Assignment 6: Decision Tree

A. Import the usual libraries for pandas and plotting.

In [1]:
```
import pandas as pd
import seaborn as sb
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.tree import  DecisionTreeClassifier
import matplotlib.pyplot as plt
%matplotlib.inline
```

```
UsageError: Line magic function `%matplotlib.inline` not found.
```

In [2]:
```
from sklearn.datasets import load_iris
```

In [3]:
```
iris=load_iris()
```

In [4]:
```
iris.data
```

Out[4]:
```
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, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 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 [5]:
```
iris.target
```

Out[5]:
```
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 [6]:
```
X = iris.data
Y = iris.target
```

B)split train test datasets

In [7]:
```
X_train ,X_test,Y_train,Y_test = train_test_split(X,Y,random_state=42,test_size=0.30
```

In [8]:
```
X_train
```

Out[8]:
```
array([[5.5, 2.4, 3.7, 1. ],
       [6.3, 2.8, 5.1, 1.5],
       [6.4, 3.1, 5.5, 1.8],
       [6.6, 3. , 4.4, 1.4],
       [7.2, 3.6, 6.1, 2.5],
       [5.7, 2.9, 4.2, 1.3],
       [7.6, 3. , 6.6, 2.1],
       [5.6, 3. , 4.5, 1.5],
       [5.1, 3.5, 1.4, 0.2],
       [7.7, 2.8, 6.7, 2. ],
```

```
       [5.8, 2.7, 4.1, 1. ],
       [5.2, 3.4, 1.4, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [5.1, 3.8, 1.9, 0.4],
       [5. , 2. , 3.5, 1. ],
       [6.3, 2.7, 4.9, 1.8],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [5.6, 2.7, 4.2, 1.3],
       [5.1, 3.4, 1.5, 0.2],
       [5.7, 3. , 4.2, 1.2],
       [7.7, 3.8, 6.7, 2.2],
       [4.6, 3.2, 1.4, 0.2],
       [6.2, 2.9, 4.3, 1.3],
       [5.7, 2.5, 5. , 2. ],
       [5.5, 4.2, 1.4, 0.2],
       [6. , 3. , 4.8, 1.8],
       [5.8, 2.7, 5.1, 1.9],
       [6. , 2.2, 4. , 1. ],
       [5.4, 3. , 4.5, 1.5],
       [6.2, 3.4, 5.4, 2.3],
       [5.5, 2.3, 4. , 1.3],
       [5.4, 3.9, 1.7, 0.4],
       [5. , 2.3, 3.3, 1. ],
       [6.4, 2.7, 5.3, 1.9],
       [5. , 3.3, 1.4, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 2.4, 3.8, 1.1],
       [6.7, 3. , 5. , 1.7],
       [4.9, 3.1, 1.5, 0.2],
       [5.8, 2.8, 5.1, 2.4],
       [5. , 3.4, 1.5, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.9, 3.2, 4.8, 1.8],
       [5.1, 2.5, 3. , 1.1],
       [6.9, 3.2, 5.7, 2.3],
       [6. , 2.7, 5.1, 1.6],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [5.5, 2.5, 4. , 1.3],
       [4.4, 2.9, 1.4, 0.2],
       [4.3, 3. , 1.1, 0.1],
       [6. , 2.2, 5. , 1.5],
       [7.2, 3.2, 6. , 1.8],
       [4.6, 3.1, 1.5, 0.2],
       [5.1, 3.5, 1.4, 0.3],
       [4.4, 3. , 1.3, 0.2],
       [6.3, 2.5, 4.9, 1.5],
       [6.3, 3.4, 5.6, 2.4],
       [4.6, 3.4, 1.4, 0.3],
       [6.8, 3. , 5.5, 2.1],
       [6.3, 3.3, 6. , 2.5],
       [4.7, 3.2, 1.3, 0.2],
       [6.1, 2.9, 4.7, 1.4],
       [6.5, 2.8, 4.6, 1.5],
       [6.2, 2.8, 4.8, 1.8],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 5.3, 2.3],
       [5.1, 3.8, 1.6, 0.2],
       [6.9, 3.1, 5.4, 2.1],
       [5.9, 3. , 4.2, 1.5],
       [6.5, 3. , 5.2, 2. ],
       [5.7, 2.6, 3.5, 1. ],
       [5.2, 2.7, 3.9, 1.4],
       [6.1, 3. , 4.6, 1.4],
       [4.5, 2.3, 1.3, 0.3],
       [6.6, 2.9, 4.6, 1.3],
       [5.5, 2.6, 4.4, 1.2],
```

```
       [5.3, 3.7, 1.5, 0.2],
       [5.6, 3. , 4.1, 1.3],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [5.1, 3.7, 1.5, 0.4],
       [4.9, 2.4, 3.3, 1. ],
       [6.7, 3.3, 5.7, 2.5],
       [7.2, 3. , 5.8, 1.6],
       [4.9, 3.6, 1.4, 0.1],
       [6.7, 3.1, 5.6, 2.4],
       [4.9, 3. , 1.4, 0.2],
       [6.9, 3.1, 4.9, 1.5],
       [7.4, 2.8, 6.1, 1.9],
       [6.3, 2.9, 5.6, 1.8],
       [5.7, 2.8, 4.1, 1.3],
       [6.5, 3. , 5.5, 1.8],
       [6.3, 2.3, 4.4, 1.3],
       [6.4, 2.9, 4.3, 1.3],
       [5.6, 2.8, 4.9, 2. ],
       [5.9, 3. , 5.1, 1.8],
       [5.4, 3.4, 1.7, 0.2],
       [6.1, 2.8, 4. , 1.3],
       [4.9, 2.5, 4.5, 1.7],
       [5.8, 4. , 1.2, 0.2],
       [5.8, 2.6, 4. , 1.2],
       [7.1, 3. , 5.9, 2.1]])
```

