Prepunning_Decision_Tree_Machine_Learning

In [6]:

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [7]:

```
from sklearn.datasets import load_iris
```

In [8]:

```
iris = load_iris()
```

In [9]:

```
iris
```

```
        [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],
```

In [10]:

```
iris.data
```

Out[10]:

```
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],
```

In [11]:

```
iris.target
```

Out[11]:

```
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, 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 [12]:

```
import seaborn as sns
```

In [13]:

```
df = sns.load_dataset('iris')
```

In [14]:

```
df.head()
```

Out[14]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [15]:

```
#independent feature and dependent features
X = df.iloc[:,:-1]
y = iris.target
```

In [16]:

```
X,y
```

Out[16]:

```
(     sepal_length  sepal_width  petal_length  petal_width
 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 x 4 columns],
 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 [17]:

```
### train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state
```

In [18]:

```
X_train
```

Out[18]:

	sepal_length	sepal_width	petal_length	petal_width
96	5.7	2.9	4.2	1.3
105	7.6	3.0	6.6	2.1
66	5.6	3.0	4.5	1.5
0	5.1	3.5	1.4	0.2
122	7.7	2.8	6.7	2.0
...	...	...	...	...
71	6.1	2.8	4.0	1.3
106	4.9	2.5	4.5	1.7
14	5.8	4.0	1.2	0.2
92	5.8	2.6	4.0	1.2
102	7.1	3.0	5.9	2.1

100 rows × 4 columns

In [19]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [ ]:

```

```

Post-Prunning

In [20]:

```
treemodel = DecisionTreeClassifier(max_depth = 2)
```

In [21]:

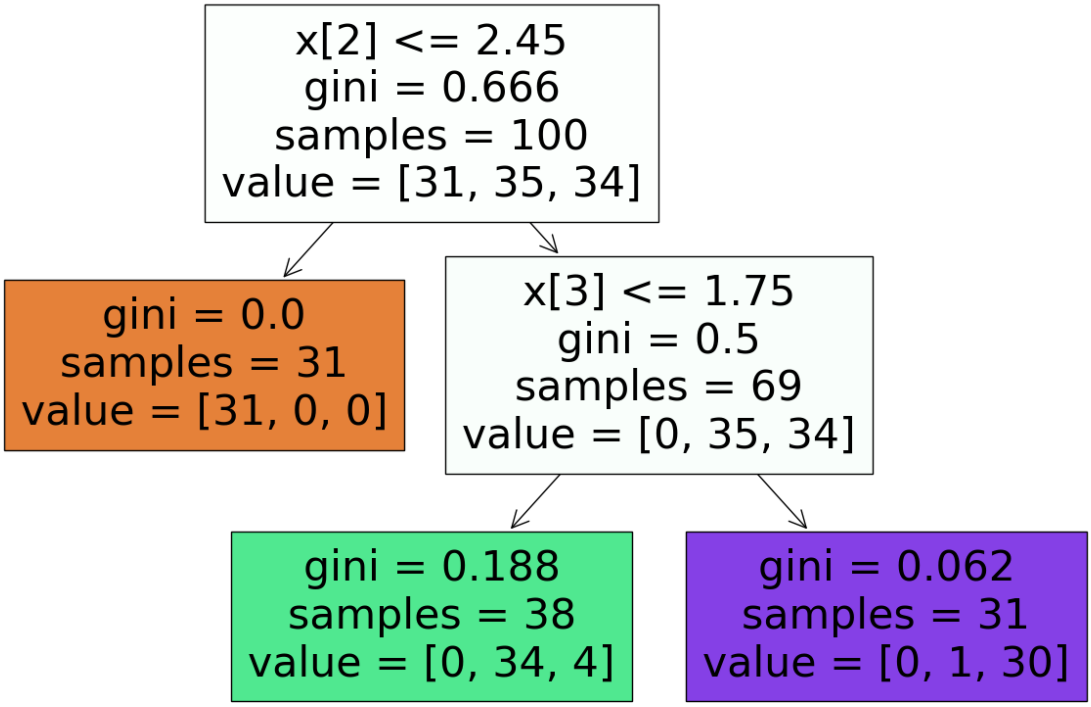
```
treemodel.fit(X_train,y_train)
```

Out[21]:

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=2)
```

In [22]:

```
from sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(treemodel,filled=True)
plt.show()
```



Prediction

In [23]:

```
y_pred = treemodel.predict(X_test)
```

In [24]:

```
y_pred
```

Out[24]:

```
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, 1, 1, 2, 1, 2])
```

In [25]:

```
from sklearn.metrics import accuracy_score,classification_report
```

In [26]:

```
score = accuracy_score(y_pred,y_test)
```

In [27]:

```
y_pred
```

Out[27]:

```
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, 1, 1, 2, 1, 2])
```

In [28]:

```
score
```

Out[28]:

```
0.98
```

In [29]:

```
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.94      0.97        16
           2       0.94      1.00      0.97        15

    accuracy                           0.98        50
   macro avg       0.98      0.98      0.98        50
weighted avg       0.98      0.98      0.98        50
```

Pre-Prunning

In [30]:

```
parameter = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'splitter':['best','random'],
    'max_depth': [1,2,3,4,5],
    'max_features': ['auto','sqrt','log2']
}
```

In [31]:

```
from sklearn.model_selection import GridSearchCV
```

In [32]:

```
treemodel=DecisionTreeClassifier()
cv=GridSearchCV(treemodel,param_grid=parameter,cv=5,scoring='accuracy')
```

In [33]:

```
cv.fit(X_train,y_train)
```

```
C:\Users\baps\anaconda3\lib\site-packages\sklearn\tree\_classes.py:269: F
utureWarning: `max_features='auto'` has been deprecated in 1.1 and will b
e removed in 1.3. To keep the past behaviour, explicitly set `max_feature
s='sqrt'`.
  warnings.warn(
C:\Users\baps\anaconda3\lib\site-packages\sklearn\tree\_classes.py:269: F
utureWarning: `max_features='auto'` has been deprecated in 1.1 and will b
e removed in 1.3. To keep the past behaviour, explicitly set `max_feature
s='sqrt'`.
  warnings.warn(
C:\Users\baps\anaconda3\lib\site-packages\sklearn\tree\_classes.py:269: F
utureWarning: `max_features='auto'` has been deprecated in 1.1 and will b
e removed in 1.3. To keep the past behaviour, explicitly set `max_feature
s='sqrt'`.
  warnings.warn(
C:\Users\baps\anaconda3\lib\site-packages\sklearn\tree\_classes.py:269: F
utureWarning: `max_features='auto'` has been deprecated in 1.1 and will b
e removed in 1.3. To keep the past behaviour, explicitly set `max_feature
s='sqrt'`.
```

In [34]:

```
cv.best_params_
```

Out[34]:

```
{'criterion': 'entropy',
 'max_depth': 5,
 'max_features': 'log2',
 'splitter': 'random'}
```

In [35]:

```
y_test
```

Out[35]:

```
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, 1, 2, 2, 1, 2])
```

In [36]:

```
y_pred
```

Out[36]:

```
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, 1, 1, 2, 1, 2])
```

In [37]:

```
y_pred = cv.predict(X_test)
```

In [38]:

```
from sklearn.metrics import accuracy_score, classification_report
```

In [39]:

```
score = accuracy_score(y_pred, y_test)
```

In [40]:

```
score
```

Out[40]:

```
0.94
```

In [41]:

```
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        17
           1       1.00      0.83      0.91        18
           2       0.94      1.00      0.97        15

    accuracy                           0.94        50
   macro avg       0.94      0.94      0.94        50
weighted avg       0.95      0.94      0.94        50
```

In [ ]:

```

```