Prediction using Decision Tree Algorithm

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as s
%matplotlib inline
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn import tree
```

In [2]:
```
#Importing dataset
df=pd.read_csv("C:\\Users\\User\\Downloads\\Iris (2).csv")
```

In [3]:
```
df.head()
```

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

In [4]:
```
df.shape
```

Out[4]:
```
(150, 6)
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [6]:
```
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 [7]:
```
df.describe()
```

Out[7]:

	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 [8]:
```
df['Species'].value_counts()
```

Out[8]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

In [9]:
```
#Splitting the data into independent and dependent variables

x=df.iloc[:,:4].values
y=df.iloc[:,-1:].values
```

In [10]:
```
x
```

```
[132. ,   7.9,   3.8,   6.4],
[133. ,   6.4,   2.8,   5.6],
[134. ,   6.3,   2.8,   5.1],
[135. ,   6.1,   2.6,   5.6],
[136. ,   7.7,   3. ,   6.1],
[137. ,   6.3,   3.4,   5.6],
[138. ,   6.4,   3.1,   5.5],
[139. ,   6. ,   3. ,   4.8],
[140. ,   6.9,   3.1,   5.4],
[141. ,   6.7,   3.1,   5.6],
[142. ,   6.9,   3.1,   5.1],
[143. ,   5.8,   2.7,   5.1],
[144. ,   6.8,   3.2,   5.9],
[145. ,   6.7,   3.3,   5.7],
[146. ,   6.7,   3. ,   5.2],
[147. ,   6.3,   2.5,   5. ],
[148. ,   6.5,   3. ,   5.2],
[149. ,   6.2,   3.4,   5.4],
[150. ,   5.9,   3. ,   5.1]])
```

In [11]:
```
y
```

```
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica'],
['Iris-virginica']], dtype=object)
```

In [12]:
```
#Converting labels into integers

l=LabelBinarizer()
y=l.fit_transform(y)
```

In [13]:
```
y
```

Out[13]:
```
array([[1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
```

In [14]:
```
#Splitting the data into testing and training sets

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [15]:
```
print(x_train.shape,x_test.shape)
```

```
(120, 4) (30, 4)
```

In [16]:
```
#Train the model

d=DecisionTreeClassifier()
d.fit(x_train,y_train)
```

Out[16]:
```
DecisionTreeClassifier()
```

In [17]:
```
#Making predictions

y_pred=d.predict(x_test)
y
```

Out[17]:
```
array([[1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
```

In [18]:
```
#Evaluating the model

c=confusion_matrix(y_test.argmax(axis=1),y_pred.argmax(axis=1))
c
```

Out[18]:
```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  1,  5]], dtype=int64)
```

In [19]:
```
accuracy_score(y_test.argmax(axis=1),y_pred.argmax(axis=1))
```

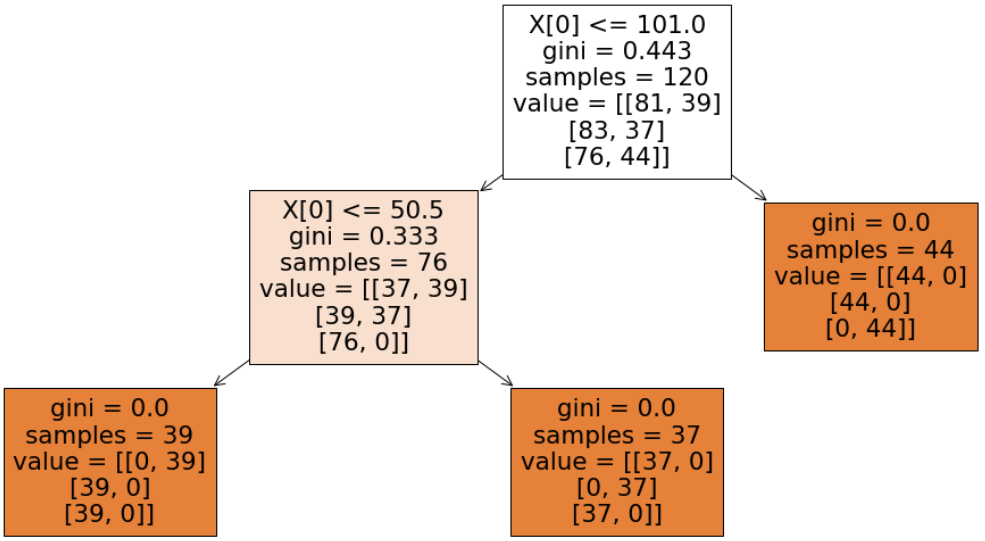
Out[19]:
```
0.9666666666666667
```

So approximate accuracy of the model is 97%

In [20]:
```
#Tree Visualization


matplotlib.rcParams['figure.figsize']=[20,9]

tree.plot_tree(d,filled=True);
```



In [ ]: