Chase Hendley & Neal Sakash

CSCI 334

April 10, 2017

Refactoring Code

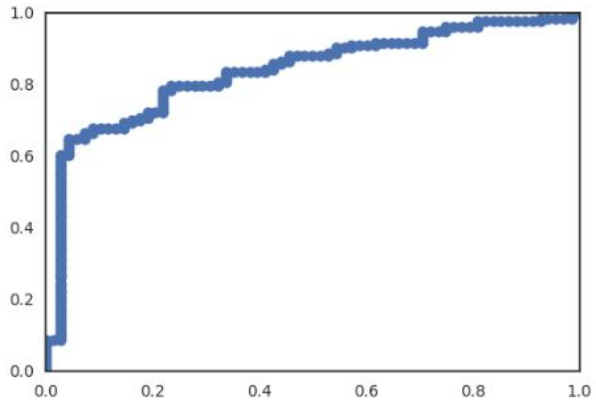
During previous assignments, there were two problems that we faced that we identified and wanted to refactor. The first was with our ROC curves. Due to various errors, we were not able to get a functioning ROC curve to be generated. The original code we used was a variation of the following:

Original plot code:

```
# Compute ROC curve and ROC area for each class
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])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
```

However, after a quick class tutorial, were were able to get an updated ROC code to put in using the classroom code:



The changes made allowed us on our Titanic code to finally create a working ROC curve model. This model uses the 50% cutoff for survived passengers. The problem, we believe with our first few attempts seems to lie with not having properly defined the fpr and tpr for the algorithm used to plot the ROC curve. By changing the way we approached to one similar to the classroom example, we were able get it working.

Our second problem was with our predictive accuracy with the Titanic dataset. When submitted to Kaggle we had a score of 0.74641. We wanted to take a further look at some of the parameters we used to see if we could increase that number.

```
trainData = pd.DataFrame.as_matrix(train[['Pclass', 'Sex', 'Fare', 'Age','Parch']])
trainTarget = pd.DataFrame.as_matrix(train[['Survived']]).ravel()
testData = pd.DataFrame.as_matrix(test[['Pclass', 'Sex', 'Fare','Age','Parch']])

classifier = GaussianNB()
classifier.fit(trainData, trainTarget)
```

These were some of our original code here, however we changed it to

```
37  # Format the data and expected values for SKLearn
38  trainData = pd.DataFrame.as_matrix(train[['Pclass', 'Sex', 'Age']])
39  trainTarget = pd.DataFrame.as_matrix(train[['Survived']]).ravel()
40  testData = pd.DataFrame.as_matrix(test[['Pclass', 'Sex', 'Age']])
41
42  classifier = GaussianNB()
43  classifier.fit(trainData, trainTarget)
44
```

By limiting the items that we predicted, we were able to increase our numbers slightly:

We could continue to refactor and work on future iterations to produce one that resulted in a higher probability score. Possibly substituting “Pclass” with “Fare” and dividing the fare amount into classes other than those found in PClass.