Decision Tree Prediction

Debashis Saha

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Data Reading

In [2]:

```
iris = pd.read_csv("Iris.csv")
iris.head()
```

Out[2]:

	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 [3]:

```
iris.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 [4]:
```
iris.Species.value_counts()
```

Out[4]:
```
Iris-virginica     50
Iris-versicolor    50
Iris-setosa        50
Name: Species, dtype: int64
```

In [5]:
```
iris['Species_class']=np.where(iris.Species=='Iris-virginica',1,np.wher
e(iris.Species=='Iris-versicolor',2,3))
```

In [6]:
```
iris.Species_class.value_counts()
```

Out[6]:
```
3    50
2    50
1    50
Name: Species_class, dtype: int64
```

In [7]:
```
iris.columns
```

Out[7]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWi
dthCm',
       'Species', 'Species_class'],
      dtype='object')
```

In [8]:
```
cols=['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
```

Model Preparation

In [9]:
```
from sklearn.model_selection import train_test_split
```

```
train_X, test_X, train_y, test_y = train_test_split( iris[cols],
                                                      iris['Species_class'
],
                                                      test_size = 0.2,
                                                      random_state = 123 )
```

Model Building

In [10]:
```
param_grid = {'max_depth': np.arange(2, 8),'max_features': np.arange(2,
5)}
```

In [12]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier, export_graphviz, expor
t
tree = GridSearchCV(DecisionTreeClassifier(), param_grid, cv = 10,verbo
se=1,n_jobs=-1)
tree.fit( train_X, train_y )
```

```
Fitting 10 folds for each of 18 candidates, totalling 180 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done 156 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done 180 out of 180 | elapsed:    1.2s finished
```

Out[12]:
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(), n_jobs=-1,
             param_grid={'max_depth': array([2, 3, 4, 5, 6, 7]),
                         'max_features': array([2, 3, 4])},
             verbose=1)
```

In [13]:
```
tree.best_score_
```

Out[13]:
```
0.9583333333333334
```

In [14]:
```
tree.best_estimator_
```

```
Out[14]: DecisionTreeClassifier(max_depth=5, max_features=2)
```

```
In [15]: tree.best_params_
```

```
Out[15]: {'max_depth': 5, 'max_features': 2}
```

```
In [16]: train_pred = tree.predict(train_X)
```

```
In [17]: test_pred = tree.predict(test_X)
```

```
In [18]: import sklearn.metrics as metrics
         print(metrics.classification_report(test_y, test_pred))
```

```
              precision    recall  f1-score   support

           1       0.92      1.00      0.96        11
           2       1.00      0.83      0.91         6
           3       1.00      1.00      1.00        13

    accuracy                           0.97        30
   macro avg       0.97      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30
```

Building Decision Tree

```
In [19]: clf_tree = DecisionTreeClassifier( max_depth = 4, max_features=2)
         clf_tree.fit( train_X, train_y )
```

```
Out[19]: DecisionTreeClassifier(max_depth=4, max_features=2)
```

```
In [20]: tree_test_pred = pd.DataFrame( { 'actual':  test_y,'predicted': clf_tre
         e.predict( test_X ) } )
```

```
In [21]: tree_test_pred.sample( n = 10 )
```

Out[21]:

	actual	predicted
31	3	3
135	1	1
42	3	3
132	1	1
37	3	3
72	2	2
90	2	2
127	1	1
133	1	2
4	3	3

In [22]:

```
metrics.accuracy_score( tree_test_pred.actual, tree_test_pred.predicted
 )
```

Out[22]:

```
0.9333333333333333
```

In [23]:

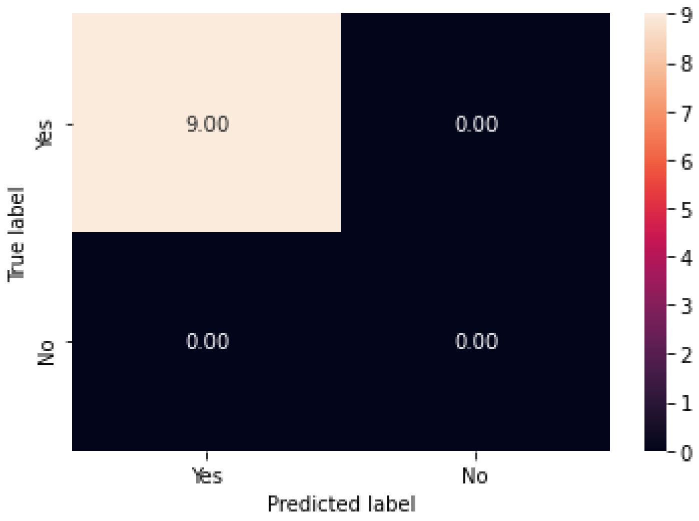
```
tree_cm = metrics.confusion_matrix( tree_test_pred.predicted,tree_test_
pred.actual,[1,0] )
sns.heatmap(tree_cm, annot=True,
         fmt='.2f',
         xticklabels = ["Yes", "No"] , yticklabels = ["Yes", "No"] )

plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:
68: FutureWarning: Pass labels=[1, 0] as keyword args. From version 0.2
5 passing these as positional arguments will result in an error
  warnings.warn("Pass {} as keyword args. From version 0.25 "
```

