Assignment 3

BIOM 5405 - Winter 2017

Name: Maryam Kaka 100929992

Date Submitted: February 23, 2017

All code used to produced plots and obtain values for this assignment was written in python. Five python libraries (`math`, `pandas`, `numpy`, `sklearn` and `matlibplot`) were also included to aid with different aspects of data manipulation and visualization. All the code used can be found in Appendix A. Code output can be found in Appendix B

1.0 - Classifier Scores

(i) (a) Tables I show the confusion matrices for a classifier assuming 100 samples in each class

Table I: Confusion matrix of classifier with 100 samples in each class

		Predicted	
		T	**F**
Actual	**T**	65	35
	F	38	62

(b) Tables II show the confusion matrices for a classifier assuming 100 positive samples and 1000 negative classes.

Table II: Confusion matrix of classifier with 100 positive samples and 1000 negative

		Predicted	
		T	**F**
Actual	**T**	65	35
	F	445	655

(c) Tables III show the confusion matrices for a classifier assuming 400 samples in each class

Table III: Confusion matrix of classifier with 400 samples in each class

		Predicted	
		T	**F**
Actual	**T**	260	140
	F	152	248

Table IV summarizes the result of a chi squared test performed for each case. It can be noted that for all there cases the p-value obtained was much less than 0.05 and therefore each of the classifiers performed better than random. This can also be seen in the precision-recall plot, Figure I, as each of the points are above the respective random classifier line

Table IV: Summary of chi-squared test performed for each case

Case	χ^2	**P-Value**
A	14.59	1.33×10^{-4}
B	27.51	1.56×10^{-7}
C	58.37	2.16×10^{-14}

(ii) Table V summarized the precision or positive predictive value (PPV) calculated for each of the cases. Equations 1 was used to calculate the values for each case.

$$PPV = \frac{TP}{TP + FP} \tag{1}$$

Table V: Summary of PPV calculated for each case

Case	**PPV**
A	0.631
B	0.146
C	0.631

(iii) (iv) Figure I shows the precision recall curve plotted for each of the cases as well as the curve for a random classifier for each of the cases. Note that a and c on the plot are overlapping.

