Prediction using Decision Tree Algorithm

By- Ketki KaleTasks: 1) Create the Decision Tree classifier and visualize it graphically. 2) The purpose is if we feed any new data to this classifier, it would be able to predict the right class accordingly.

In [35]:

```
# Load the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [36]:

```
# read the file
df=pd.read_csv("Iris.csv")
df.head()
```

Out[36]:

	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 [37]:

```
# total no of rows and columns
df.shape
```

Out[37]:

```
(150, 6)
```

In [38]:

```
# to check if any null values are present
df.isnull().sum()
```

Out[38]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [39]:

```
# information about the given data
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 [40]:

```
# statistical information about the data
df.describe()
```

Out[40]:

	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 [41]:

```
# removing id column as its not of much importance
df1 = df.drop(["Id"],axis=1)
```

In [42]:

```
df1.head()
```

Out[42]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [43]:

```
# renaming the columns
df1.rename(columns={'SepalLengthCm':'Sepal Length','SepalWidthCm': 'Sepal Width','Petal
                   'PetalWidthCm':'Petal Width'},inplace=True)
```

In [44]:

```
df1.head()
```

Out[44]:

	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa

	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [45]:

```python
# total no of each species
df1['Species'].value_counts()
```

Out[45]:

```
Iris-virginica     50
Iris-setosa        50
Iris-versicolor    50
Name: Species, dtype: int64
```

In [46]:

```python
labels=df1['Species'].unique()
labels
```

Out[46]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [47]:

```python
# defining the X(independent) and Y(dependent) variables
X=df1.iloc[:,:-1]
X
```

Out[47]:

	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 × 4 columns

In [48]:

```python
Y=df1.iloc[:,-1]
Y
```

Out[48]:

```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
```

```
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

In [49]:

```
colnames= list(X.columns.values.tolist())
```

In [50]:

```
# Divide the dataset into two parts for training and testing in 70% and 30% proportion
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=0)
```

In [51]:

```
X_train.shape
```

Out[51]:

```
(105, 4)
```

In [52]:

```
X_test.shape
```

Out[52]:

```
(45, 4)
```

In [53]:

```
#Create and train Decision Tree Model on training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train,Y_train)
```

Out[53]:

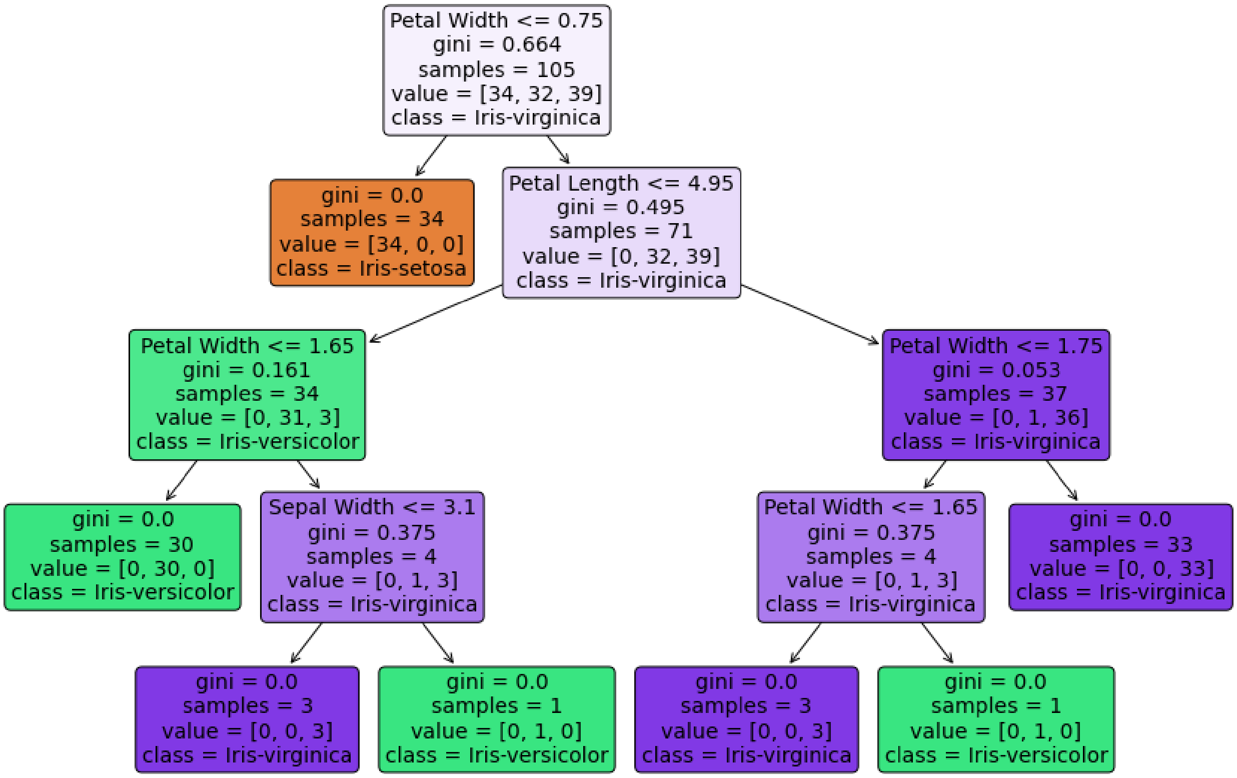
```
DecisionTreeClassifier()
```

In [54]:

```
# visualising the tree graphically
from sklearn import tree
import matplotlib.pyplot as plt

plt.figure(figsize=(15,10))
#create the tree plot
a = tree.plot_tree(classifier,
                   #use the feature names stored
                   feature_names = colnames,
                   #use the class names stored
                   class_names = labels,
                   rounded = True,
                   filled = True,
                   fontsize=14)

plt.show()
```



In [55]:

```
#Make predictions based on the testing set using the trained model
Y_pred = classifier.predict(X_test)
```

In [56]:

```
data = pd.DataFrame({'Actual': Y_test, 'Predicted' : Y_pred})
data
```

Out[56]:

	Actual	Predicted
114	Iris-virginica	Iris-virginica
62	Iris-versicolor	Iris-versicolor
33	Iris-setosa	Iris-setosa
107	Iris-virginica	Iris-virginica
7	Iris-setosa	Iris-setosa
100	Iris-virginica	Iris-virginica
40	Iris-setosa	Iris-setosa
86	Iris-versicolor	Iris-versicolor
76	Iris-versicolor	Iris-versicolor
71	Iris-versicolor	Iris-versicolor
134	Iris-virginica	Iris-virginica
51	Iris-versicolor	Iris-versicolor
73	Iris-versicolor	Iris-versicolor

	Actual	Predicted
54	Iris-versicolor	Iris-versicolor
63	Iris-versicolor	Iris-versicolor
37	Iris-setosa	Iris-setosa
78	Iris-versicolor	Iris-versicolor
90	Iris-versicolor	Iris-versicolor
45	Iris-setosa	Iris-setosa
16	Iris-setosa	Iris-setosa
121	Iris-virginica	Iris-virginica
66	Iris-versicolor	Iris-versicolor
24	Iris-setosa	Iris-setosa
8	Iris-setosa	Iris-setosa
126	Iris-virginica	Iris-virginica
22	Iris-setosa	Iris-setosa
44	Iris-setosa	Iris-setosa
97	Iris-versicolor	Iris-versicolor
93	Iris-versicolor	Iris-versicolor
26	Iris-setosa	Iris-setosa
137	Iris-virginica	Iris-virginica
84	Iris-versicolor	Iris-versicolor
27	Iris-setosa	Iris-setosa
127	Iris-virginica	Iris-virginica
132	Iris-virginica	Iris-virginica
59	Iris-versicolor	Iris-versicolor
18	Iris-setosa	Iris-setosa
83	Iris-versicolor	Iris-virginica
61	Iris-versicolor	Iris-versicolor
92	Iris-versicolor	Iris-versicolor
112	Iris-virginica	Iris-virginica
2	Iris-setosa	Iris-setosa
141	Iris-virginica	Iris-virginica
43	Iris-setosa	Iris-setosa
10	Iris-setosa	Iris-setosa

In [57]:

```
# Check the performance of model by following parameters
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
print(confusion_matrix(Y_test,Y_pred))
```

```
[[16  0  0]
 [ 0 17  1]
 [ 0  0 11]]
```

In [58]:

```
print("Accuracy Score : ",accuracy_score(Y_test,Y_pred))
```

```
Accuracy Score :  0.9777777777777777
```

In [59]:

```
print(classification_report(Y_test,Y_pred))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        16
Iris-versicolor       1.00      0.94      0.97        18
 Iris-virginica       0.92      1.00      0.96        11

       accuracy                           0.98        45
      macro avg       0.97      0.98      0.98        45
   weighted avg       0.98      0.98      0.98        45
```

In [60]:

```
X_test.shape
```

Out[60]:

```
(45, 4)
```

In [61]:

```
#feature importance
importance = pd.DataFrame({'feature': X_train.columns,
'importance' : np.round(classifier.feature_importances_, 3)})
importance.sort_values('importance', ascending=False, inplace = True)
print(importance)
```

```
        feature  importance
3   Petal Width       0.581
2  Petal Length       0.398
1   Sepal Width       0.022
0  Sepal Length       0.000
```

In [62]:

```
# Import label encoder
from sklearn import preprocessing
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
# Encode labels in column 'Country'.
df1['Species']= label_encoder.fit_transform(df1['Species'])
y1=df1['Species']
y1=np.array(y1)
y1
```

Out[62]:

```
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 [63]:

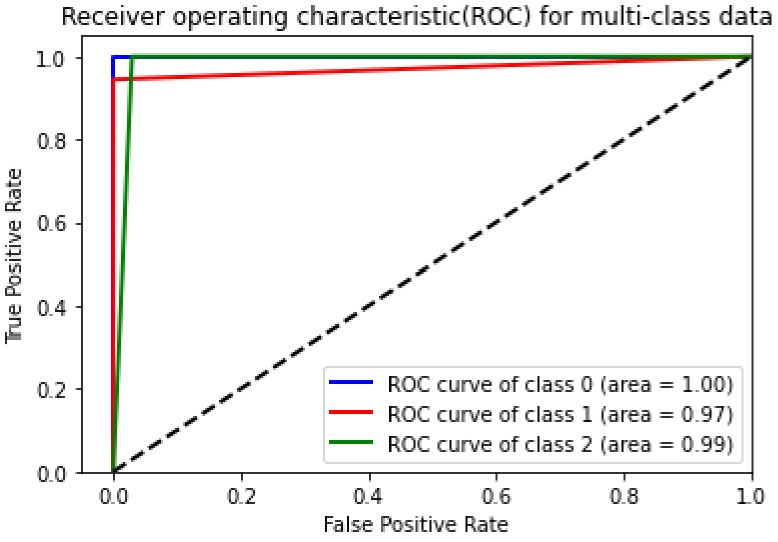
```
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve, auc
from sklearn.multiclass import OneVsRestClassifier
from itertools import cycle
# Binarize the output
y = label_binarize(y1, classes=[0, 1, 2])
n_classes = y.shape[1]


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0

classifier1 = OneVsRestClassifier(DecisionTreeClassifier(random_state=0))
y_score = classifier1.fit(X_train, y_train).predict_proba(X_test)


fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
lw=2
colors = cycle(['blue', 'red', 'green'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic(ROC) for multi-class data')
plt.legend(loc="lower right")
plt.show()
```



In [64]:

```
# out of sample prediction
features = [[7.25, 2.45, 0.34, 3.5]]
own_pred = classifier.predict(features)
print("Features = {}".format(features))
print("species = {}".format(own_pred[0]))
```

```
Features = [[7.25, 2.45, 0.34, 3.5]]
species = Iris-virginica
```

In [65]:

```
# out of sample prediction
features = [[7.25, 3.45, 5.34, 0.5]]
own_pred = classifier.predict(features)
print("Features = {}".format(features))
print("species = {}".format(own_pred[0]))
```

```
Features = [[7.25, 3.45, 5.34, 0.5]]
species = Iris-setosa
```

In [66]:

```
# out of sample prediction
features = [[4.8, 2.2, 3.34, 1.5]]
own_pred = classifier.predict(features)
print("Features = {}".format(features))
print("species = {}".format(own_pred[0]))
```

```
Features = [[4.8, 2.2, 3.34, 1.5]]
species = Iris-versicolor
```