task22

October 1, 2022

[3]:
```
def plot_colors(student_id):
    color1 = "#"+student_id[1:]
    color2 = "#"+str(hex( int("FFFFFF" ,16) - int(student_id[1:],16)))[2:]
    return color1 , color2
```

[4]:
```
psid = "2100224"
color1,color2 = plot_colors(psid)
```

[5]:
```
import pandas as pd
```

[6]:
```
data = pd.read_excel("data.xlsx")
data
```

[6]:

	Feature 1	Feature 2	Feature 3	Feature 4	Label
0	-8.612445	-9.729022	-30.050765	90.482367	1
1	0.902390	-0.316833	0.699654	0.337164	0
2	-25.307502	-20.669486	-62.605756	250.631483	1
3	-21.844937	-4.288478	-14.848927	235.177479	0
4	26.185707	-24.207999	-76.018547	-267.134888	1
…	…	…	…	…	…
8395	0.321145	-2.728815	-8.271032	-7.367157	0
8396	-21.523594	-5.790952	-19.613791	230.112038	0
8397	-24.358283	6.796865	19.012739	234.523391	0
8398	17.553608	24.473744	67.689126	-167.704393	1
8399	-6.561204	-20.195203	-65.790356	65.981372	0

[8400 rows x 5 columns]

[7]:
```
df1 = data[data['Label'] == 1]
df1
```

[7]:

	Feature 1	Feature 2	Feature 3	Feature 4	Label
0	-8.612445	-9.729022	-30.050765	90.482367	1
2	-25.307502	-20.669486	-62.605756	250.631483	1
4	26.185707	-24.207999	-76.018547	-267.134888	1
7	18.826052	-3.268000	-11.162893	-165.038169	1
9	2.474554	27.845189	82.562990	-20.989719	1

```
…          …         …         …          …    …
8390  12.956842 -12.984984 -38.551356 -122.411677      1
8391 -14.328434   5.219075  13.113559  141.002923      1
8392 -27.131936  21.599224  63.714169  262.318544      1
8393 -13.953086   4.745433  13.904520  140.452241      1
8398  17.553608  24.473744  67.689126 -167.704393      1

[3400 rows x 5 columns]
```

[8]:
```
df0 = data[data.Label == 0]
df0
```

[8]:
```
      Feature 1  Feature 2  Feature 3   Feature 4  Label
1      0.902390  -0.316833   0.699654    0.337164      0
3    -21.844937  -4.288478 -14.848927  235.177479      0
5      1.455472   1.453604   1.447083   -0.262329      0
6     -1.732871  -2.405579  -9.509719   30.246861      0
8      0.475492  -0.996478  -3.936492  -13.638168      0
…          …         …         …          …    …
8394   5.334331   5.325826  14.445657  -41.661240      0
8395   0.321145  -2.728815  -8.271032   -7.367157      0
8396 -21.523594  -5.790952 -19.613791  230.112038      0
8397 -24.358283   6.796865  19.012739  234.523391      0
8399  -6.561204 -20.195203 -65.790356   65.981372      0

[5000 rows x 5 columns]
```

[62]:
```
def probs(x):
    y = (x['Label'].value_counts()) / len(x) * 100
    return y

dta = data

probs(dta)
```

[62]:
```
0    59.52381
1    40.47619
Name: Label, dtype: float64
```

[10]:
```
def regSamp(d,q):
    y = pd.DataFrame.sample(d,q)
    return y
q = 1000
dataset2 = regSamp(data,q)
df0r = dataset2[dataset2.Label == 0]
df1r = dataset2[dataset2.Label == 1]
```

[11]:
```
probs(dataset2)
```

[11]:
```
0    58.3
1    41.7
Name: Label, dtype: float64
```

[12]:
```
def stratSamp(d,q):
    y = data.groupby('Label', group_keys=False).apply(lambda x: x.sample(frac =␣
 ↪.11904761904))
    return y

q = 1000
dataset3 =stratSamp(data,q)
```

[13]:
```
probs(dataset3)
```

[13]:
```
0    59.5
1    40.5
Name: Label, dtype: float64
```

[14]:
```
dataset3.cov()
```

[14]:

	Feature 1	Feature 2	Feature 3	Feature 4	Label
Feature 1	216.177858	-10.979383	-34.603409	-2159.412624	2.683170
Feature 2	-10.979383	212.553441	637.692865	112.527011	0.267821
Feature 3	-34.603409	637.692865	1919.517993	354.459262	0.811568
Feature 4	-2159.412624	112.527011	354.459262	21638.091978	-26.986461
Label	2.683170	0.267821	0.811568	-26.986461	0.241216

[15]:
```
dataset4 = dataset3[['Feature 1','Feature 2', 'Label']]
dataset4
```

[15]:

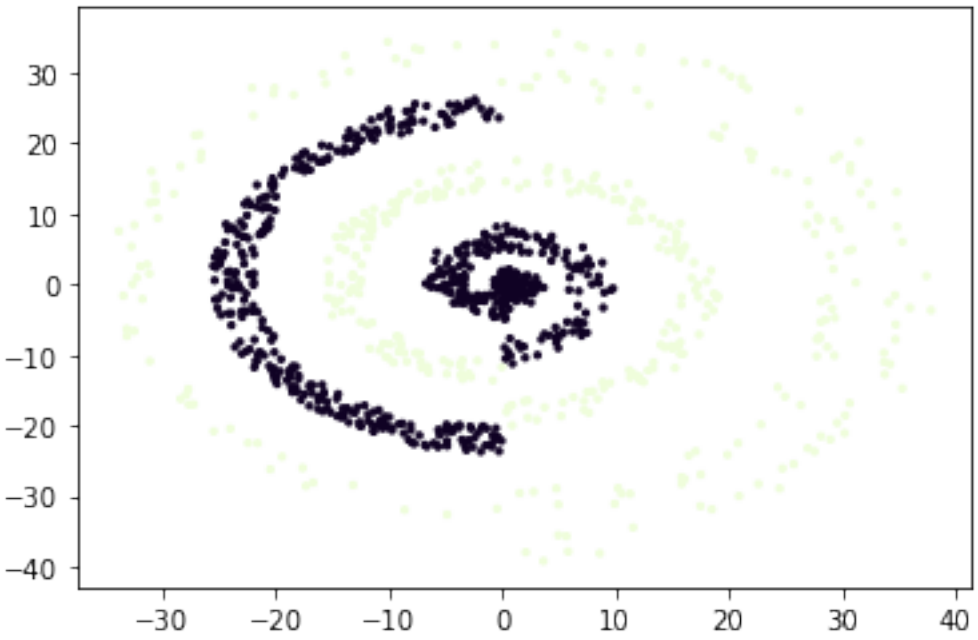
	Feature 1	Feature 2	Label
1929	-1.293707	23.505185	0
6332	1.833443	7.149872	0
5350	-2.232512	-2.441199	0
867	-19.468731	-10.620176	0
1465	-3.316252	-22.747998	0
…	…	…	…
454	-30.342330	9.357923	1
590	29.577231	-19.621972	1
2665	12.287519	33.468853	1
5822	21.786561	18.821661	1
5349	-13.281972	6.963911	1

[1000 rows x 3 columns]

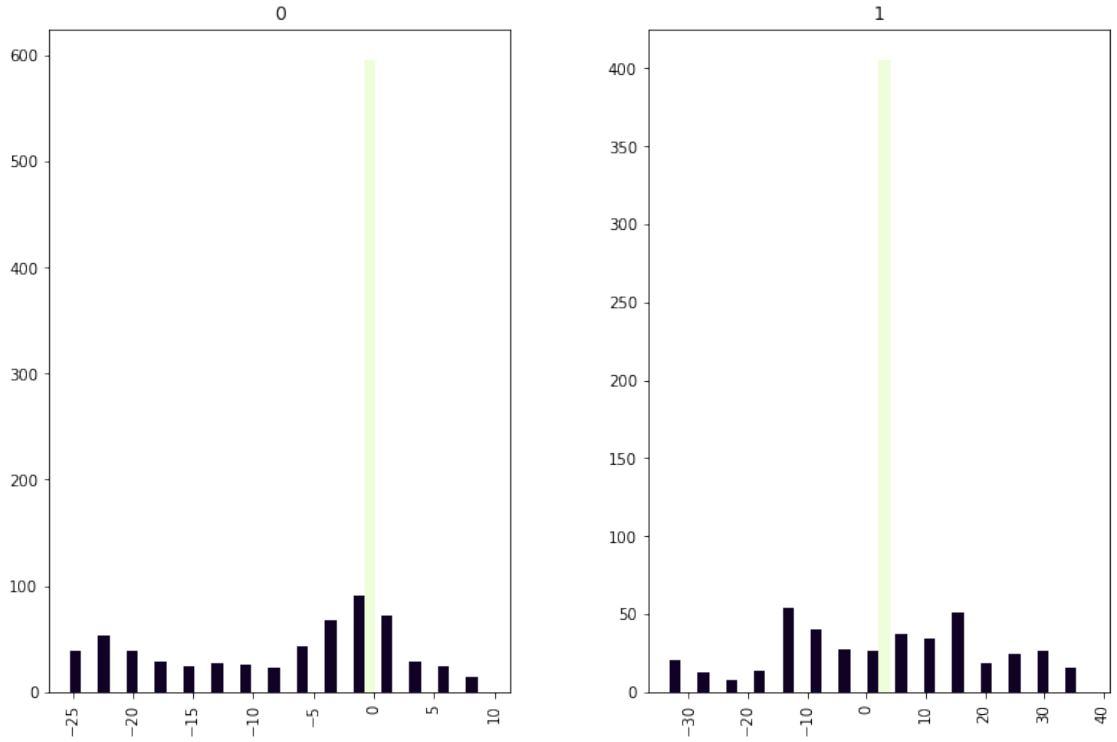
[17]:
```
import matplotlib.pyplot as plt
import numpy as np
c = []
colNames = list(dataset4.columns)
for index, row in dataset4.iterrows():
    if row['Label'] == 0:
        c.append(color1)
    else:
        c.append(color2)
dataset4['Color']=c

plt.scatter(dataset4[colNames[0]],dataset4[colNames[1]], c=dataset4.Color, s =␣
↪5)
```

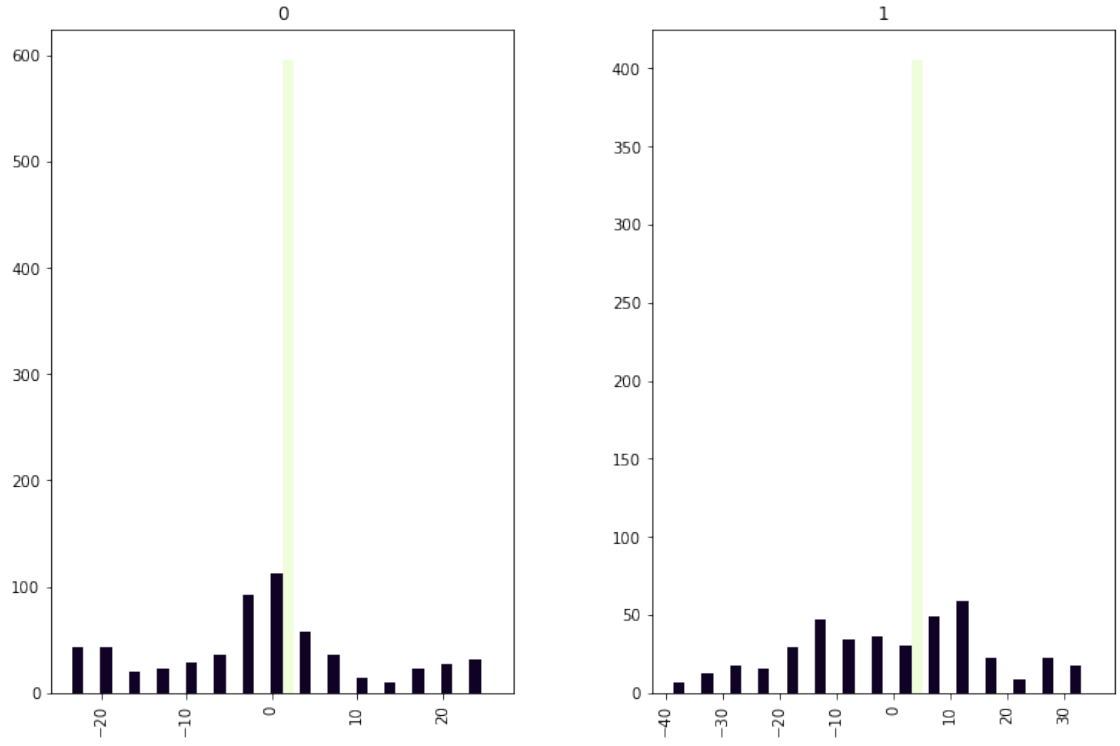
```
C:\Users\saima\AppData\Local\Temp\ipykernel_10216\1272977297.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dataset4['Color']=c
```

[17]:
```
<matplotlib.collections.PathCollection at 0x1aa358e6190>
```



[18]:
```
d4f1 = dataset4[['Feature 1', 'Label']]
d4f2 = dataset4[['Feature 2', 'Label']]
d4f1.hist(by='Label',color = [color1,color2],figsize=[12, 8], bins=15)
d4f2.hist(by='Label',color = [color1,color2], figsize=[12, 8], bins=15)

plt.show()
```





[50]:
```
from sklearn.model_selection import StratifiedShuffleSplit
def task3(d,q):
    split = StratifiedShuffleSplit(n_splits = 1, test_size = 1 - (q/len(d)))
    for train_index, test_index in split.split(d,d['Label']):
        data0 = d.loc[train_index]
    return data0
ds3= task3(data,1000)
#dds = task3(dataset4,10)
ds3
```

[50]:

	Feature 1	Feature 2	Feature 3	Feature 4	Label
5454	9.552762	1.108415	0.855754	-71.877656	0
7086	6.017297	12.922220	42.479529	-61.115280	1
1111	-19.888929	-14.641213	-41.258410	188.372671	0
5768	1.426530	-2.495459	-8.270337	-5.219081	0
8191	-0.272313	-22.965549	-67.398963	7.891908	0
…	…	…	…	…	…
5152	10.061940	-27.906883	-89.143509	-99.551655	1
3865	-11.777522	6.091903	16.179624	136.840028	1
3719	27.235293	-12.309822	-37.277280	-269.727126	1
1678	-15.951045	-15.582476	-46.691442	178.993454	0
5885	-15.299125	-2.226072	-7.809841	141.163477	1

```
[1000 rows x 5 columns]
```

[49]:
```
import numpy as np
>>> from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
dataset4
```

[49]:
```
      Feature 1  Feature 2  Label    Color
1929  -1.293707  23.505185      0  #100224
6332   1.833443   7.149872      0  #100224
5350  -2.232512  -2.441199      0  #100224
867  -19.468731 -10.620176      0  #100224
1465  -3.316252 -22.747998      0  #100224
…          …          …    …       …
454  -30.342330   9.357923      1  #effddb
590   29.577231 -19.621972      1  #effddb
2665  12.287519  33.468853      1  #effddb
5822  21.786561  18.821661      1  #effddb
5349 -13.281972   6.963911      1  #effddb

[1000 rows x 4 columns]
```

[23]:
```
def newstratSamp(d,q):
    y = d.groupby('Label', group_keys=False).apply(lambda x: x.sample(frac = .
 ↪7))
    return y
```

[94]:
```
from sklearn.model_selection import train_test_split
X = dataset4[['Feature 1', 'Feature 2']]
xe = dataset4[['Feature 1', 'Feature 2']]
y =dataset4['Label']
size=.3
def seven(x,y,size):
    x_train, x_test, y_train, y_test = train_test_split(x, y ,test_size = size)


#X_train, X_test, y_train, y_test = train_test_split(X, y ,test_size = size)
```

```
  Input In [94]
    return x_test as X_TEST
                  ^
SyntaxError: invalid syntax
```

[96]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(max_depth=3)
```

```
clf = clf.fit(X_train,y_train)
```

[188]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(max_depth=3)
clf = clf.fit(X_train,y_train)
from sklearn.metrics import classification_report
predictions = clf.predict(X_test)
print(classification_report(y_test,predictions,target_names = ['Class 0','Class␣
↪1']))
from sklearn.metrics import accuracy_score
accuracy_score(y_test,predictions)
X_test
```

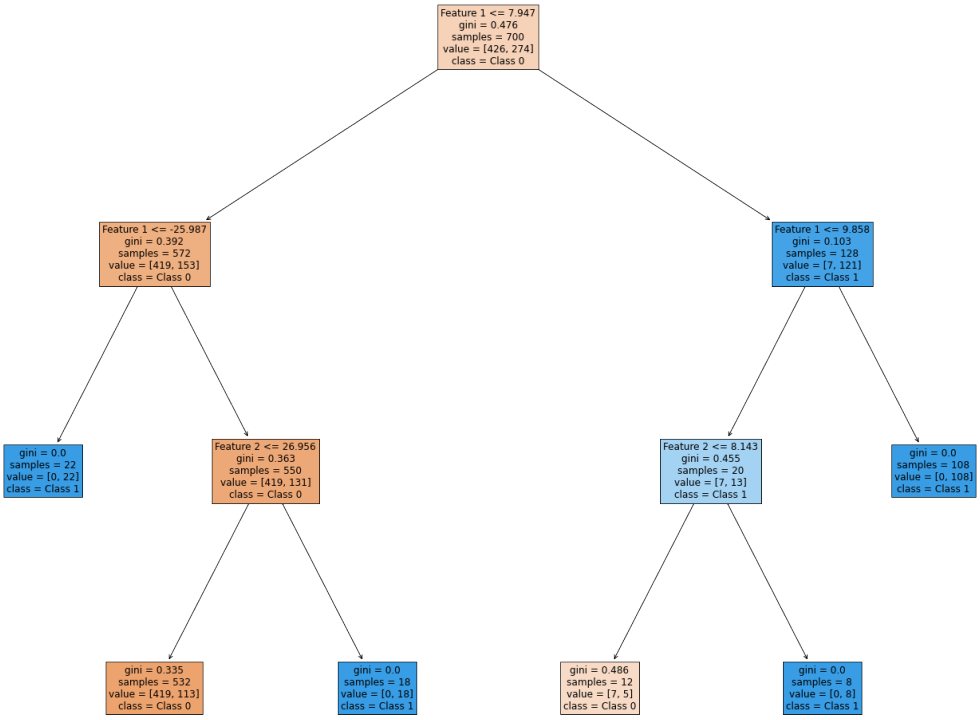
```
              precision    recall  f1-score   support

     Class 0       0.74      1.00      0.85       169
     Class 1       1.00      0.56      0.72       131

    accuracy                           0.81       300
   macro avg       0.87      0.78      0.78       300
weighted avg       0.86      0.81      0.79       300
```

[188]:

	Feature 1	Feature 2
1700	-33.070207	-7.318502
7614	30.276667	-18.580268
7787	-21.932054	-3.035472
238	2.215298	-1.344041
3902	1.034994	-1.314554
…	…	…
697	-2.926842	-22.107159
6814	-21.906774	-0.119377
3319	15.390055	7.235376
6912	15.432638	7.149995
2387	-10.125536	24.196573

