DATA WAREHOUSE AND DATA MINING

LAB DA-3

NAME:HRITHIK HEM SUNDAR.B

REGNO:19MID0021

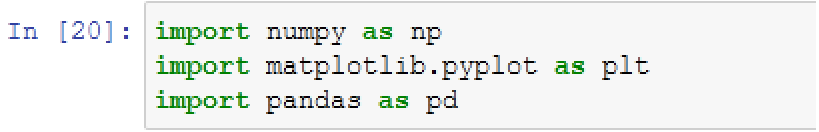
1.DECISION TREE USING PLAYTENNIS DATASET.

CODE:

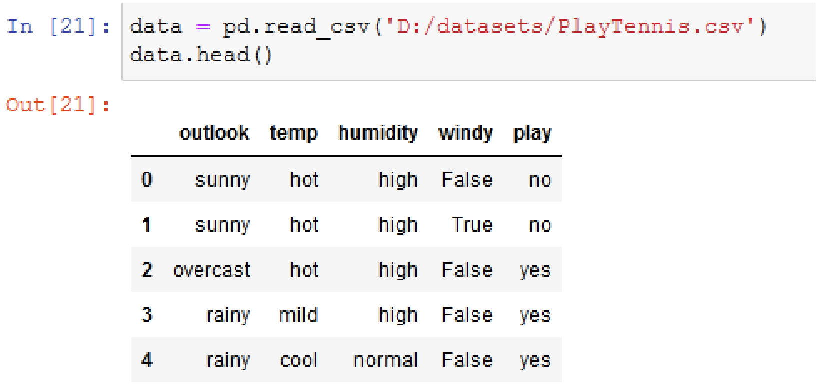
```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

data = pd.read_csv('D:/datasets/PlayTennis.csv')

data.head()

from sklearn.preprocessing import LabelEncoder

data['outlook']=LabelEncoder().fit_transform(data.outlook)

data['temp']=LabelEncoder().fit_transform(data.temp)

data['humidity']=LabelEncoder().fit_transform(data.humidity)

data['windy']=LabelEncoder().fit_transform(data.windy)

data.head()

feature_cols = ['outlook','temp','humidity','windy']

X = data.iloc[:,[0,1,2,3]].values

y = data.iloc[:,4].values

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =  train_test_split(X,y,test_size = 0.30, random_state= 47)

from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()

X_train = sc_X.fit_transform(X_train)
```

```
X_test = sc_X.transform(X_test)

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier()

classifier = classifier.fit(X_train,y_train)

y_pred = classifier.predict(X_test)#Accuracy

from sklearn import metrics

print('Accuracy Score:', metrics.accuracy_score(y_test,y_pred))

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(y_test, y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=classifier.classes_)

disp.plot()


import matplotlib.pyplot as plt

from sklearn import tree

fig = plt.figure(figsize=(10,15))

tree.plot_tree(classifier,feature_names=data.columns,class_names=data.columns,filled=True)
```

SCREENSHOTS WITH OUTPUT:

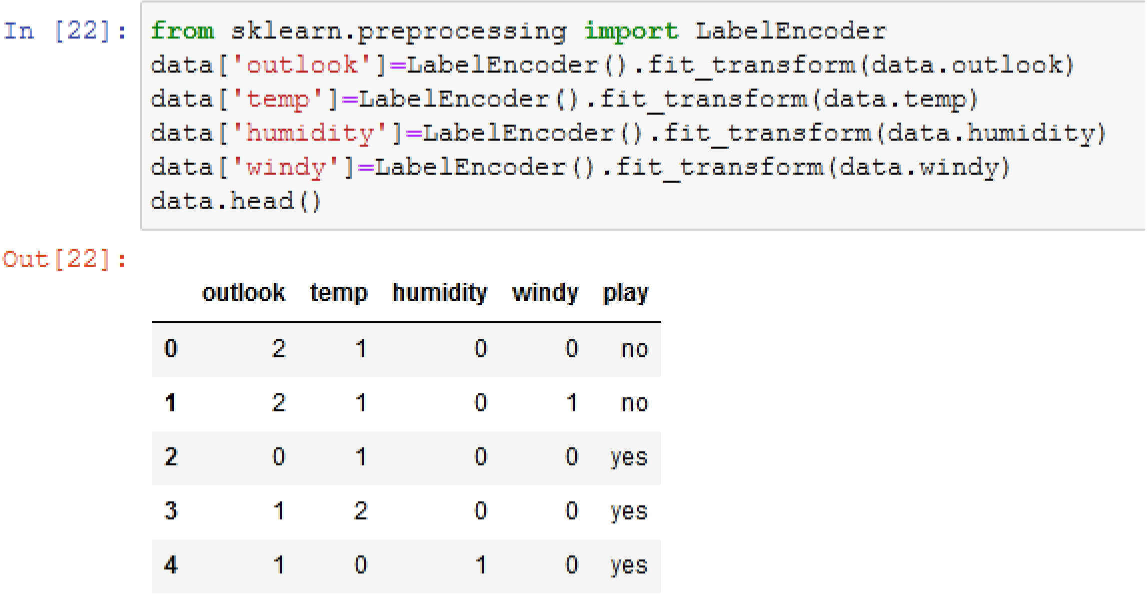


```
In [20]: import numpy as np
         import matplotlib.pyplot as plt
         import pandas as pd
```



```
In [21]: data = pd.read_csv('D:/datasets/PlayTennis.csv')
         data.head()

Out[21]:
```

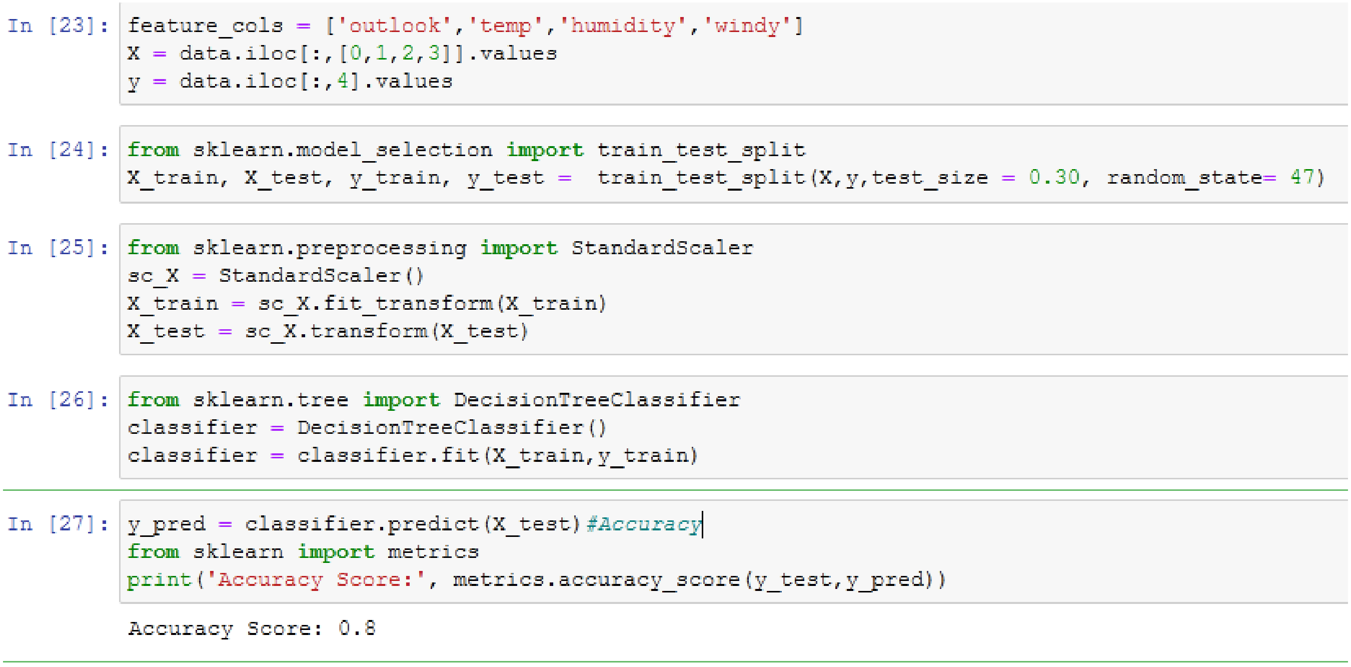
	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes



In [22]:
```
from sklearn.preprocessing import LabelEncoder
data['outlook']=LabelEncoder().fit_transform(data.outlook)
data['temp']=LabelEncoder().fit_transform(data.temp)
data['humidity']=LabelEncoder().fit_transform(data.humidity)
data['windy']=LabelEncoder().fit_transform(data.windy)
data.head()
```

Out[22]:

	outlook	temp	humidity	windy	play
0	2	1	0	0	no
1	2	1	0	1	no
2	0	1	0	0	yes
3	1	2	0	0	yes
4	1	0	1	0	yes



In [23]:
```
feature_cols = ['outlook','temp','humidity','windy']
X = data.iloc[:,[0,1,2,3]].values
y = data.iloc[:,4].values
```

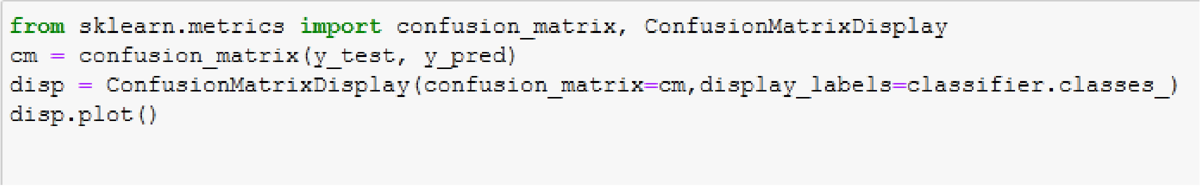
In [24]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =  train_test_split(X,y,test_size = 0.30, random_state= 47)
```

In [25]:
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

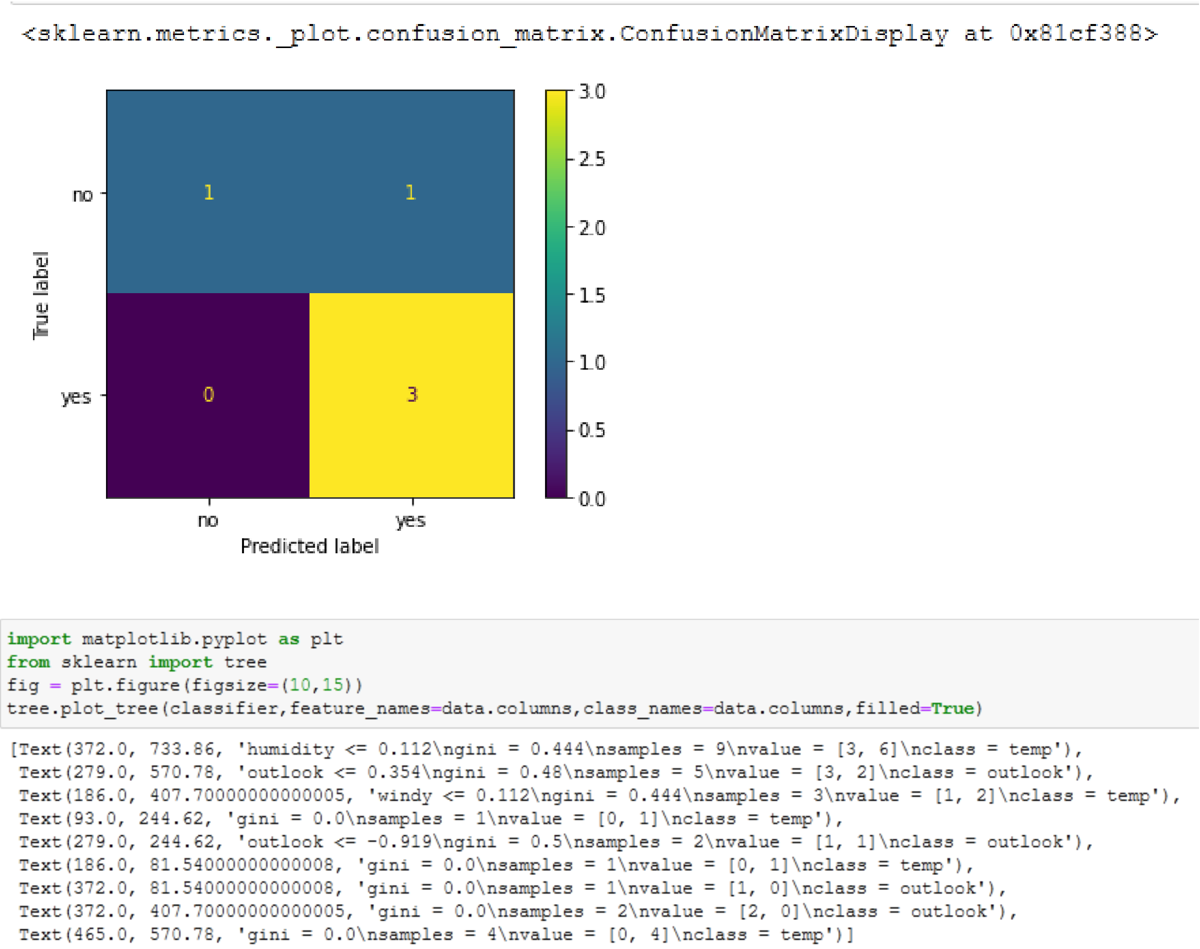
In [26]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier = classifier.fit(X_train,y_train)
```

In [27]:
```
y_pred = classifier.predict(X_test) #Accuracy
from sklearn import metrics
print('Accuracy Score:', metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy Score: 0.8
```



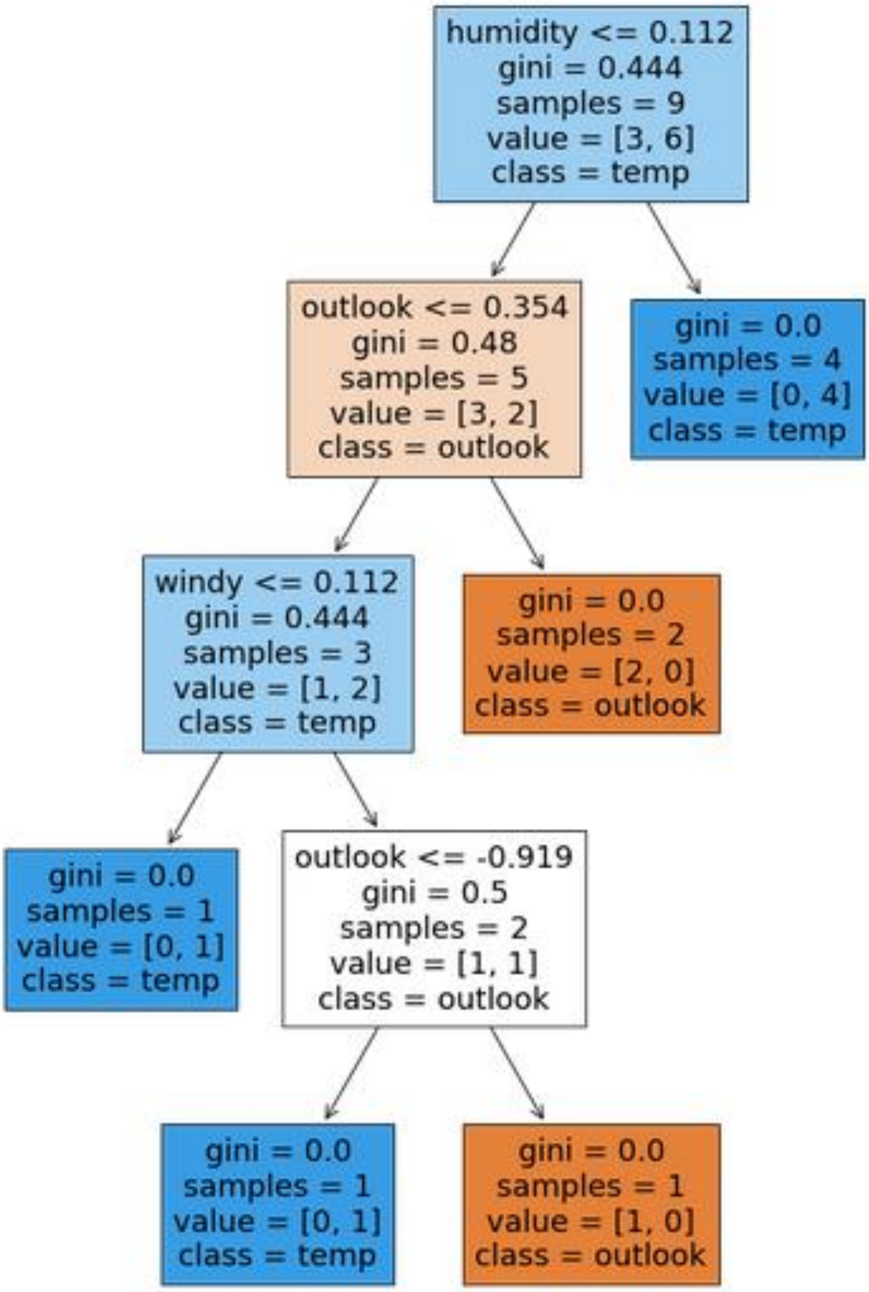
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=classifier.classes_)
disp.plot()
```



```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x81cf388>
```

```
import matplotlib.pyplot as plt
from sklearn import tree
fig = plt.figure(figsize=(10,15))
tree.plot_tree(classifier,feature_names=data.columns,class_names=data.columns,filled=True)
```

```
[Text(372.0, 733.86, 'humidity <= 0.112\ngini = 0.444\nsamples = 9\nvalue = [3, 6]\nclass = temp'),
 Text(279.0, 570.78, 'outlook <= 0.354\ngini = 0.48\nsamples = 5\nvalue = [3, 2]\nclass = outlook'),
 Text(186.0, 407.70000000000005, 'windy <= 0.112\ngini = 0.444\nsamples = 3\nvalue = [1, 2]\nclass = temp'),
 Text(93.0, 244.62, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]\nclass = temp'),
 Text(279.0, 244.62, 'outlook <= -0.919\ngini = 0.5\nsamples = 2\nvalue = [1, 1]\nclass = outlook'),
 Text(186.0, 81.54000000000008, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]\nclass = temp'),
 Text(372.0, 81.54000000000008, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]\nclass = outlook'),
 Text(372.0, 407.70000000000005, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]\nclass = outlook'),
 Text(465.0, 570.78, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]\nclass = temp')]
```

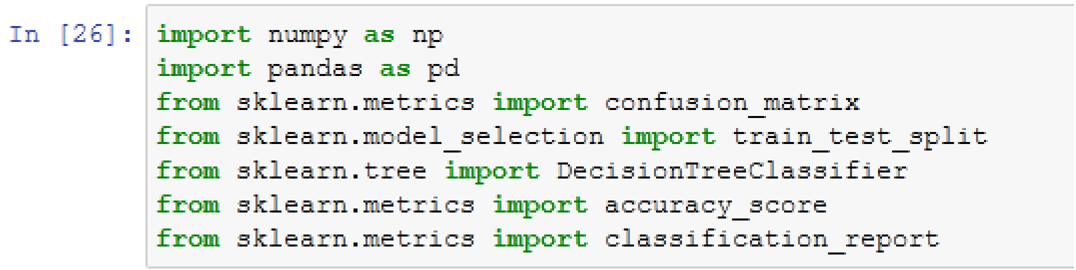

humidity <= 0.112
gini = 0.444
samples = 9
value = [3, 6]
class = temp
outlook <= 0.354
gini = 0.48
samples = 5
value = [3, 2]
class = outlook
gini = 0.0
samples = 4
value = [0, 4]
class = temp
windy <= 0.112
gini = 0.444
samples = 3
value = [1, 2]
class = temp
gini = 0.0
samples = 2
value = [2, 0]
class = outlook
gini = 0.0
samples = 1
value = [0, 1]
class = temp
outlook <= -0.919
gini = 0.5
samples = 2
value = [1, 1]
class = outlook
gini = 0.0
samples = 1
value = [0, 1]
class = temp
gini = 0.0
samples = 1
value = [1, 0]
class = outlook

2.DECISION TREE USING BALANCE SCALE DATASET

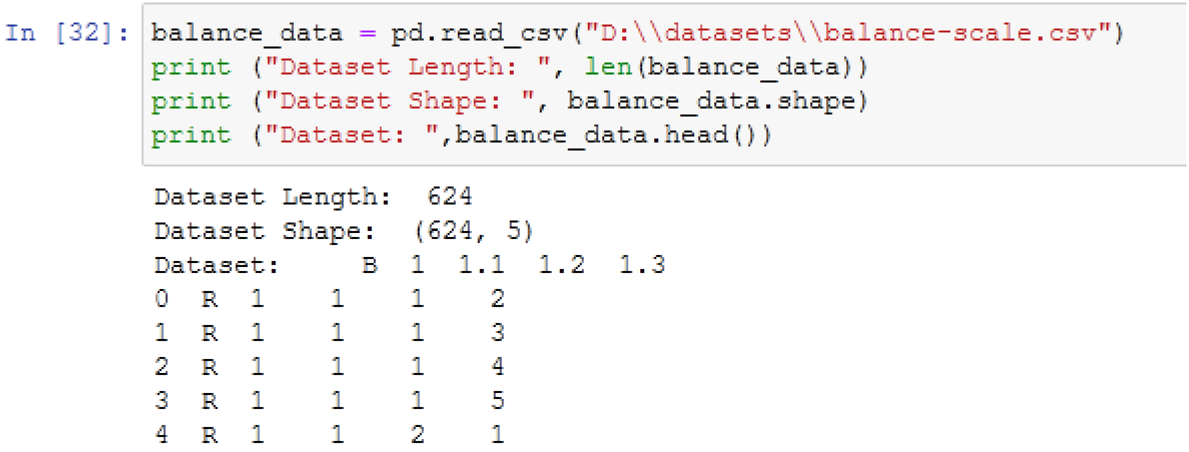
CODE:

```
import numpy as np

import pandas as pd

from sklearn.metrics import confusion_matrix

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score

from sklearn.metrics import classification_report

balance_data = pd.read_csv("D:\\datasets\\balance-scale.csv")

print ("Dataset Length: ", len(balance_data))

print ("Dataset Shape: ", balance_data.shape)

print ("Dataset: ",balance_data.head())

X = balance_data.values[:, 1:5]

Y = balance_data.values[:, 0]

X_train, X_test, y_train, y_test = train_test_split(

X, Y, test_size = 0.3, random_state = 100)

clf_gini = DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=3, min_samples_leaf=5)

clf_gini.fit(X_train, y_train)

clf_entropy = DecisionTreeClassifier(criterion = "entropy", random_state = 100,max_depth = 3, min_samples_leaf = 5)

clf_entropy.fit(X_train, y_train)

y_pred = clf_entropy.predict(X_test)

print("Predicted values:")

print(y_pred)

print("Confusion Matrix: ",
confusion_matrix(y_test, y_pred))

print ("Accuracy : ",

accuracy_score(y_test,y_pred)*100)

print("Report : ",

classification_report(y_test, y_pred))

data = balance_data

print("Results Using Gini Index:")

y_pred_gini = prediction(X_test, clf_gini)

cal_accuracy(y_test, y_pred_gini)

import matplotlib.pyplot as plt

from sklearn import tree

fig = plt.figure(figsize=(25,20))

tree.plot_tree(clf_entropy,feature_names=balance_data.columns,class_names=balance_data.columns,filled=True)

fig = plt.figure(figsize=(25,20))

tree.plot_tree(clf_gini,feature_names=balance_data.columns,class_names=balance_data.columns,filled=True)
```

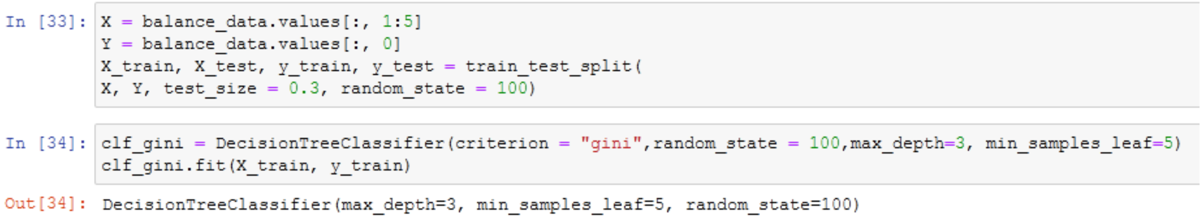
SCREENSHOTS WITH OUTPUT:



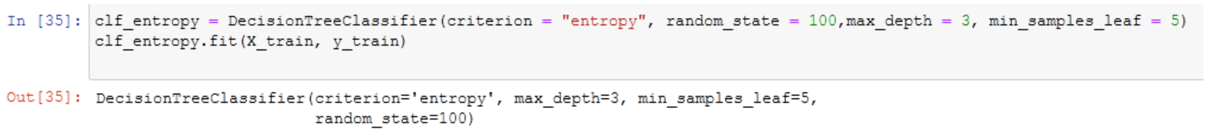
```
In [26]: import numpy as np
         import pandas as pd
         from sklearn.metrics import confusion_matrix
         from sklearn.model_selection import train_test_split
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.metrics import accuracy_score
         from sklearn.metrics import classification_report
```



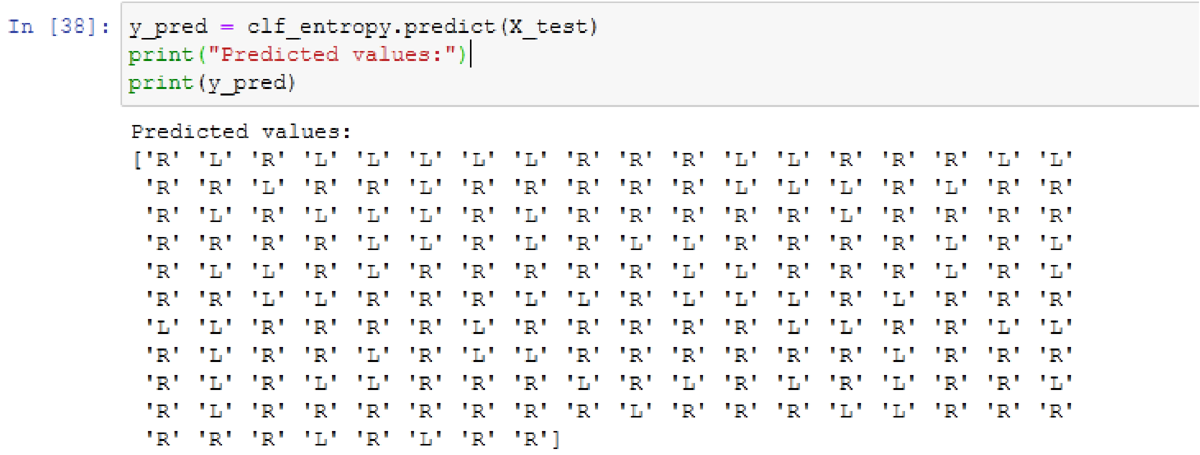
```
In [32]: balance_data = pd.read_csv("D:\\datasets\\balance-scale.csv")
         print ("Dataset Length: ", len(balance_data))
         print ("Dataset Shape: ", balance_data.shape)
         print ("Dataset: ",balance_data.head())

         Dataset Length:  624
         Dataset Shape:  (624, 5)
         Dataset:      B  1  1.1  1.2  1.3
         0  R  1    1    1    2
         1  R  1    1    1    3
         2  R  1    1    1    4
         3  R  1    1    1    5
         4  R  1    1    2    1
```



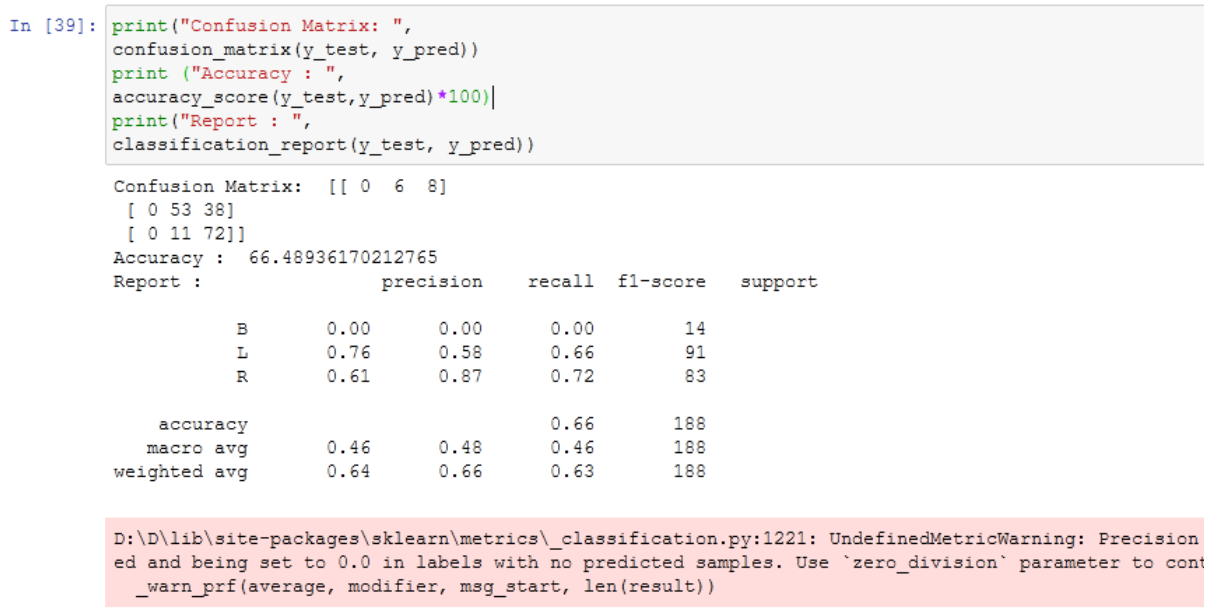
```
In [33]: X = balance_data.values[:, 1:5]
         Y = balance_data.values[:, 0]
         X_train, X_test, y_train, y_test = train_test_split(
         X, Y, test_size = 0.3, random_state = 100)

In [34]: clf_gini = DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=3, min_samples_leaf=5)
         clf_gini.fit(X_train, y_train)

Out[34]: DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```



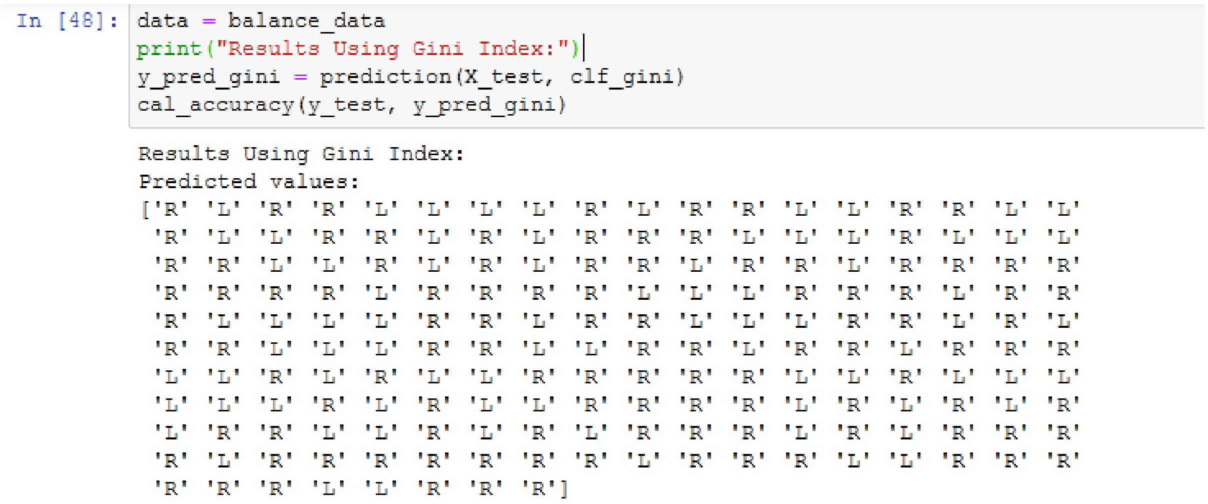
```
In [35]: clf_entropy = DecisionTreeClassifier(criterion = "entropy", random_state = 100,max_depth = 3, min_samples_leaf = 5)
         clf_entropy.fit(X_train, y_train)

Out[35]: DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samples_leaf=5,
                                random_state=100)
```



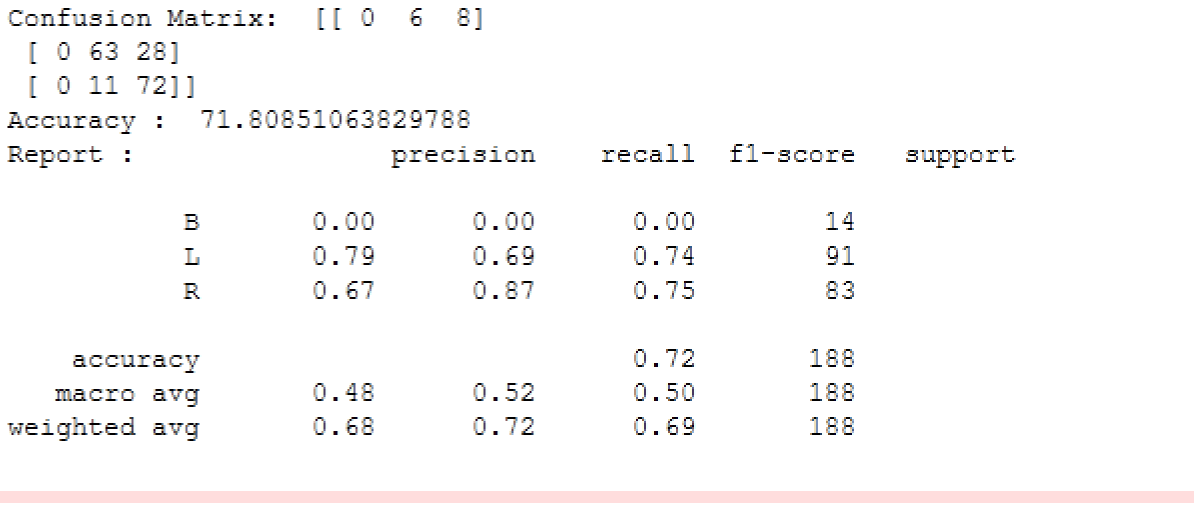
```
In [38]: y_pred = clf_entropy.predict(X_test)
         print("Predicted values:")
         print(y_pred)

         Predicted values:
         ['R' 'L' 'R' 'L' 'L' 'L' 'L' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'R' 'R' 'L' 'L'
          'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'R'
          'R' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R'
          'R' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'L'
          'R' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'L' 'R' 'R' 'R' 'L' 'R' 'L'
          'R' 'R' 'L' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'R' 'R'
          'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'L' 'R' 'R' 'L' 'L'
          'R' 'L' 'R' 'R' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R'
          'R' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L'
          'R' 'L' 'R' 'R' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'R' 'R'
          'R' 'R' 'R' 'L' 'R' 'L' 'R' 'R']
```



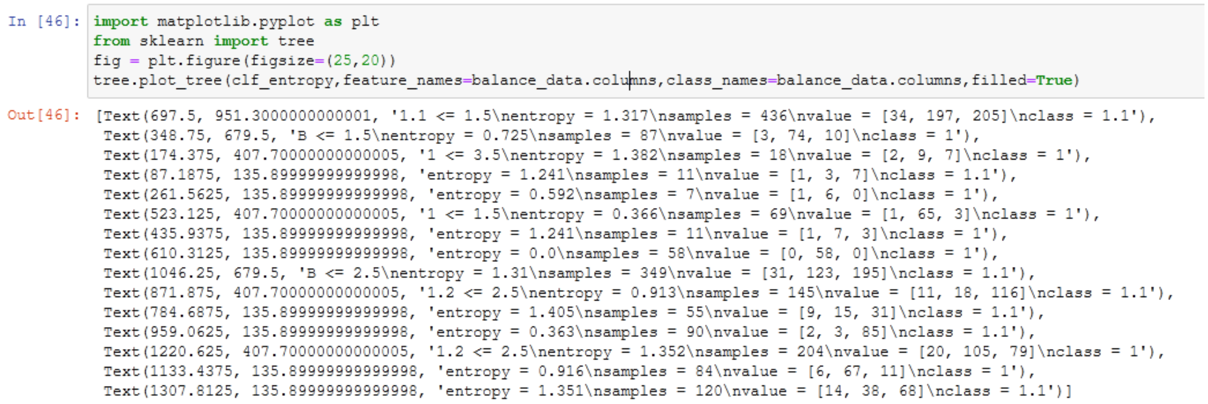
```
In [39]: print("Confusion Matrix: ",
         confusion_matrix(y_test, y_pred))
         print ("Accuracy : ",
         accuracy_score(y_test,y_pred)*100)
         print("Report : ",
         classification_report(y_test, y_pred))

         Confusion Matrix:  [[ 0  6  8]
          [ 0 53 38]
          [ 0 11 72]]
         Accuracy :  66.48936170212765
         Report :                precision    recall  f1-score   support

                    B       0.00      0.00      0.00        14
                    L       0.76      0.58      0.66        91
                    R       0.61      0.87      0.72        83

             accuracy                           0.66       188
            macro avg       0.46      0.48      0.46       188
         weighted avg       0.64      0.66      0.63       188

         D:\D\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision
         ed and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to cont
           _warn_prf(average, modifier, msg_start, len(result))
```



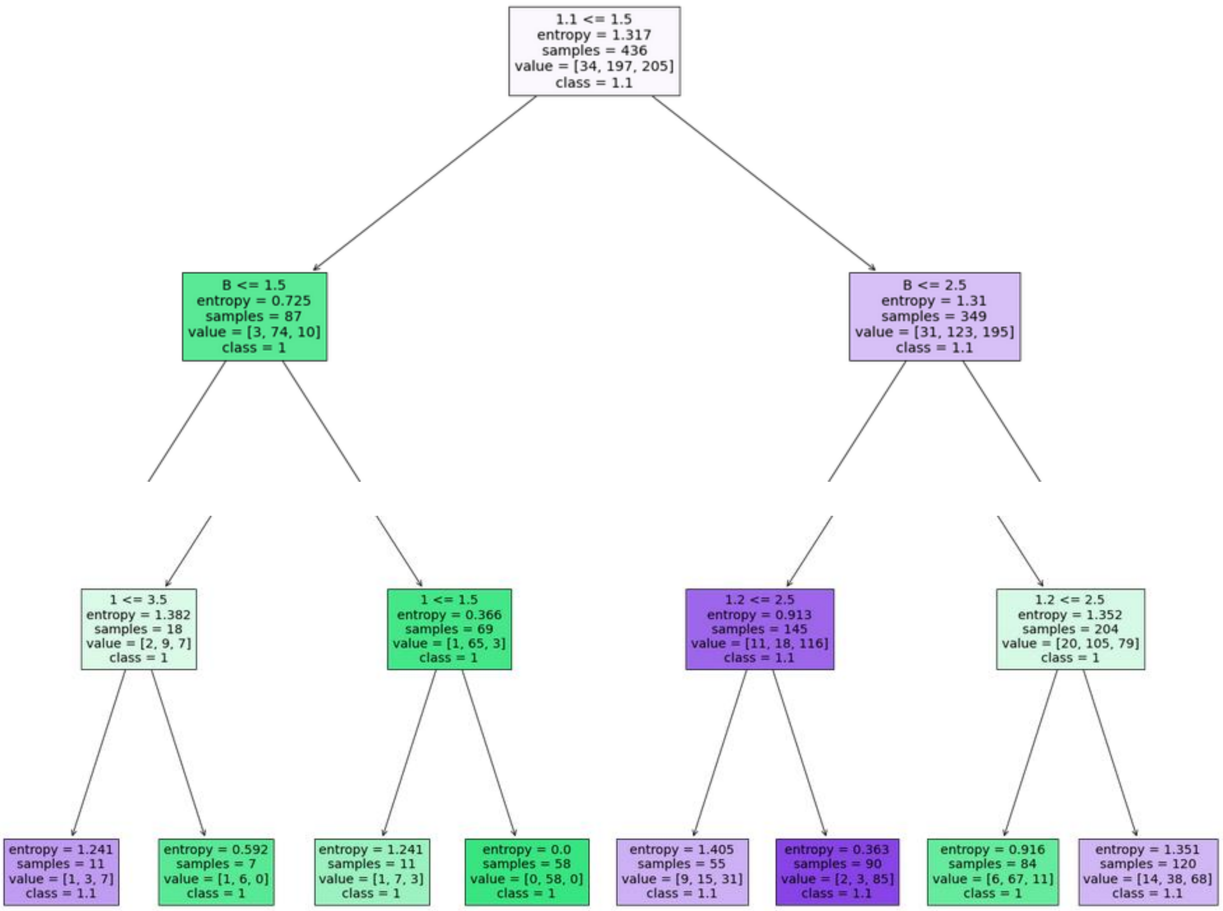
```
In [48]: data = balance_data
         print("Results Using Gini Index:")
         y_pred_gini = prediction(X_test, clf_gini)
         cal_accuracy(y_test, y_pred_gini)

         Results Using Gini Index:
         Predicted values:
         ['R' 'L' 'R' 'R' 'L' 'L' 'L' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'L' 'L'
          'R' 'L' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L'
          'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'R'
          'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'R' 'L' 'R' 'R'
          'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
          'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R'
          'L' 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'L'
          'L' 'L' 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'L' 'R' 'L' 'R'
          'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'R' 'R' 'R'
          'R' 'L' 'R' 'R' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'R' 'R'
          'R' 'R' 'R' 'L' 'L' 'R' 'R' 'R']
```

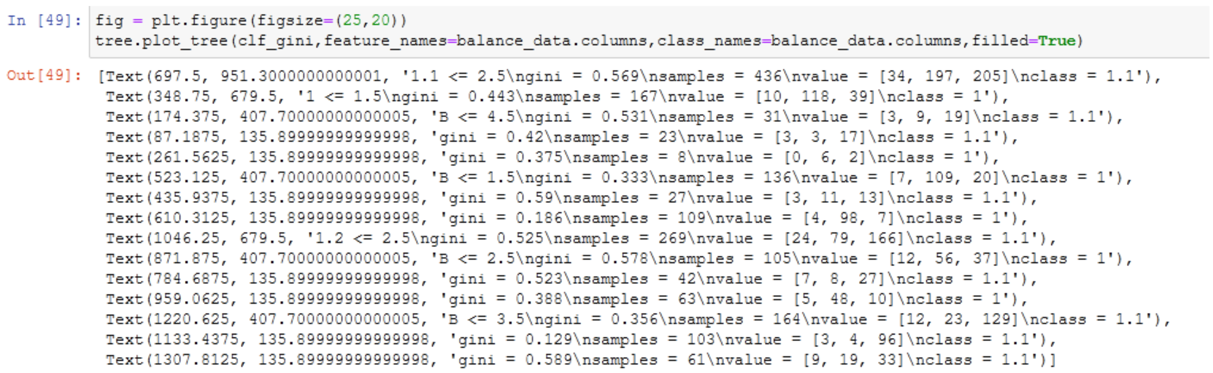


```
Confusion Matrix:  [[ 0  6  8]
 [ 0 63 28]
 [ 0 11 72]]
Accuracy :  71.80851063829788
Report :                precision    recall  f1-score   support

           B       0.00      0.00      0.00        14
           L       0.79      0.69      0.74        91
           R       0.67      0.87      0.75        83

    accuracy                           0.72       188
   macro avg       0.48      0.52      0.50       188
weighted avg       0.68      0.72      0.69       188
```

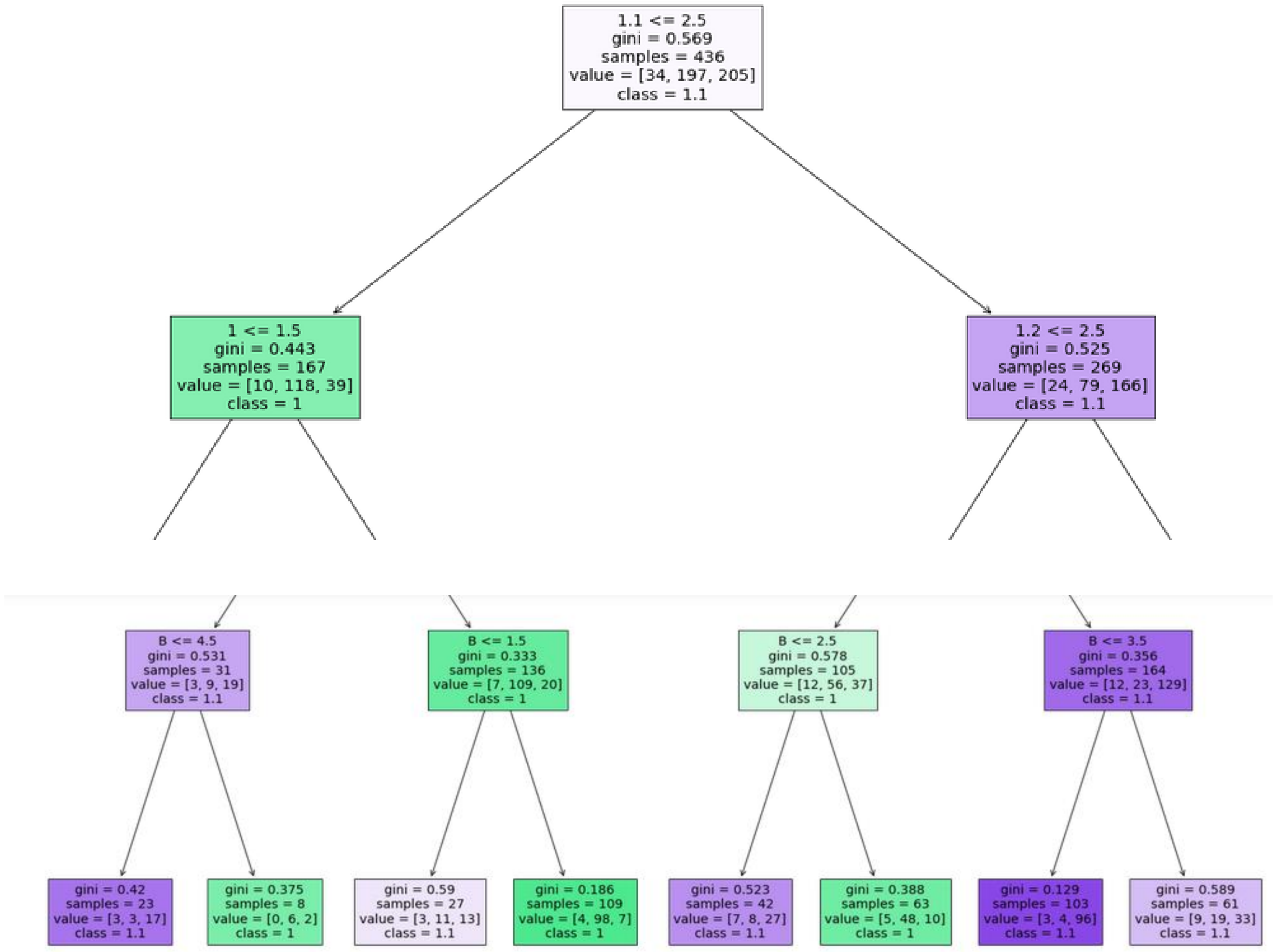


```
In [46]: import matplotlib.pyplot as plt
         from sklearn import tree
         fig = plt.figure(figsize=(25,20))
         tree.plot_tree(clf_entropy,feature_names=balance_data.columns,class_names=balance_data.columns,filled=True)

Out[46]: [Text(697.5, 951.3000000000001, '1.1 <= 1.5\nentropy = 1.317\nsamples = 436\nvalue = [34, 197, 205]\nclass = 1.1'),
          Text(348.75, 679.5, 'B <= 1.5\nentropy = 0.725\nsamples = 87\nvalue = [3, 74, 10]\nclass = 1'),
          Text(174.375, 407.70000000000005, '1 <= 3.5\nentropy = 1.382\nsamples = 18\nvalue = [2, 9, 7]\nclass = 1'),
          Text(87.1875, 135.89999999999998, 'entropy = 1.241\nsamples = 11\nvalue = [1, 3, 7]\nclass = 1.1'),
          Text(261.5625, 135.89999999999998, 'entropy = 0.592\nsamples = 7\nvalue = [1, 6, 0]\nclass = 1'),
          Text(523.125, 407.70000000000005, '1 <= 1.5\nentropy = 0.366\nsamples = 69\nvalue = [1, 65, 3]\nclass = 1'),
          Text(435.9375, 135.89999999999998, 'entropy = 1.241\nsamples = 11\nvalue = [1, 7, 3]\nclass = 1'),
          Text(610.3125, 135.89999999999998, 'entropy = 0.0\nsamples = 58\nvalue = [0, 58, 0]\nclass = 1'),
          Text(1046.25, 679.5, 'B <= 2.5\nentropy = 1.31\nsamples = 349\nvalue = [31, 123, 195]\nclass = 1.1'),
          Text(871.875, 407.70000000000005, '1.2 <= 2.5\nentropy = 0.913\nsamples = 145\nvalue = [11, 18, 116]\nclass = 1.1'),
          Text(784.6875, 135.89999999999998, 'entropy = 1.405\nsamples = 55\nvalue = [9, 15, 31]\nclass = 1.1'),
          Text(959.0625, 135.89999999999998, 'entropy = 0.363\nsamples = 90\nvalue = [2, 3, 85]\nclass = 1.1'),
          Text(1220.625, 407.70000000000005, '1.2 <= 2.5\nentropy = 1.352\nsamples = 204\nvalue = [20, 105, 79]\nclass = 1'),
          Text(1133.4375, 135.89999999999998, 'entropy = 0.916\nsamples = 84\nvalue = [6, 67, 11]\nclass = 1'),
          Text(1307.8125, 135.89999999999998, 'entropy = 1.351\nsamples = 120\nvalue = [14, 38, 68]\nclass = 1.1')]
```





```
In [49]: fig = plt.figure(figsize=(25,20))
         tree.plot_tree(clf_gini,feature_names=balance_data.columns,class_names=balance_data.columns,filled=True)

Out[49]: [Text(697.5, 951.3000000000001, '1.1 <= 2.5\ngini = 0.569\nsamples = 436\nvalue = [34, 197, 205]\nclass = 1.1'),
          Text(348.75, 679.5, '1 <= 1.5\ngini = 0.443\nsamples = 167\nvalue = [10, 118, 39]\nclass = 1'),
          Text(174.375, 407.70000000000005, 'B <= 4.5\ngini = 0.531\nsamples = 31\nvalue = [3, 9, 19]\nclass = 1.1'),
          Text(87.1875, 135.89999999999998, 'gini = 0.42\nsamples = 23\nvalue = [3, 3, 17]\nclass = 1.1'),
          Text(261.5625, 135.89999999999998, 'gini = 0.375\nsamples = 8\nvalue = [0, 6, 2]\nclass = 1'),
          Text(523.125, 407.70000000000005, 'B <= 1.5\ngini = 0.333\nsamples = 136\nvalue = [7, 109, 20]\nclass = 1'),
          Text(435.9375, 135.89999999999998, 'gini = 0.59\nsamples = 27\nvalue = [3, 11, 13]\nclass = 1.1'),
          Text(610.3125, 135.89999999999998, 'gini = 0.186\nsamples = 109\nvalue = [4, 98, 7]\nclass = 1'),
          Text(1046.25, 679.5, '1.2 <= 2.5\ngini = 0.525\nsamples = 269\nvalue = [24, 79, 166]\nclass = 1.1'),
          Text(871.875, 407.70000000000005, 'B <= 2.5\ngini = 0.578\nsamples = 105\nvalue = [12, 56, 37]\nclass = 1'),
          Text(784.6875, 135.89999999999998, 'gini = 0.523\nsamples = 42\nvalue = [7, 8, 27]\nclass = 1.1'),
          Text(959.0625, 135.89999999999998, 'gini = 0.388\nsamples = 63\nvalue = [5, 48, 10]\nclass = 1'),
          Text(1220.625, 407.70000000000005, 'B <= 3.5\ngini = 0.356\nsamples = 164\nvalue = [12, 23, 129]\nclass = 1.1'),
          Text(1133.4375, 135.89999999999998, 'gini = 0.129\nsamples = 103\nvalue = [3, 4, 96]\nclass = 1.1'),
          Text(1307.8125, 135.89999999999998, 'gini = 0.589\nsamples = 61\nvalue = [9, 19, 33]\nclass = 1.1')]
```


1.1 <= 2.5
gini = 0.569
samples = 436
value = [34, 197, 205]
class = 1.1
1 <= 1.5
gini = 0.443
samples = 167
value = [10, 118, 39]
class = 1
1.2 <= 2.5
gini = 0.525
samples = 269
value = [24, 79, 166]
class = 1.1
B <= 4.5
gini = 0.531
samples = 31
value = [3, 9, 19]
class = 1.1
B <= 1.5
gini = 0.333
samples = 136
value = [7, 109, 20]
class = 1
B <= 2.5
gini = 0.578
samples = 105
value = [12, 56, 37]
class = 1
B <= 3.5
gini = 0.356
samples = 164
value = [12, 23, 129]
class = 1.1
gini = 0.42
samples = 23
value = [3, 3, 17]
class = 1.1
gini = 0.375
samples = 8
value = [0, 6, 2]
class = 1
gini = 0.59
samples = 27
value = [3, 11, 13]
class = 1.1
gini = 0.186
samples = 109
value = [4, 98, 7]
class = 1
gini = 0.523
samples = 42
value = [7, 8, 27]
class = 1.1
gini = 0.388
samples = 63
value = [5, 48, 10]
class = 1
gini = 0.129
samples = 103
value = [3, 4, 96]
class = 1.1
gini = 0.589
samples = 61
value = [9, 19, 33]
class = 1.1