Decision Tree Classifier Implementation with post Preprunning

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

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

In [4]:
```
dataset=load_iris()
```

In [6]:
```
print(dataset.DESCR)
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

|details-start|
**References**
```

```
|details-split|

- Fisher, R.A. "The use of multiple measurements in taxonomic problems"
  Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
  Mathematical Statistics" (John Wiley, NY, 1950).
- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
  (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
  Structure and Classification Rule for Recognition in Partially Exposed
  Environments".  IEEE Transactions on Pattern Analysis and Machine
  Intelligence, Vol. PAMI-2, No. 1, 67-71.
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
  on Information Theory, May 1972, 431-433.
- See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
  conceptual clustering system finds 3 classes in the data.
- Many, many more ...

|details-end|
```

In [8]:
```python
import seaborn as sns
```

In [10]:
```python
df=sns.load_dataset('iris')
```

In [ ]:
```python

```

In [15]:
```python
x=df.iloc[:,:-1]
y=dataset.target
```

In [13]:
```python
x,y
```

Out[13]:
```
(     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, 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 [16]:
```python
from sklearn.model_selection import train_test_split
```

In [17]:
```python
x_tarin,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=54)
```

In [18]:
```python
from sklearn.tree import DecisionTreeClassifier
```

In [19]:
```python
#Post Pruning
```

In [20]:
```python
treeclassifier=DecisionTreeClassifier()
```

In [21]:
```
treeclassifier
```

Out[21]:

▾ DecisionTreeClassifier

DecisionTreeClassifier()

In [22]:
```
treeclassifier.fit(x_tarin,y_train)
```

Out[22]:

▾ DecisionTreeClassifier

DecisionTreeClassifier()

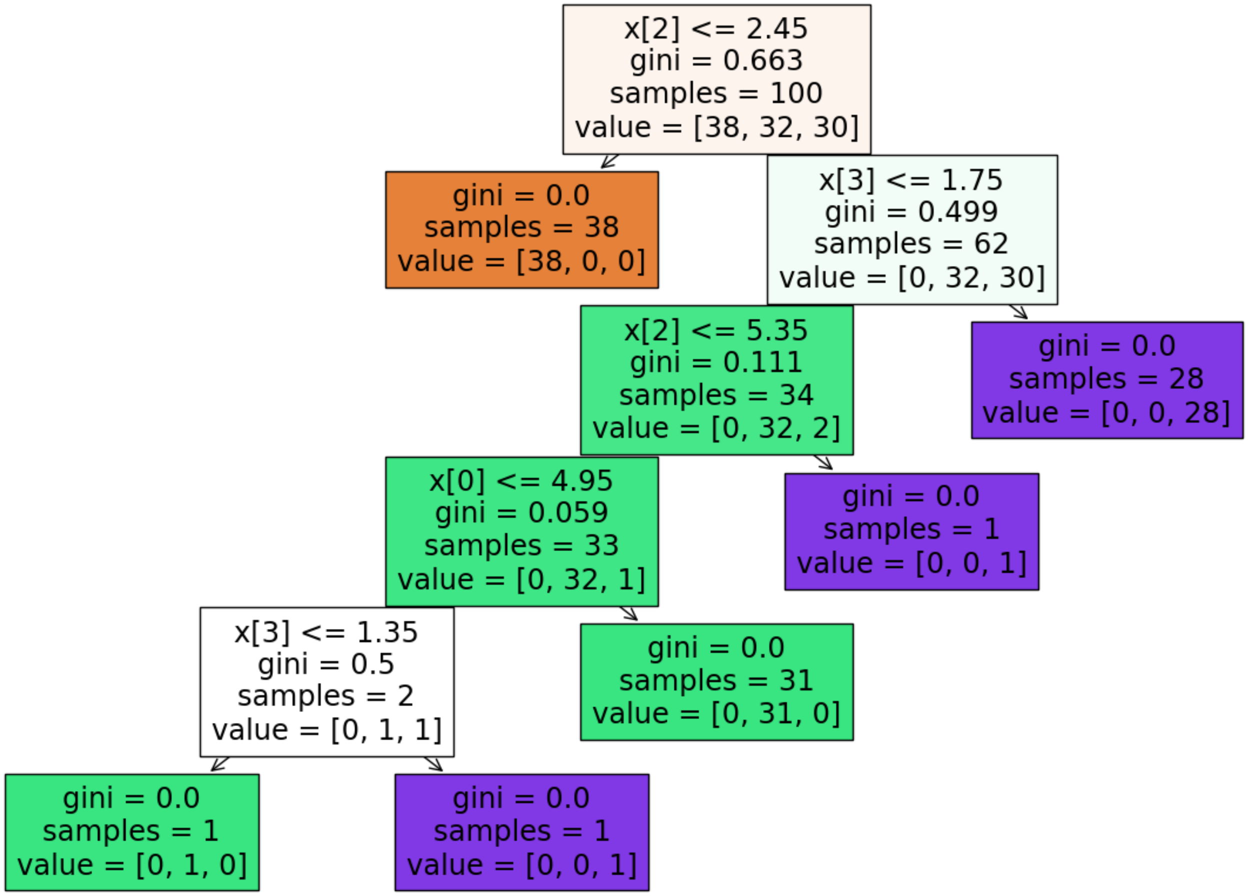
In [24]:
```
x_tarin.head()
```

Out[24]:

	sepal_length	sepal_width	petal_length	petal_width
108	6.7	2.5	5.8	1.8
139	6.9	3.1	5.4	2.1
48	5.3	3.7	1.5	0.2
36	5.5	3.5	1.3	0.2
109	7.2	3.6	6.1	2.5

In [23]:
```
from sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(treeclassifier,filled=True)
```

Out[23]:
```
[Text(0.5714285714285714, 0.9166666666666666, 'x[2] <= 2.45\ngini = 0.663\nsamples = 100
\nvalue = [38, 32, 30]'),
 Text(0.42857142857142855, 0.75, 'gini = 0.0\nsamples = 38\nvalue = [38, 0, 0]'),
 Text(0.7142857142857143, 0.75, 'x[3] <= 1.75\ngini = 0.499\nsamples = 62\nvalue = [0, 3
2, 30]'),
 Text(0.5714285714285714, 0.5833333333333334, 'x[2] <= 5.35\ngini = 0.111\nsamples = 34
\nvalue = [0, 32, 2]'),
 Text(0.42857142857142855, 0.4166666666666667, 'x[0] <= 4.95\ngini = 0.059\nsamples = 33
\nvalue = [0, 32, 1]'),
 Text(0.2857142857142857, 0.25, 'x[3] <= 1.35\ngini = 0.5\nsamples = 2\nvalue = [0, 1,
1]'),
 Text(0.14285714285714285, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
0]'),
 Text(0.42857142857142855, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
1]'),
 Text(0.5714285714285714, 0.25, 'gini = 0.0\nsamples = 31\nvalue = [0, 31, 0]'),
 Text(0.7142857142857143, 0.4166666666666667, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
1]'),
 Text(0.8571428571428571, 0.5833333333333334, 'gini = 0.0\nsamples = 28\nvalue = [0, 0,
28]')]
```

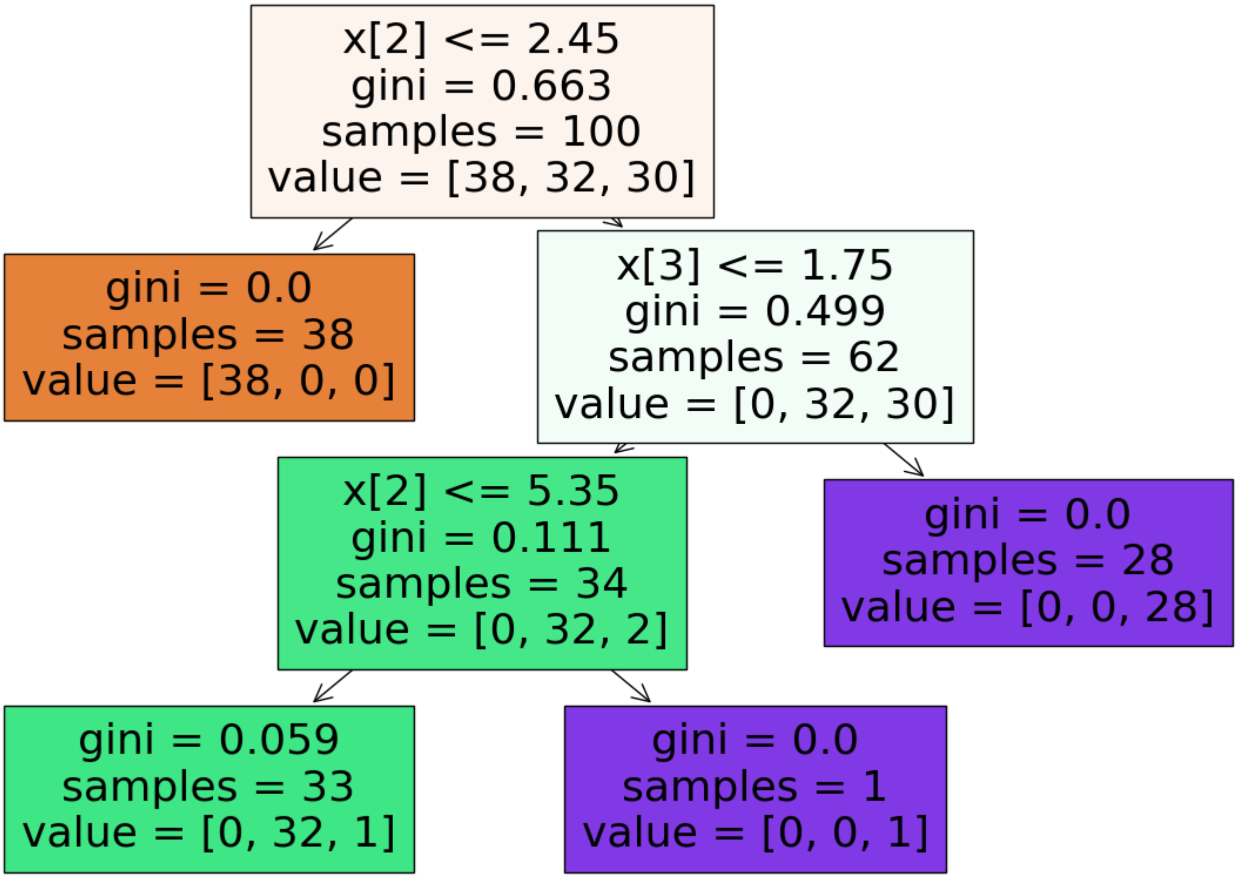


In [29]:
```
# post pruning with max_depth = 2
treeclassifier=DecisionTreeClassifier(max_depth=3)
treeclassifier.fit(x_tarin,y_train)
```

Out[29]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

In [30]:
```
from sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(treeclassifier,filled=True)
```

Out[30]:
```
[Text(0.4, 0.875, 'x[2] <= 2.45\ngini = 0.663\nsamples = 100\nvalue = [38, 32, 30]'),
 Text(0.2, 0.625, 'gini = 0.0\nsamples = 38\nvalue = [38, 0, 0]'),
 Text(0.6, 0.625, 'x[3] <= 1.75\ngini = 0.499\nsamples = 62\nvalue = [0, 32, 30]'),
 Text(0.4, 0.375, 'x[2] <= 5.35\ngini = 0.111\nsamples = 34\nvalue = [0, 32, 2]'),
 Text(0.2, 0.125, 'gini = 0.059\nsamples = 33\nvalue = [0, 32, 1]'),
 Text(0.6, 0.125, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.8, 0.375, 'gini = 0.0\nsamples = 28\nvalue = [0, 0, 28]')]
```



In [31]:
```
#prediction
y_pred=treeclassifier.predict(x_test)
```

In [32]:
```
y_pred
```

Out[32]:
```
array([0, 0, 1, 2, 1, 1, 0, 1, 2, 0, 0, 2, 2, 2, 1, 1, 2, 2, 0, 0, 1, 2,
       1, 1, 2, 1, 1, 0, 1, 1, 0, 2, 0, 1, 1, 2, 2, 1, 2, 0, 1, 1, 2, 2,
       1, 0, 2, 2, 2, 2])
```

In [33]:
```
from sklearn.metrics import accuracy_score,classification_report
```

In [35]:
```
score=accuracy_score(y_pred,y_test)
print(score)
print(classification_report(y_pred,y_test))
```

