In [38]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn import tree
import warnings
warnings.filterwarnings('ignore')
```

In [63]:
```
iris=datasets.load_iris()
print(iris.target_names)
print("data set str: ",dir(iris))
```

```
['setosa' 'versicolor' 'virginica']
data set str:  ['DESCR', 'data', 'data_module', 'feature_names', 'filename',
'frame', 'target', 'target_names']
```

In [60]:
```
df=pd.DataFrame(iris.data,columns=iris.feature_names)
df
df["target"]=iris.target
df.target.unique()
df['flower_species']=df.target.apply(lambda x : iris.target_names[x])
print("unique target values=",df['target'].unique())
df.sample(5)
```

```
unique target values= [0 1 2]
```

Out[60]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_species
140	6.7	3.1	5.6	2.4	2	virginica
116	6.5	3.0	5.5	1.8	2	virginica
85	6.0	3.4	4.5	1.6	1	versicolor
40	5.0	3.5	1.3	0.3	0	setosa
147	6.5	3.0	5.2	2.0	2	virginica

In [4]:
```
#df = pd.DataFrame(X, columns=iris.feature_names)
#df['species'] = iris.target
#df['species'] = df['species'].replace(to_replace= [0, 1, 2], value = ['setosa
```

In [80]:

Out[80]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

0=setosa,1=versicolor,2=virginica

In [5]:
```
df[df.target==0].head(3)
```

Out[5]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_species
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa

In [6]:
```
df[df.target==1].head(3)
```

Out[6]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_species
50	7.0	3.2	4.7	1.4	1	versicolor
51	6.4	3.2	4.5	1.5	1	versicolor
52	6.9	3.1	4.9	1.5	1	versicolor

In [7]:
```
df[df.target==2].head(3)
```

Out[7]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_species
100	6.3	3.3	6.0	2.5	2	virginica
101	5.8	2.7	5.1	1.9	2	virginica
102	7.1	3.0	5.9	2.1	2	virginica

In [8]:
```
df.columns
```

Out[8]:
```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)', 'target', 'flower_species'],
      dtype='object')
```

In [12]:
```
x=df[['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)']]
```

In [9]:
```
y=df['target']
```

In [38]:
```
x.shape
```

Out[38]:
```
(150, 4)
```

In [39]:
```
y.shape
```

Out[39]:
```
(150,)
```

In [13]:
```
from sklearn.model_selection import train_test_split
```

In [14]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.40,random_state
```

In [15]:
```
x_train
```

Out[15]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
11	4.8	3.4	1.6	0.2
113	5.7	2.5	5.0	2.0
123	6.3	2.7	4.9	1.8
12	4.8	3.0	1.4	0.1
2	4.7	3.2	1.3	0.2
...	...	...	...	...
133	6.3	2.8	5.1	1.5
137	6.4	3.1	5.5	1.8
72	6.3	2.5	4.9	1.5
140	6.7	3.1	5.6	2.4
37	4.9	3.6	1.4	0.1

90 rows × 4 columns

In [16]:
```
y_train.shape
```

Out[16]:
```
(90,)
```

In [15]:
```
x_test.shape
```

Out[15]:
```
(60, 4)
```

In [16]:
```
y_test.shape
```

Out[16]:
```
(60,)
```

In [17]:
```
from sklearn.tree import DecisionTreeClassifier
dtc= tree.DecisionTreeClassifier(random_state=1)
dtc.fit(x_train,y_train)
```

Out[17]:
```
DecisionTreeClassifier(random_state=1)
```

In [112]:
```

```

Out[112]:
```
DecisionTreeClassifier(random_state=1)
```

In [18]:
```
y_pred=dtc.predict(x_test)
y_pred
```

Out[18]:
```
array([0, 1, 1, 0, 2, 1, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       2, 0, 2, 1, 0, 0, 1, 2, 1, 2, 1, 2, 2, 0, 1, 0, 1, 2, 2, 0, 1, 2,
       1, 2, 0, 0, 0, 1, 0, 0, 2, 2, 2, 2, 2, 1, 2, 1])
```

In [ ]:
```
#checking the predicted x_test with y_test
a=iris.target_names[y_test.iloc[12]]
b=iris.target_names[dtc.predict([x_test.iloc[12]])]
print("iris y_test:" ,a)
print("iris y_pred:", b)
```

In [20]:
```
from sklearn.metrics import classification_report
classification_report(y_test,y_pred)
```

Out[20]:
```
'              precision    recall  f1-score   support\n\n           0
1.00      1.00      1.00        19\n           1       0.95      0.95      0.
95        21\n           2       0.95      0.95      0.95        20\n\n    ac
curacy                           0.97        60\n   macro avg       0.97
0.97      0.97        60\nweighted avg       0.97      0.97      0.97
60\n'
```

In [21]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[21]:
```
array([[19,  0,  0],
       [ 0, 20,  1],
       [ 0,  1, 19]], dtype=int64)
```

In [22]:
```
from sklearn import metrics
accuracy=metrics.accuracy_score(y_test,y_pred)*100
accuracy
```

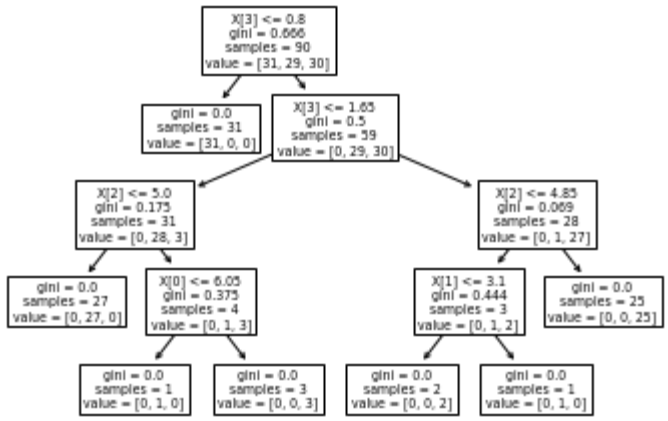
Out[22]:
```
96.66666666666667
```

In [23]:
```
#dtc.score(x_test,y_test)
```

Out[23]:
```
0.9666666666666667
```

In [24]:
```
tree.plot_tree(dtc)
```

Out[24]:
```
[Text(0.4, 0.9, 'X[3] <= 0.8\ngini = 0.666\nsamples = 90\nvalue = [31, 29, 3
0]'),
 Text(0.3, 0.7, 'gini = 0.0\nsamples = 31\nvalue = [31, 0, 0]'),
 Text(0.5, 0.7, 'X[3] <= 1.65\ngini = 0.5\nsamples = 59\nvalue = [0, 29, 3
0]'),
 Text(0.2, 0.5, 'X[2] <= 5.0\ngini = 0.175\nsamples = 31\nvalue = [0, 28,
3]'),
 Text(0.1, 0.3, 'gini = 0.0\nsamples = 27\nvalue = [0, 27, 0]'),
 Text(0.3, 0.3, 'X[0] <= 6.05\ngini = 0.375\nsamples = 4\nvalue = [0, 1,
3]'),
 Text(0.2, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.4, 0.1, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]'),
 Text(0.8, 0.5, 'X[2] <= 4.85\ngini = 0.069\nsamples = 28\nvalue = [0, 1, 2
7]'),
 Text(0.7, 0.3, 'X[1] <= 3.1\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(0.6, 0.1, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(0.8, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.9, 0.3, 'gini = 0.0\nsamples = 25\nvalue = [0, 0, 25]')]
```



In [25]:
```
import graphviz
```

In [42]:
```
#saving my tree in pdf
dot_data=tree.export_graphviz(dtc,out_file=None)
graph=graphviz.Source(dot_data)
graph.render#("isi")
```

Out[42]:
```
<bound method Render.render of <graphviz.sources.Source object at 0x0000020CC
DBACB50>>
```

In [52]:
```
view=tree.export_graphviz(dtc,out_file=None,feature_names=iris.feature_names,
                          class_names=iris.target_names,rounded=True,filled=T
graph=graphviz.Source(view)
graph
```

Out[52]: