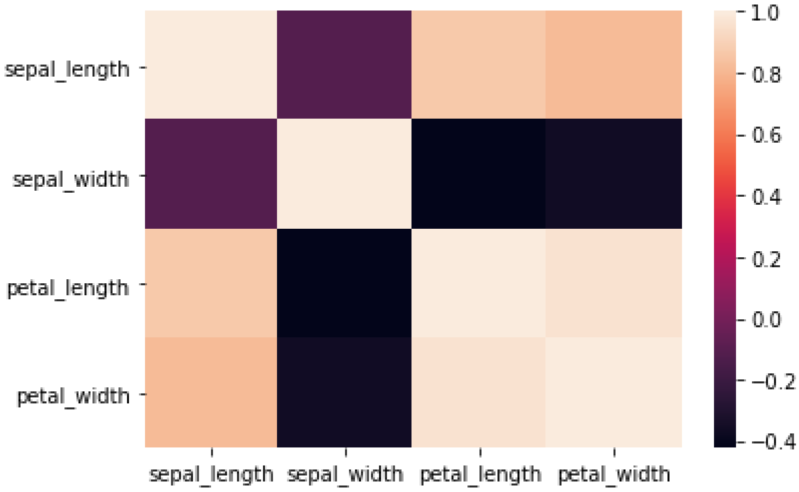
Out[17]:

	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 [18]:

```
sns.heatmap(df.corr())
```

Out[18]: <AxesSubplot:>



Preparation of Data

In [29]:

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

In [31]:

```
z
```

Out[31]:

```
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 [35]:

```
y
```

Out[35]:

```
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]:

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

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

Design and Train the Decision Tree Model

In [37]:

```
dtree = DecisionTreeClassifier()
dtree.fit(y_train,z_train)
print("Decision Tree classifier Created")
```

```
Decision Tree classifier Created
```

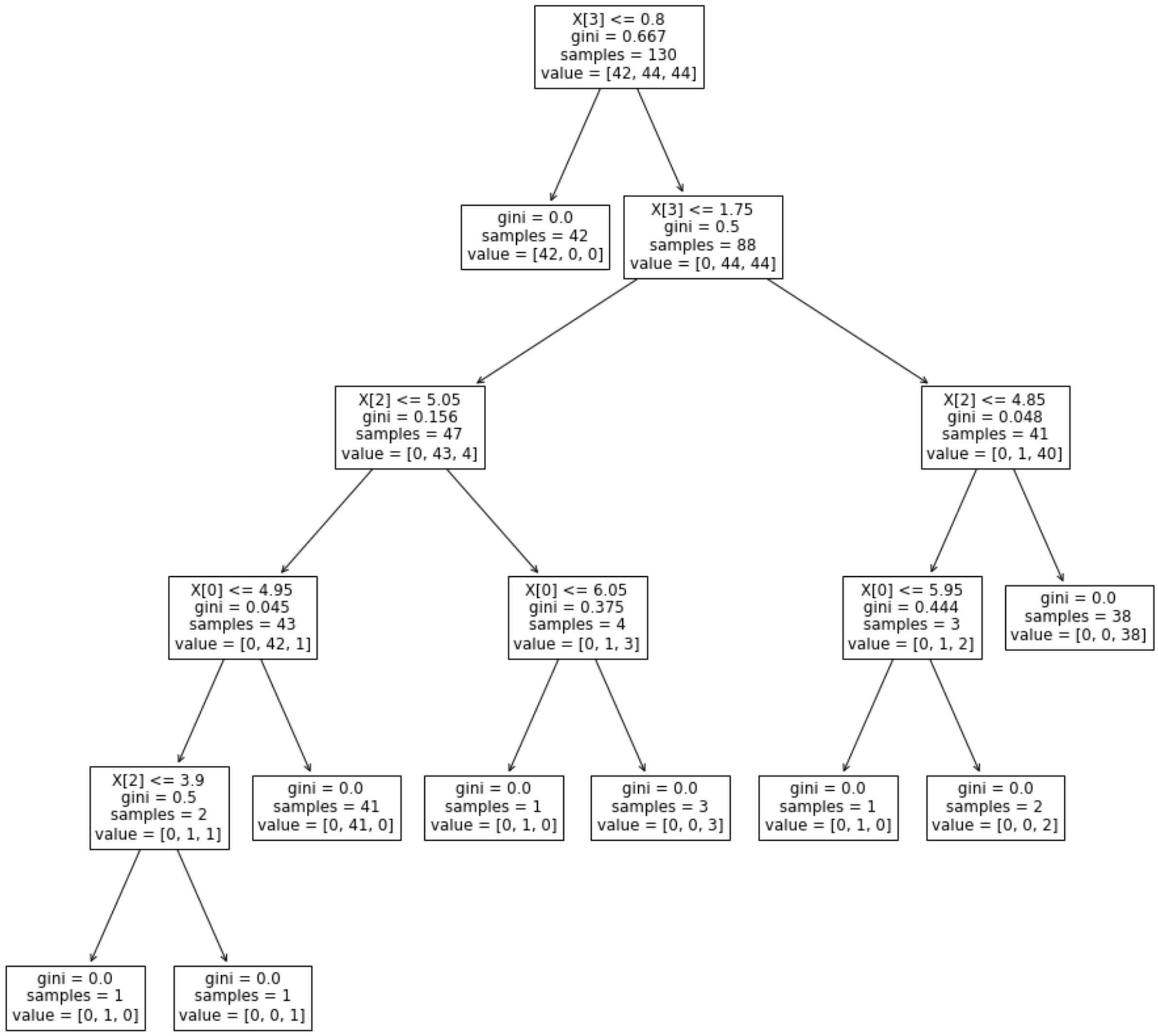
Visualize the Decision Tree Model

In [39]:

```
mt.figure(figsize=(16,16))
tree.plot_tree(dtree)
```

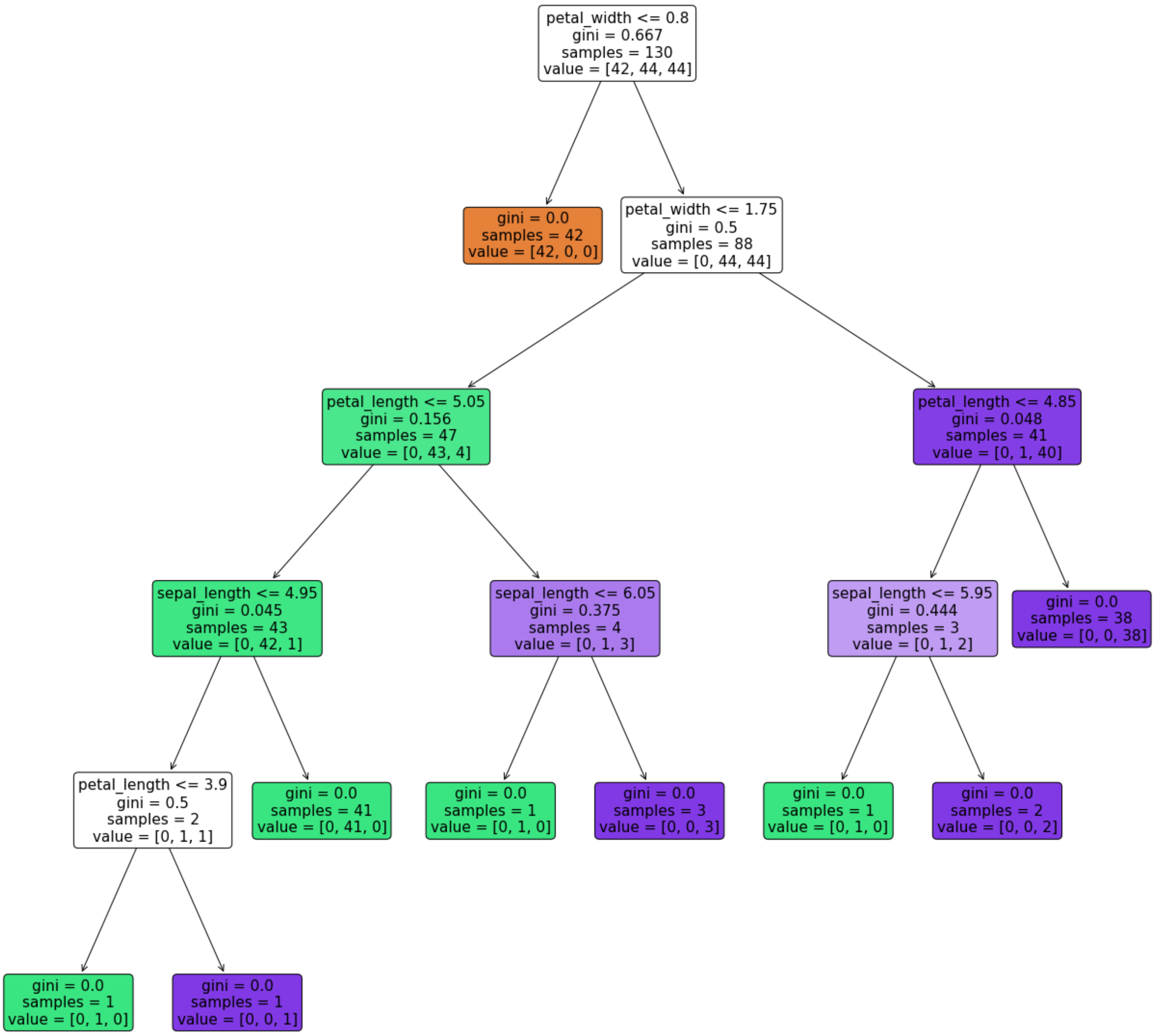
Out[39]:

```
[Text(478.2857142857142, 797.28, 'X[3] <= 0.8\ngini = 0.667\nsamples = 130\nvalue =
[42, 44, 44]'),
 Text(414.5142857142857, 652.3199999999999, 'gini = 0.0\nsamples = 42\nvalue = [42,
0, 0]'),
 Text(542.0571428571428, 652.3199999999999, 'X[3] <= 1.75\ngini = 0.5\nsamples = 88
\nvalue = [0, 44, 44]'),
 Text(318.85714285714283, 507.35999999999996, 'X[2] <= 5.05\ngini = 0.156\nsamples =
47\nvalue = [0, 43, 4]'),
 Text(191.3142857142857, 362.4, 'X[0] <= 4.95\ngini = 0.045\nsamples = 43\nvalue =
[0, 42, 1]'),
 Text(127.54285714285713, 217.43999999999994, 'X[2] <= 3.9\ngini = 0.5\nsamples = 2
\nvalue = [0, 1, 1]'),
 Text(63.771428571428565, 72.4799999999999, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
0]'),
 Text(191.3142857142857, 72.4799999999999, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
1]'),
 Text(255.08571428571426, 217.43999999999994, 'gini = 0.0\nsamples = 41\nvalue = [0,
41, 0]'),
 Text(446.4, 362.4, 'X[0] <= 6.05\ngini = 0.375\nsamples = 4\nvalue = [0, 1, 3]'),