In [9]:
```
Y_train
```

Out[9]:
```
array([1, 2, 2, 1, 2, 1, 2, 1, 0, 2, 1, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 1,
       2, 0, 1, 2, 0, 2, 2, 1, 1, 2, 1, 0, 1, 2, 0, 0, 1, 1, 0, 2, 0, 0,
       1, 1, 2, 1, 2, 2, 1, 0, 0, 2, 2, 0, 0, 0, 1, 2, 0, 2, 2, 0, 1, 1,
       2, 1, 2, 0, 2, 1, 2, 1, 1, 1, 0, 1, 1, 0, 1, 2, 2, 0, 1, 2, 2, 0,
       2, 0, 1, 2, 2, 1, 2, 1, 1, 2, 2, 0, 1, 2, 0, 1, 2])
```

C)train the model

In [10]:
```
model = DecisionTreeClassifier()
```

In [11]:
```
model.fit(X_train,Y_train)
```

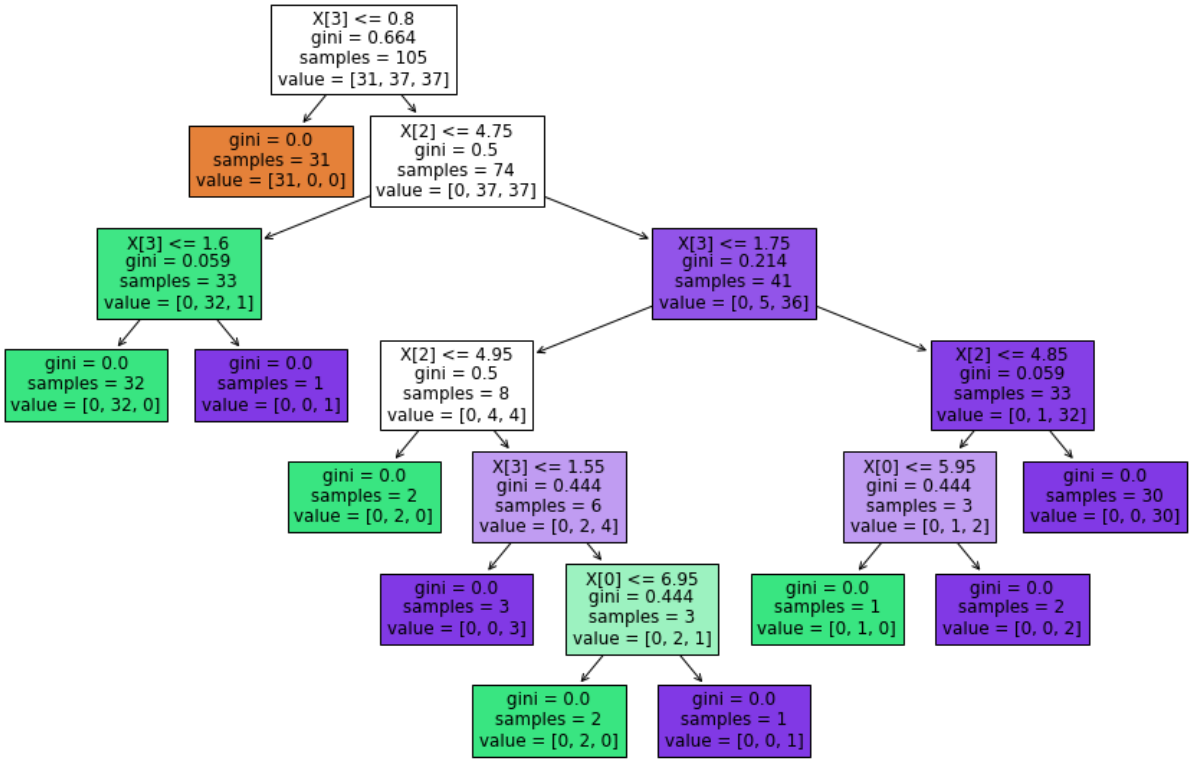
Out[11]:
```
DecisionTreeClassifier()
```