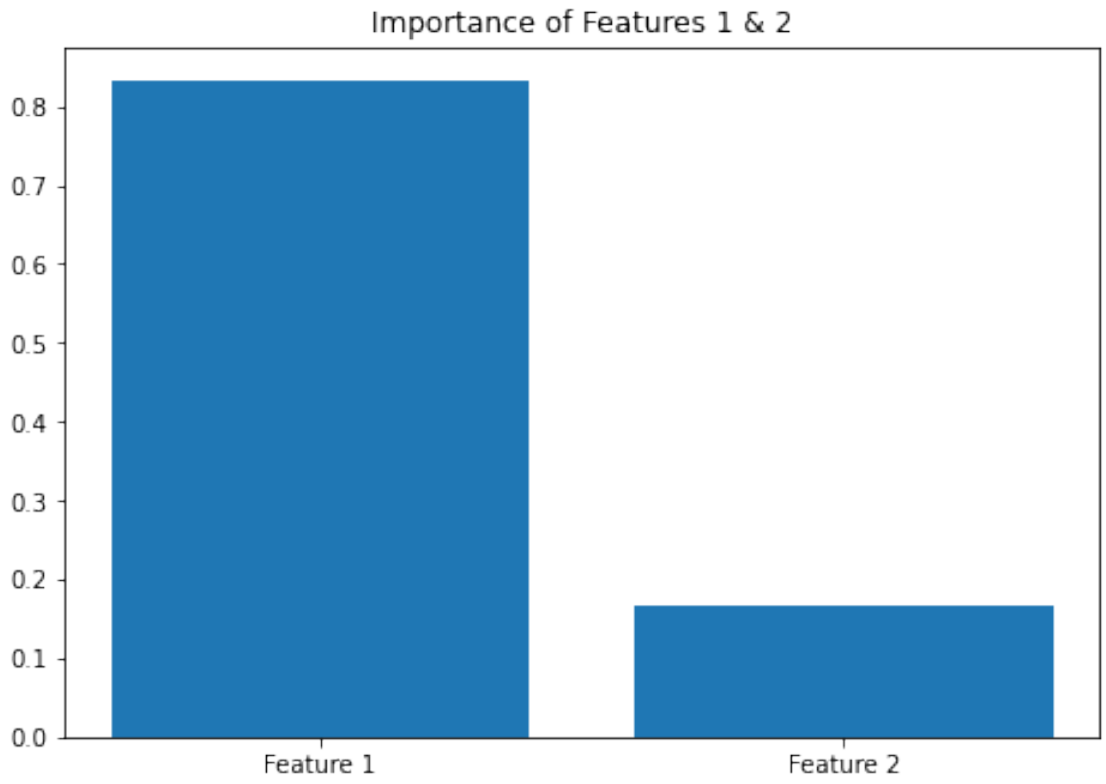
[300 rows x 2 columns]

[98]:
```
from sklearn import tree
from matplotlib import pyplot as plt
clf.get_params()
feature_names=X.columns
fig = plt.figure(figsize = (25,20))
_ = tree.plot_tree(clf,
                   feature_names = feature_names,
                   class_names = {0:'Class 0', 1:'Class 1'},
                   filled = True,
                   fontsize = 12)
```



[99]:
```
feature_names = X_test.columns
feature_names
imp=clf.feature_importances_
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(feature_names,imp)
ax.set_title('Importance of Features 1 & 2')
```

[99]:
```
Text(0.5, 1.0, 'Importance of Features 1 & 2')
```



[160]:
```
#9
def newer(data):
    fn = np.sqrt((data['Feature 1']**2 + (data['Feature 2'])**2))
    #df = data['fn']
    return fn

c_training_set = newer(X_train)
c_testing_set = newer(X_test)

c = pd.DataFrame(c_testing_set)
c.columns = ['New Feature']
c['Label'] = y_test
c
d= pd.DataFrame(c_training_set)
d.columns = ['New Feature']
d['Label'] = y_train
d
```

