Course: COMP 671D, Machine Learning

Homework Assignment 2

Problem 3

Name: Samuel Eure

In [11]:

```
import numpy as np
import sklearn as sk
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification
import seaborn as sns
import pandas as pd
import random
random.seed(10)
def showMessage(dropped):
    print("----------------------------------------------------------------")
    print("Feature importances having dropped "+dropped)
    print("----------------------------------------------------------------")
    print("Feature \t : Importance" )
```

Problem 3a)

In [3]:

```
df = pd.read_csv("ProPublica_COMPAS_preprocessed.csv")
data = np.array(df.iloc[:,1:])
features = df.columns.tolist()[1:-1]
```

Splitting the data into $4/5$ training data and $1/5$ testing data.

In [4]:

```
x_train,x_test, y_train, y_test = train_test_split(data[:,:-1],
                                                    data[:,-1],
                                                    test_size=0.2,
                                                    random_state=42)
```

Creating and fitting the Random Forest

In [5]:

```
clf = RandomForestClassifier()
clf.fit(x_train, y_train)
```

Out[5]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='g
ini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

Evaluation. Below, my test accuracy and F1 score should be printed out.

In [6]:

```
def getValues(clf, x_test, y_test):
    predictions = clf.predict(x_test)

    tp = sum(predictions*y_test)
    tn = sum((predictions-1)*(y_test-1))
    n  = len(y_test)
    fp = abs(sum(predictions*(y_test-1)))
    fn = abs(sum((predictions-1)*y_test))

    accuracy  = (tp+tn)/n
    precision = tp/(tp+fn)
    recall    = tp/(tn+fp)
    f1        = 2*(precision*recall/(precision+recall))

    print("Test Accuracy :", accuracy)
    print("     F1 Score :", f1)
getValues(clf, x_test, y_test)
```

```
Test Accuracy : 0.6431064572425829
     F1 Score : 0.2792321116928446
```

Problem 3b)

Below, I will print out the relative importances of each of variables:

In [7]:

```
print("Feature : Importance" )
importance = clf.feature_importances_
for i in range(len(features)):
    print(features[i]+": \t "+str(np.round(importance[i],4)))
```

```
Feature : Importance
p_current_age:   0.2754
p_age_first_offense:     0.252
p_charge:        0.2842
p_jail30:        0.0122
p_prison:        0.0369
p_probation:     0.0772
race_black:      0.0257
race_white:      0.0194
race_hispanic:   0.0123
race_asian:      0.0025
race_native:     0.0023
```

From what is shown above, it appears that given the age, first offense, and charge of a person, their race makes little difference. That is, the relative importance of these three variables (which convey the information of age and part of the criminal history) outways that of race by roughly a factor of 10.

Problem 3c)

Removing the top two.

I will first remove charge and current_age from the data set since they have the highest feature importances. Then, I will retrain the model.

In [8]:

```
twoDf = df.drop(['p_charge', 'p_current_age'], axis = 1)
dataTrim = np.array(twoDf.iloc[:,1:])
features = twoDf.columns.tolist()[1:-1]
x_train2,x_test2, y_train2, y_test2 = train_test_split(dataTrim[:,:-1],
                                                       dataTrim[:,-1],
                                                       test_size=0.2,
                                                       random_state=42)
clf = RandomForestClassifier()
clf.fit(x_train2, y_train2)
print("From This Model:")
getValues(clf, x_test2, y_test2)
showMessage("CHARGE and CURRENT_AGE")
importance = clf.feature_importances_
for i in range(len(features)):
    print(features[i]+": \t "+str(np.round(importance[i],4)))
```

```
From This Model:
Test Accuracy : 0.6300174520069808
     F1 Score : 0.23385689354275743
---------------------------------------------------------------
Feature importances having dropped CHARGE and CURRENT_AGE
---------------------------------------------------------------
Feature           : Importance
p_age_first_offense:     0.6052
p_jail30:         0.0328
p_prison:         0.1001
p_probation:      0.1745
race_black:       0.048
race_white:       0.0182
race_hispanic:    0.013
race_asian:       0.0051
race_native:      0.0033
```

It appears that the importance of race may have increased compared to what it was prior to the

removal of charge and current_age, however all the contributions of race are still small compared to factors such as prison, probation and age_first_offense. Thus, this model doesn't place race as a primary factor into the classification of recidivism either.

Removing only one feature. First, I'll train the model by only removing charge, and then I'll train a model by only removing current_age.