D)decision tree plot

In [12]:
```
plt.figure(figsize=(15,10))
tree.plot_tree(model,filled=True)
```

Out[12]:
```
[Text(257.53846153846155, 504.7714285714286, 'X[3] <= 0.8\ngini = 0.664\nsamples = 1
05\nvalue = [31, 37, 37]'),
 Text(193.15384615384616, 427.11428571428576, 'gini = 0.0\nsamples = 31\nvalue = [3
1, 0, 0]'),
 Text(321.9230769230769, 427.11428571428576, 'X[2] <= 4.75\ngini = 0.5\nsamples = 74
\nvalue = [0, 37, 37]'),
 Text(128.76923076923077, 349.4571428571429, 'X[3] <= 1.6\ngini = 0.059\nsamples = 3
3\nvalue = [0, 32, 1]'),
 Text(64.38461538461539, 271.8, 'gini = 0.0\nsamples = 32\nvalue = [0, 32, 0]'),
 Text(193.15384615384616, 271.8, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(515.0769230769231, 349.4571428571429, 'X[3] <= 1.75\ngini = 0.214\nsamples = 4
1\nvalue = [0, 5, 36]'),
 Text(321.9230769230769, 271.8, 'X[2] <= 4.95\ngini = 0.5\nsamples = 8\nvalue = [0,
4, 4]'),
 Text(257.53846153846155, 194.14285714285717, 'gini = 0.0\nsamples = 2\nvalue = [0,
2, 0]'),
 Text(386.3076923076923, 194.14285714285717, 'X[3] <= 1.55\ngini = 0.444\nsamples =
6\nvalue = [0, 2, 4]'),
 Text(321.9230769230769, 116.48571428571432, 'gini = 0.0\nsamples = 3\nvalue = [0,
0, 3]'),
```