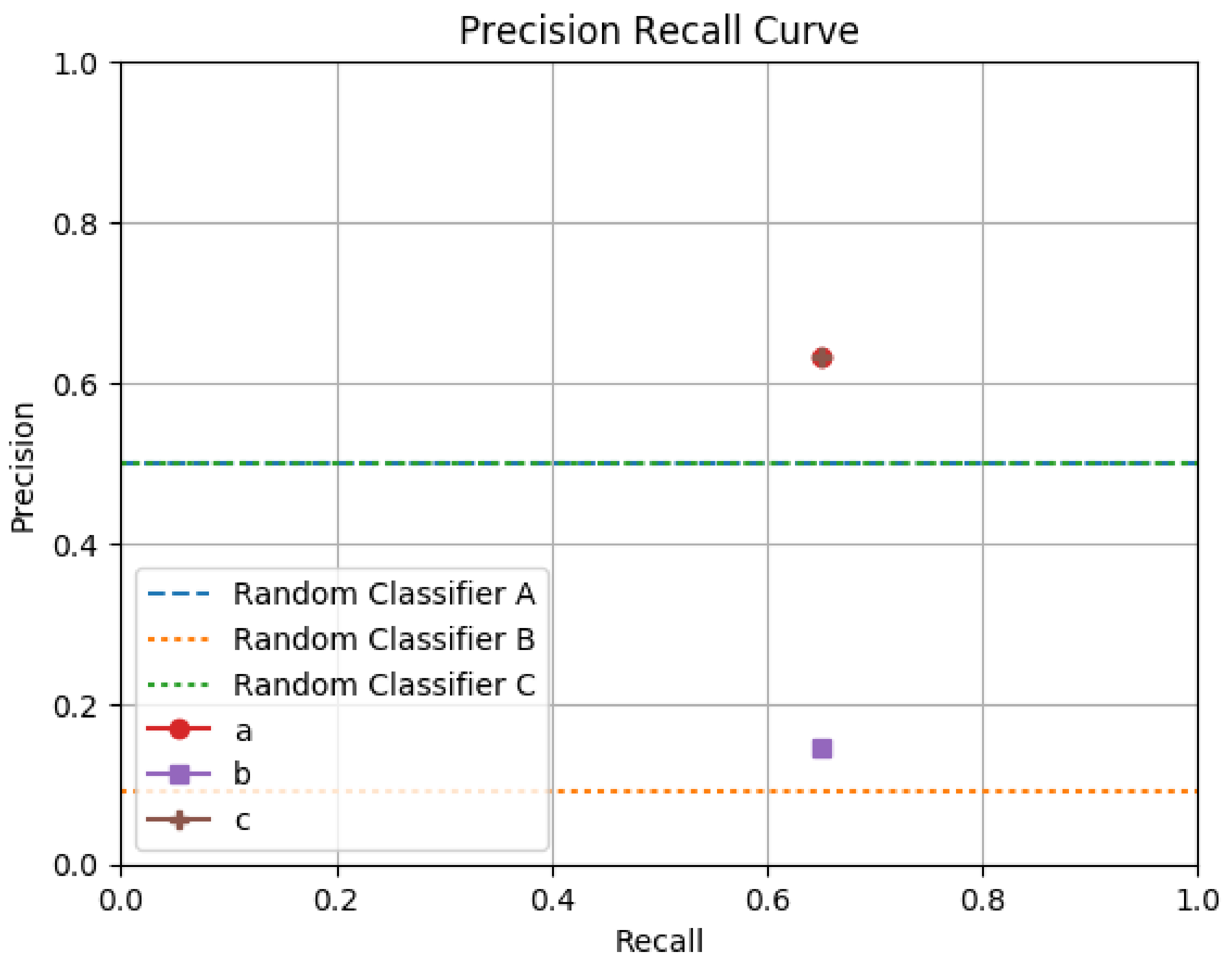

Figure I: Precision Recall curve of each of random classifier performance as well each classifier for each case

2.0

(i) Figure II shows the receiver operating characteristic curve for the given dataset. The area under the curve was found to be 0.709494.

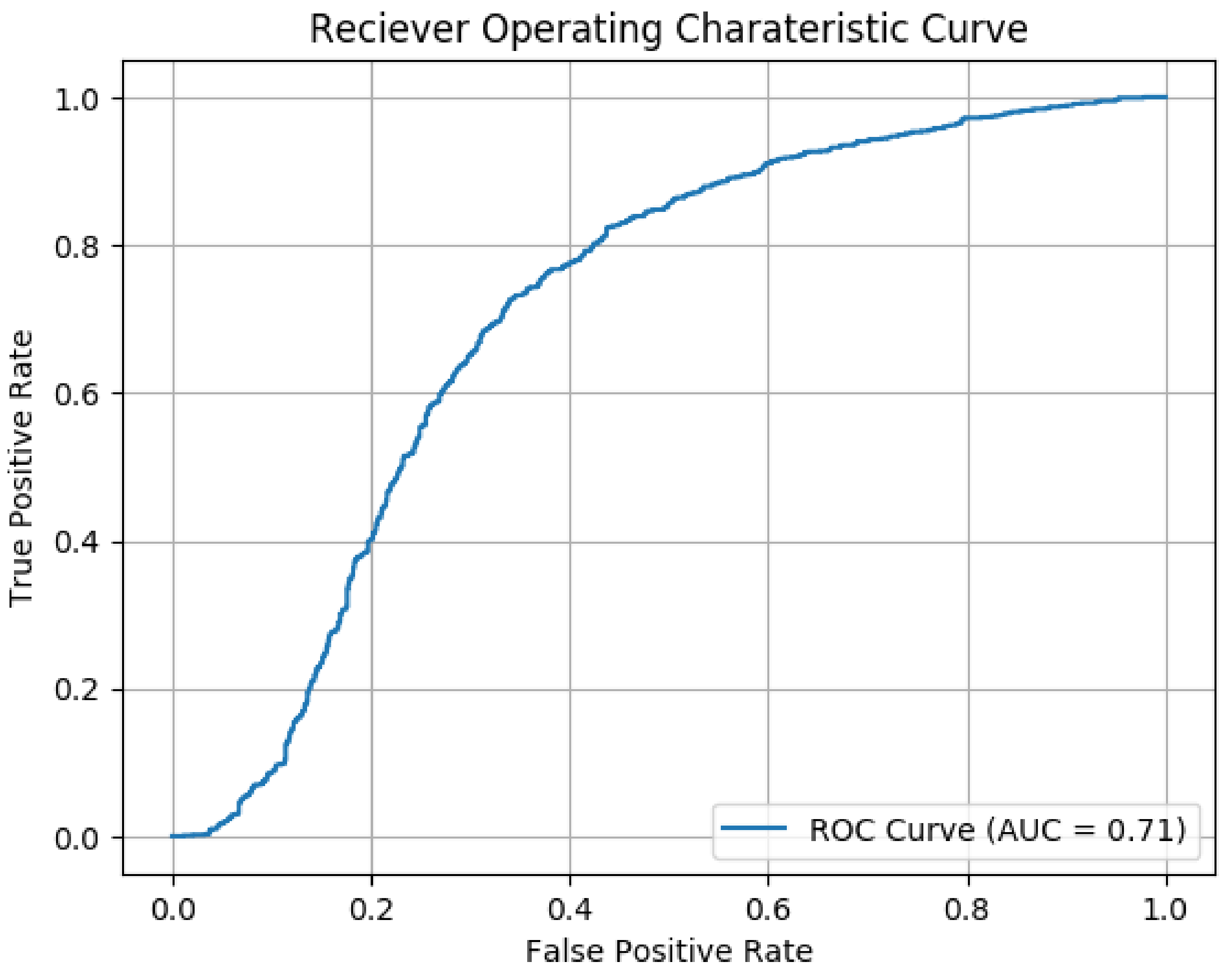

Figure II: Plot of receiver operating characteristic curve for the collected data

(ii) Looking at Figure II it can be seen that in order to obtain a sensitivity (i.e. TPR) of at leas 75% the maximum specificity (i.e. 1-FPR) that can be obtained is 63%

(iii) (iv) Figure III shows the precision recall curve obtained. It can be seen on the plot that in order to obtain a sensitivity (i.e. recall) of 75% then a precision of 66.96 % is required.