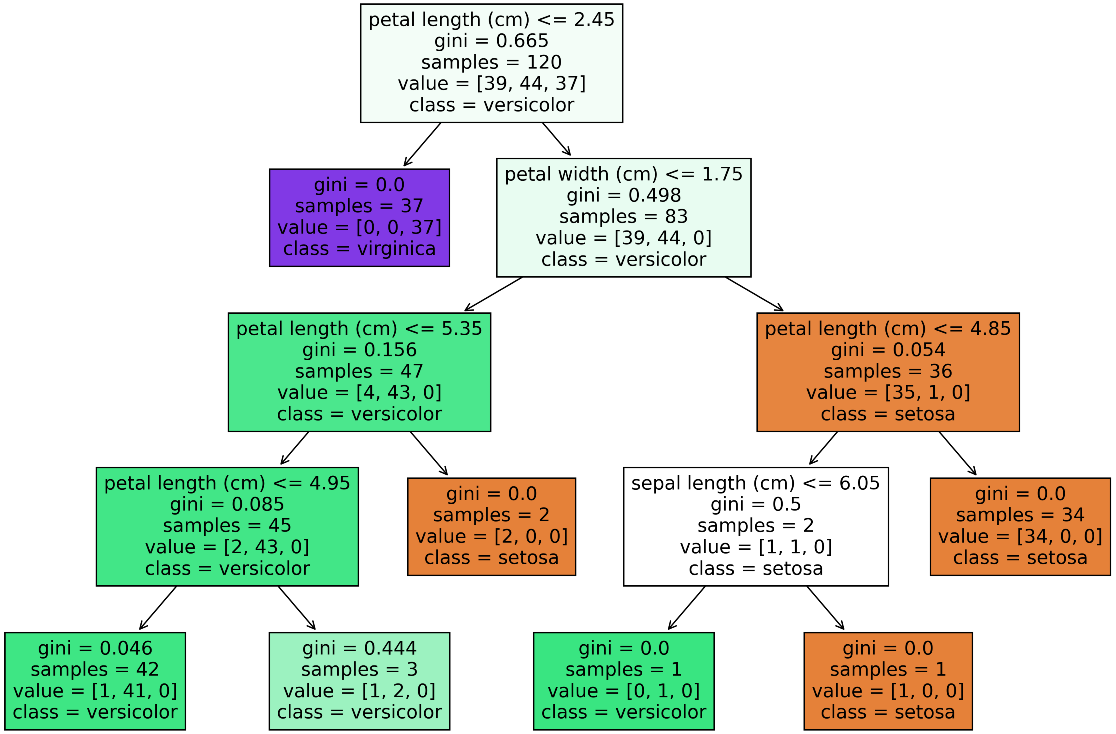
Out[23]:

```
Text(0.5, 15.0, 'Predicted label')
```



Graphical Representation of Decision Tree

In [24]:
```
from sklearn import tree
fn=['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)']
cn=['setosa', 'versicolor', 'virginica']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (15,10), dpi=300)
tree.plot_tree(clf_tree,
               feature_names = fn,
               class_names=cn,
               filled = True);
fig.savefig('imagename.png')
```


petal length (cm) <= 2.45
gini = 0.665
samples = 120
value = [39, 44, 37]
class = versicolor
gini = 0.0
samples = 37
value = [0, 0, 37]
class = virginica
petal width (cm) <= 1.75
gini = 0.498
samples = 83
value = [39, 44, 0]
class = versicolor
petal length (cm) <= 5.35
gini = 0.156
samples = 47
value = [4, 43, 0]
class = versicolor
petal length (cm) <= 4.85
gini = 0.054
samples = 36
value = [35, 1, 0]
class = setosa
petal length (cm) <= 4.95
gini = 0.085
samples = 45
value = [2, 43, 0]
class = versicolor
gini = 0.0
samples = 2
value = [2, 0, 0]
class = setosa
sepal length (cm) <= 6.05
gini = 0.5
samples = 2
value = [1, 1, 0]
class = setosa
gini = 0.0
samples = 34
value = [34, 0, 0]
class = setosa
gini = 0.046
samples = 42
value = [1, 41, 0]
class = versicolor
gini = 0.444
samples = 3
value = [1, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 1
value = [1, 0, 0]
class = setosa