```
0.94
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       0.94      0.89      0.92        19
           2       0.90      0.95      0.92        19

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

DecisionTree Prepruning And Hyperparameter Tuning For Huge Data

In [36]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [37]:
```
parameter={
    'criterion':['gini','entropy','log_loss'],
```

```python
    'splitter':['best','random'],
    'max_depth':[1,2,3,4,5],
    'max_features':['auto','sqrt','log2']
}
```

In [39]:

```python

```

In [41]:

```python
treeclassifier=DecisionTreeClassifier()
clf=GridSearchCV(treeclassifier,param_grid=parameter,cv=5,scoring='accuracy')
```

In [42]:

```python
#Train the data
clf.fit(x_tarin,y_train)
```

Out[42]:

```
▸ GridSearchCV
▸ estimator: DecisionTreeClassifier
    ▸ DecisionTreeClassifier
```

In [43]:

```python
clf.best_params_
```

Out[43]:

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

In [45]:

```python
y_pred=clf.predict(x_test)
```

In [46]:

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

In [47]:

```python
score
```

Out[47]:

```
0.92
```

In [48]:

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

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       0.94      0.85      0.89        20
           2       0.85      0.94      0.89        18

    accuracy                           0.92        50
   macro avg       0.93      0.93      0.93        50
weighted avg       0.92      0.92      0.92        50
```

In [ ]:

```python

```