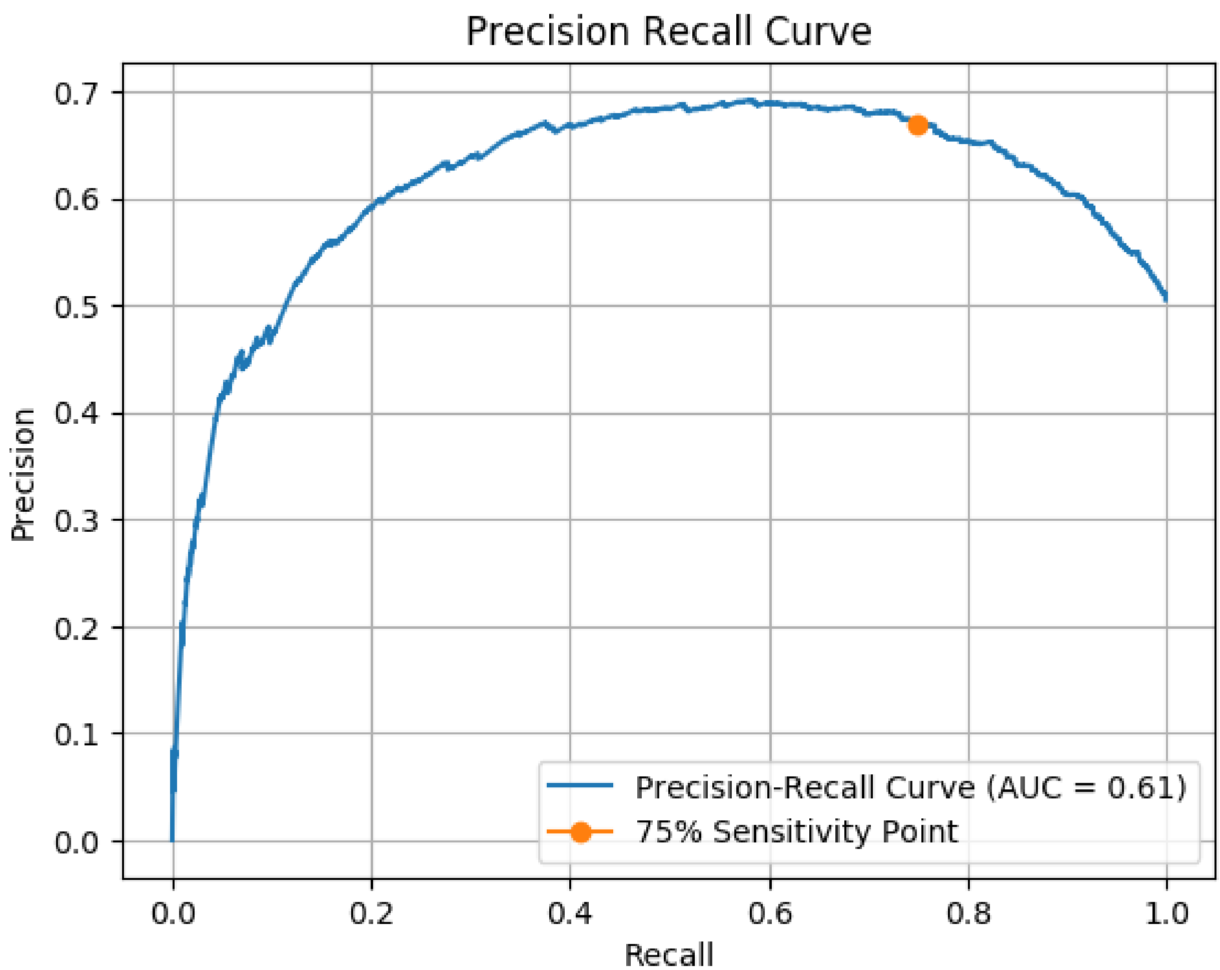

Figure III: Plot of precision recall curve with 75% sensitivity marked

(v) Using a bootstrap with 2000 bootstrapped samples resulted in a 95% confidence interval of [0.666, 0.674].

(vi) A class imbalance of 1000:1 would result in a precision of 0.1% for a recall of 75%. This value was calculated by generating a new dataset (through random sampling), based on the original, which reflects the class imbalance.

(vii) The fall discovery rate (FDR) is a measure of the number of false positive (i.e. Type I error) over the number of positive predictions