 Text(382.6285714285714, 217.43999999999994, 'gini = 0.0\nsamples = 1\nvalue = [0,
1, 0]'),
 Text(510.1714285714285, 217.43999999999994, 'gini = 0.0\nsamples = 3\nvalue = [0,
0, 3]'),
 Text(765.2571428571428, 507.35999999999996, 'X[2] <= 4.85\ngini = 0.048\nsamples =
41\nvalue = [0, 1, 40]'),
 Text(701.4857142857143, 362.4, 'X[0] <= 5.95\ngini = 0.444\nsamples = 3\nvalue =
[0, 1, 2]'),
 Text(637.7142857142857, 217.43999999999994, 'gini = 0.0\nsamples = 1\nvalue = [0,
1, 0]'),
 Text(765.2571428571428, 217.43999999999994, 'gini = 0.0\nsamples = 2\nvalue = [0,
0, 2]'),
 Text(829.0285714285714, 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]:

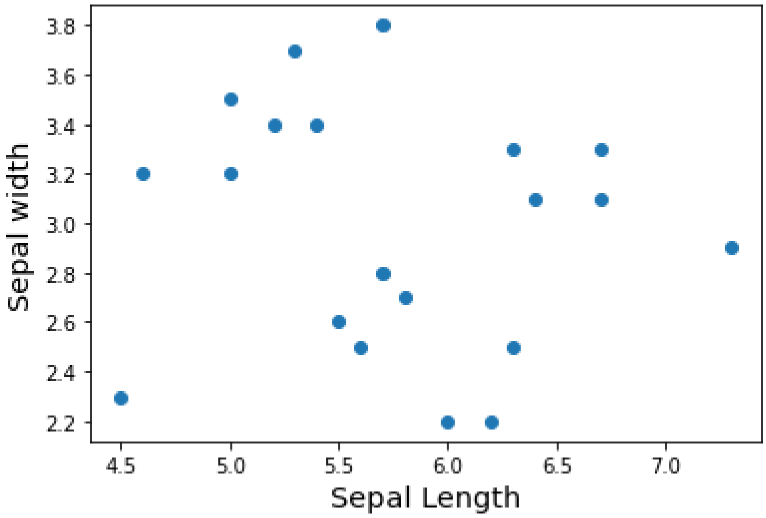
```
#actual vs predicted
result_df = pd.DataFrame({"ACTUAL":z_test,"PREDICTED":z_pred})
result_df
```

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 [45]:

```
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 [46]:
```
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 [47]:
```
#confusion matrix alone
conf_matrix=confusion_matrix(z_test,z_pred)
conf_matrix
```

Out[47]:
```
array([[8, 0, 0],
       [0, 6, 0],
       [0, 1, 5]], 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