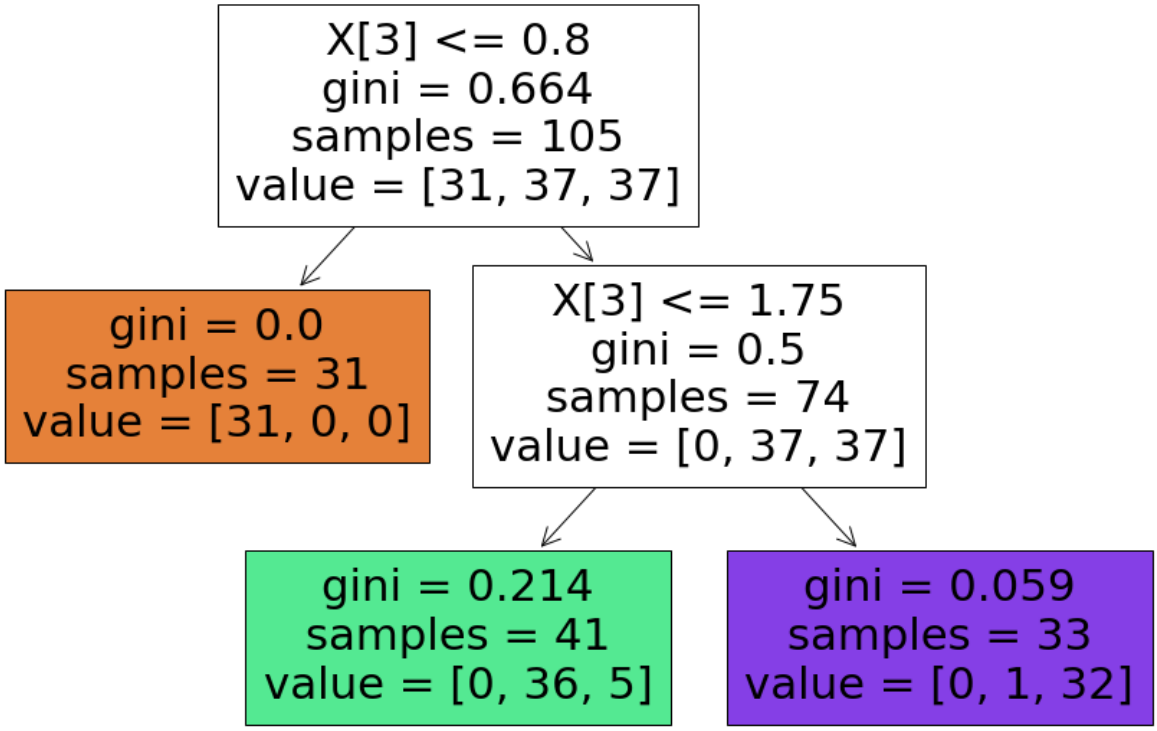
```
 Text(450.69230769230774, 116.48571428571432, 'X[0] <= 6.95\ngini = 0.444\nsamples =
3\nvalue = [0, 2, 1]'),
 Text(386.3076923076923, 38.82857142857142, 'gini = 0.0\nsamples = 2\nvalue = [0, 2,
0]'),
 Text(515.0769230769231, 38.82857142857142, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
1]'),
 Text(708.2307692307693, 271.8, 'X[2] <= 4.85\ngini = 0.059\nsamples = 33\nvalue =
[0, 1, 32]'),
 Text(643.8461538461538, 194.14285714285717, 'X[0] <= 5.95\ngini = 0.444\nsamples =
3\nvalue = [0, 1, 2]'),
 Text(579.4615384615385, 116.48571428571432, 'gini = 0.0\nsamples = 1\nvalue = [0,
1, 0]'),
 Text(708.2307692307693, 116.48571428571432, 'gini = 0.0\nsamples = 2\nvalue = [0,
0, 2]'),
 Text(772.6153846153846, 194.14285714285717, 'gini = 0.0\nsamples = 30\nvalue = [0,
0, 30]')]
```



E)decision tree plot with depth 2

In [13]:
```
model=DecisionTreeClassifier(max_depth=2)
model.fit(X_train,Y_train)
plt.figure(figsize=(15,10))
tree.plot_tree(model,filled=True)
```

Out[13]:
```
[Text(334.8, 453.0, 'X[3] <= 0.8\ngini = 0.664\nsamples = 105\nvalue = [31, 37, 3
7]'),
 Text(167.4, 271.8, 'gini = 0.0\nsamples = 31\nvalue = [31, 0, 0]'),
 Text(502.20000000000005, 271.8, 'X[3] <= 1.75\ngini = 0.5\nsamples = 74\nvalue =
[0, 37, 37]'),
 Text(334.8, 90.59999999999997, 'gini = 0.214\nsamples = 41\nvalue = [0, 36, 5]'),
 Text(669.6, 90.59999999999997, 'gini = 0.059\nsamples = 33\nvalue = [0, 1, 32]')]
```



F)predict the test results

In [16]:
```
Y_pred=model.predict(X_test)
Y_pred
```

Out[16]:
```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2,
       0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1, 0,
       0])
```

G)accuracy score,classification report

In [15]:
```
accuracy_score(Y_pred,Y_test)
print(classification_report(Y_pred,Y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```