Appendix A - Python Code

```
import pandas as pd
import numpy as np
import scipy.stats as stats
import math
import matplotlib.pyplot as plt
import bisect
from sklearn import metrics
from sklearn import utils

outputLocation = 'Assignment #3/images/'

def chiSquared(data):
    df = ((data.shape[0] - 1) - 1) * ((data.shape[1] - 1) - 1)
    contingencyTable = np.outer(data.RowTotal[0:-1], data.ix["ColTotal"][0:-1])\
        /data.ix["ColTotal","RowTotal"]
    contingencyTable = pd.DataFrame(contingencyTable, index=rows, columns = col)

    x, p = stats.chisquare(data.ix[0:-1, 0:-1], contingencyTable, ddof = df)
    x = x.sum()
    p = 1 - stats.chi2.cdf(x, df)

    print('Degrees of Freedom: ' + str(df))
    print("Chi^2: " + str(x))
    print("p-value: " + str(p))

def confusionMatrix(tp, fn, fp, tn):
    data = pd.DataFrame([[tp, fn], [fp, tn]], index=rows, columns=col)
    data = data.append(pd.Series(data.sum(), name="ColTotal"))
```

```
    data = data.assign(RowTotal=pd.Series(data.sum(axis=1)))
    print(data)
    return data

def bootstrap(data, nSamples):
    stats = np.empty(nSamples)
    for i in range(0, nSamples):
        sample = utils.resample(data)
        stats[i] = findPrecision(sample['class'], sample['score'])
    np.sort(stats)
    return [stats[math.floor(nSamples*0.025)], stats[math.ceil(nSamples*0.975)]]

def findPrecision(c, s):
    precision, recall, t = metrics.precision_recall_curve(c, s, pos_label=1)
    points = np.array([[i, x] for i,x in enumerate(recall) if x >= 0.75])
    points = points[np.argsort(points[:, 1])]
    return precision[points[0][0]]

print("QUESTION 1")
col = ["TestT", "TestF"]
rows = ["ActualT", "ActualF"]
SENS = 0.65
SPEC = 0.62

ppv = lambda tp, fp: tp/(tp+fp)

print('(i)')
print('a.')
nPos = 100; nNeg = 100
tp = SENS*nPos; fn = nPos - tp; tn = SPEC*nNeg; fp = nNeg - tn
```

```
PPVa = ppv(tp, fp)
chiSquared(confusionMatrix(tp, fn, fp, tn))

print('\nb.')
nPos = 100; nNeg = 1000
tp = SENS*nPos; fn = nPos - tp; tn = SPEC*nNeg; fp = nNeg - tn
PPVb = ppv(tp, fp)
chiSquared(confusionMatrix(tp, fn, fp, tn))

print('\nc.')
nPos = 400; nNeg = 400
tp = SENS*nPos; fn = nPos - tp; tn = SPEC*nNeg; fp = nNeg - tn
PPVc = ppv(tp, fp)
chiSquared(confusionMatrix(tp, fn, fp, tn))

print('\n(ii)')
print('a. PPV: ' + str(PPVa))
print('b. PPV: ' + str(PPVb))
print('c. PPV: ' + str(PPVc))

plt.figure()
plt.plot([0, 1],[0.5, 0.5], linestyle='dashed', label='Random Classifier A')
plt.plot([0, 1],[100/1100, 100/1100], linestyle='dotted', label='Random Classifier B')
plt.plot([0, 1],[0.5, 0.5], linestyle='dotted', label='Random Classifier C')
plt.plot(SENS, PPVa, marker='o', label='a')
plt.plot(SENS, PPVb, marker='s', label='b')
plt.plot(SENS, PPVc, marker='P', label='c')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.title('Precision Recall Curve')
```

```
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = 'lower left')
plt.grid(True)
plt.savefig(outputLocation + 'precision-recall.png', bbox_inches='tight')

print('\nQUESTION 2')
data = pd.read_csv("Assignment #3/assigData3.tsv", sep='\t', index_col=False,
    header=None, names=["score", "class"])

print('(i)')
fpr, tpr, thresh = metrics.roc_curve(data['class'], data['score'], pos_label=1)
auc = metrics.auc(fpr, tpr)
print('AUC: ' + str(auc))

plt.figure()
plt.plot(fpr, tpr, label='ROC Curve (AUC = %0.2f)' %auc)
plt.title('Reciever Operating Charateristic Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc = 'lower right')
plt.grid(True)
plt.savefig(outputLocation + 'ROC.png', bbox_inches='tight')

print('(ii)')
minSens = 0.75
index = bisect.bisect_left(tpr, minSens)
print('Max Specificity: ' + str(1-fpr[index]))

print('(iii)')
```

```
precision, recall, t = metrics.precision_recall_curve(data['class'],
    data['score'], pos_label=1)
auc = metrics.auc(recall[1:-1], precision[1:-1])

print('(iv)')
index = [i for i,x in enumerate(recall) if x == 0.75][0]
print('Precision (@75'+ '%' +' Sensitivity): ' + str(findPrecision(data['class'],
    data['score'])))

plt.figure()
plt.plot(recall[1:-1], precision[1:-1],
    label='Precision-Recall Curve (AUC = %0.2f)' %auc)
plt.plot(recall[index], precision[index], marker='o', label='75% Sensitivity Point')
plt.title('Precision Recall Curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = 'lower right')
plt.grid(True)
plt.savefig(outputLocation + 'precision-recall2.png', bbox_inches='tight')

print('(v)')
print(bootstrap(data, 2000))

print('(vi)')
modifiedSet = utils.resample(data[data['class'] == 0], n_samples=1000)
modifiedSet = modifiedSet.append(data[data['class'] == 1][:1])
print('Expected Precision: ' + str(findPrecision(modifiedSet['class'],
    modifiedSet['score'])))
```

Appendix B - Code Output

```
QUESTION 1
(i)
a.
          TestT  TestF  RowTotal
ActualT    65.0   35.0     100.0
ActualF    38.0   62.0     100.0
ColTotal  103.0   97.0     200.0
Degrees of Freedom: 1
Chi^2: 14.5931338204
p-value: 0.000133399716113

b.
          TestT  TestF  RowTotal


ActualT    65.0   35.0     100.0
ActualF   380.0  620.0    1000.0
ColTotal  445.0  655.0    1100.0
Degrees of Freedom: 1
Chi^2: 27.5117934643
p-value: 1.5613949178e-07

c.
          TestT  TestF  RowTotal
ActualT   260.0  140.0     400.0
ActualF   152.0  248.0     400.0
ColTotal  412.0  388.0     800.0
Degrees of Freedom: 1
Chi^2: 58.3725352818
```

p-value: 2.16493489802e-14

(ii)

a. PPV: 0.6310679611650486

b. PPV: 0.14606741573033707

c. PPV: 0.6310679611650486

QUESTION2

(i)

AUC:0.709494

(ii)

MaxSpecificity:0.63

(iii)

(iv)

Precision(@75%Sensitivity):0.669642857143

(v)

[0.67393278837420523,0.66607773851590102]

(vi)

Expected Precision: 0.00102249488753