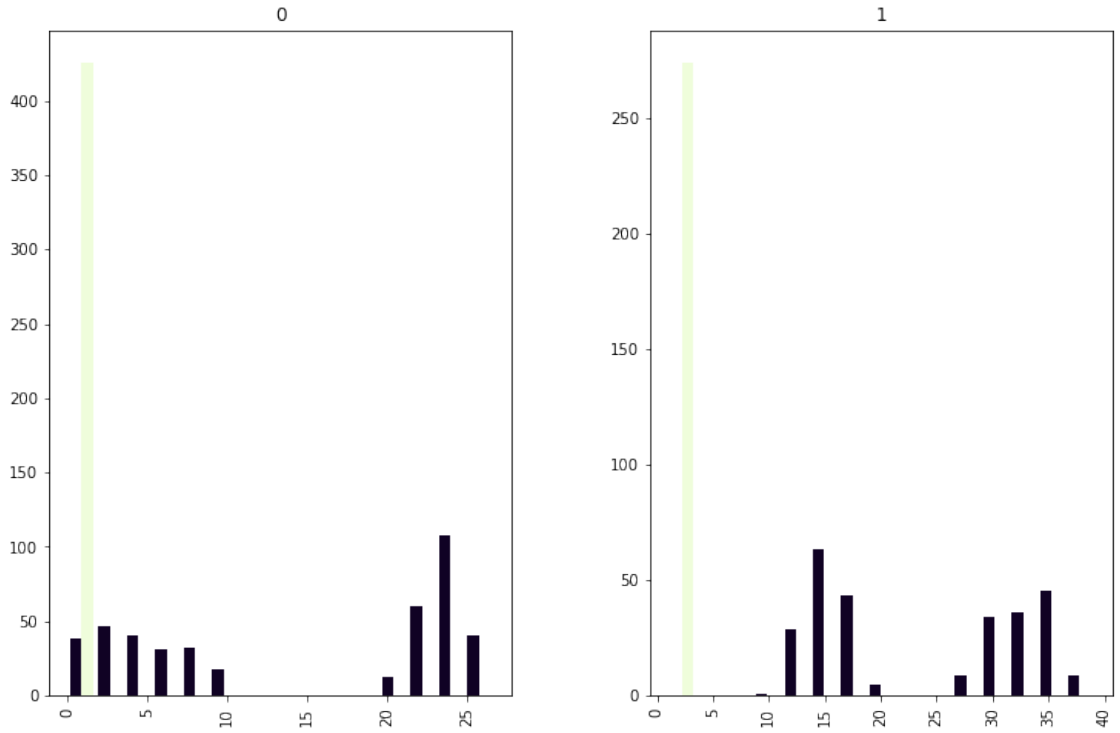
[160]:

	New Feature	Label
810	2.525564	0
2366	25.815432	0
2922	31.578777	1

```
455     23.633357       0
4130    36.098584       1
…             …     …
7643    16.906501       1
1327     0.567506       0
7344    18.525187       1
812      5.109203       0
337     38.025893       1

[700 rows x 2 columns]
```

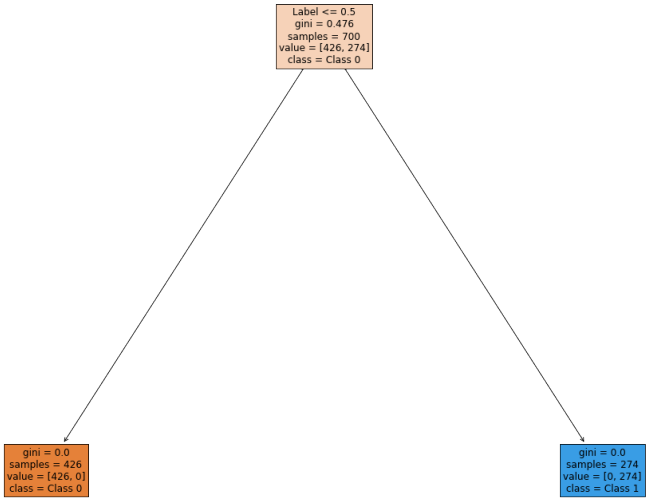
[162]:
```
#10
cf0 = c[['New Feature', 'Label']]
cf1 = d[['New Feature', 'Label']]
#cf0.hist(by='Label',color = [color1,color2],figsize=[12, 8], bins=15)
cf1.hist(by='Label',color = [color1,color2], figsize=[12, 8], bins=15)
```

[162]:
```
array([<AxesSubplot:title={'center':'0'}>,
       <AxesSubplot:title={'center':'1'}>], dtype=object)
```



[167]:
```
dlf = DecisionTreeClassifier(max_depth=3)
dlf = dlf.fit(d,y_train)
```

[168]:
```
feature_names=d.columns
fig = plt.figure(figsize = (25,20))
_ = tree.plot_tree(dlf,
                   feature_names = feature_names,
                   class_names = {0:'Class 0', 1:'Class 1'},
                   filled = True,
                   fontsize = 12)
```



[201]:
```
predictions = dlf.predict(d)
```