In [9]:

```
oneDf1 = df.drop(['p_charge'], axis = 1)
dataTrim = np.array(oneDf1.iloc[:,1:])
features = oneDf1.columns.tolist()[1:-1]
x_train11,x_test11, y_train11, y_test11 = train_test_split(dataTrim[:,:-1],
                                                  dataTrim[:,-1],
                                                  test_size=0.2,
                                                  random_state=42)
clf = RandomForestClassifier()
clf.fit(x_train11, y_train11)
getValues(clf, x_test11, y_test11)
showMessage("CHARGE")
importance = clf.feature_importances_
for i in range(len(features)):
    print(features[i]+": \t "+str(np.round(importance[i],4)))
```

```
Test Accuracy : 0.6413612565445026
     F1 Score : 0.3019197207678883
---------------------------------------------------------------
Feature importances having dropped CHARGE
---------------------------------------------------------------
Feature           : Importance
p_current_age:    0.3852
p_age_first_offense:     0.3689
p_jail30:         0.0145
p_prison:         0.0569
p_probation:      0.1163
race_black:       0.0218
race_white:       0.0177
race_hispanic:    0.0123
race_asian:       0.0044
race_native:      0.002
```

In [10]:

```
oneDf2 = df.drop(['p_current_age'], axis = 1)
dataTrim = np.array(oneDf2.iloc[:,1:])
features = oneDf2.columns.tolist()[1:-1]
x_train12,x_test12, y_train12, y_test12 = train_test_split(dataTrim[:,:-1],
                                                            dataTrim[:,-1],
                                                            test_size=0.2,
                                                            random_state=42)
clf = RandomForestClassifier()
clf.fit(x_train12, y_train12)
getValues(clf, x_test12, y_test12)
showMessage("CURRENT_AGE")
importance = clf.feature_importances_
for i in range(len(features)):
    print(features[i]+": \t "+str(np.round(importance[i],4)))
```

```
Test Accuracy : 0.5881326352530541
     F1 Score : 0.2530541012216405
---------------------------------------------------------------
Feature importances having dropped CURRENT_AGE
---------------------------------------------------------------
Feature          : Importance
p_age_first_offense:     0.4398
p_charge:        0.3516
p_jail30:        0.0138
p_prison:        0.0493
p_probation:     0.0914
race_black:      0.0215
race_white:      0.0161
race_hispanic:   0.0113
race_asian:      0.0025
race_native:     0.0027
```

It appears that when removing only of of the original top two features, the importance of race does still not change and become one of the primary predictors of recidivism.

Problem 3d)

For this part of the problem, I will use Logistic Regression as my linear model.

In [39]:

```
model = LogisticRegression().fit(x_train, y_train)
coef = model.coef_[0]
print("Accuracy: ",str(model.score(x_test, y_test)))
```

```
Accuracy:  0.6675392670157068
```

In [40]:

```
print("Feature :\t coefficient" )
for i in range(len(features)):
    print(features[i]+": \t "+str(np.round(coef[i],4)))
```

```
Feature :        coefficient
p_age_first_offense:     -0.0217
p_charge:        -0.022
p_jail30:        0.0491
p_prison:        -0.2327
p_probation:     -0.0237
race_black:      0.0517
race_white:      0.2042
race_hispanic:   0.0454
race_asian:      -0.1538
race_native:     -0.8843
```

Above, I print the coefficients for each of the features used in the logistic regression. Since the scale of race_white, race_native and race_asian would suggest these are some of the strongest predictors of recidivism, these findings are not consistent with the results of the Random Forest Classifier. For example, predictor most heavily weighted is race_native. I'll also note that the accuracy of this model is slightly better than that of Random Forest classifier (however only the default settings of the Random Forest classifier was used, while logistic regressions doesn't really have tuning parameters).

Problem 3e)

I don't believe the answer to this question is as black and white as the question itself - no pun indented. After a short literature review of this issue, it appears that ProPublica's main claim is that when they predicted the performance of the COMPAS model using logistic regression, they ended up having race as a statistically significant predictor of recidivism and that race was used in a "biased" way in order to classify criminals. Moreover, although they seem to have performed the Logistic Regression is a correct way, the model they are evaluating is still only around 60% accurate on average anyway. As for my results after running the logistic regression, I obtained similar qualitative results (i.e. I found race to be a significant predictor of recidivism for the logistic regression), however when I fitted the data to a Random Forest model, I obtained similar accuracies and this model didn't determine race to be a main predictor in recidivism rates. It seems that the two methods may be valid and similarly accurate ways of predicting recidivism, however they do so by evaluating the factors in different ways. Furthermore, the accuracy of predictions of recidivism rates for white and black criminals is roughly

the same, however the error attributed to black criminals typically arrose by overpredicting the risk of recidivism while the opposite was true for whites.

I would argue that thet specific statistical test they implimented used race as a statistically significant predictor in classification, however had they used a differ classification model, they may have come to a different conclusion. Moreover, I believe they're focusing the errors of the models as being primarily due to race (they mention that they still find race to be a primary classification factor even after conditioning on other factors) when they should be focusing on the overall validity of the model in the first place. The COMPAS model still only correct about 60 percent of the time, so the face that there are error with it's use of classification (errors that may envolve the use of race) shouldn't be surprising in the first place.

In [ ]:

```

```