Problem 4

(I did this whole problem using the one source code)

A)

Output from my program

             precision    recall  f1-score   support

          ham       0.94      1.00      0.97       964

         spam       1.00      0.62      0.77       151

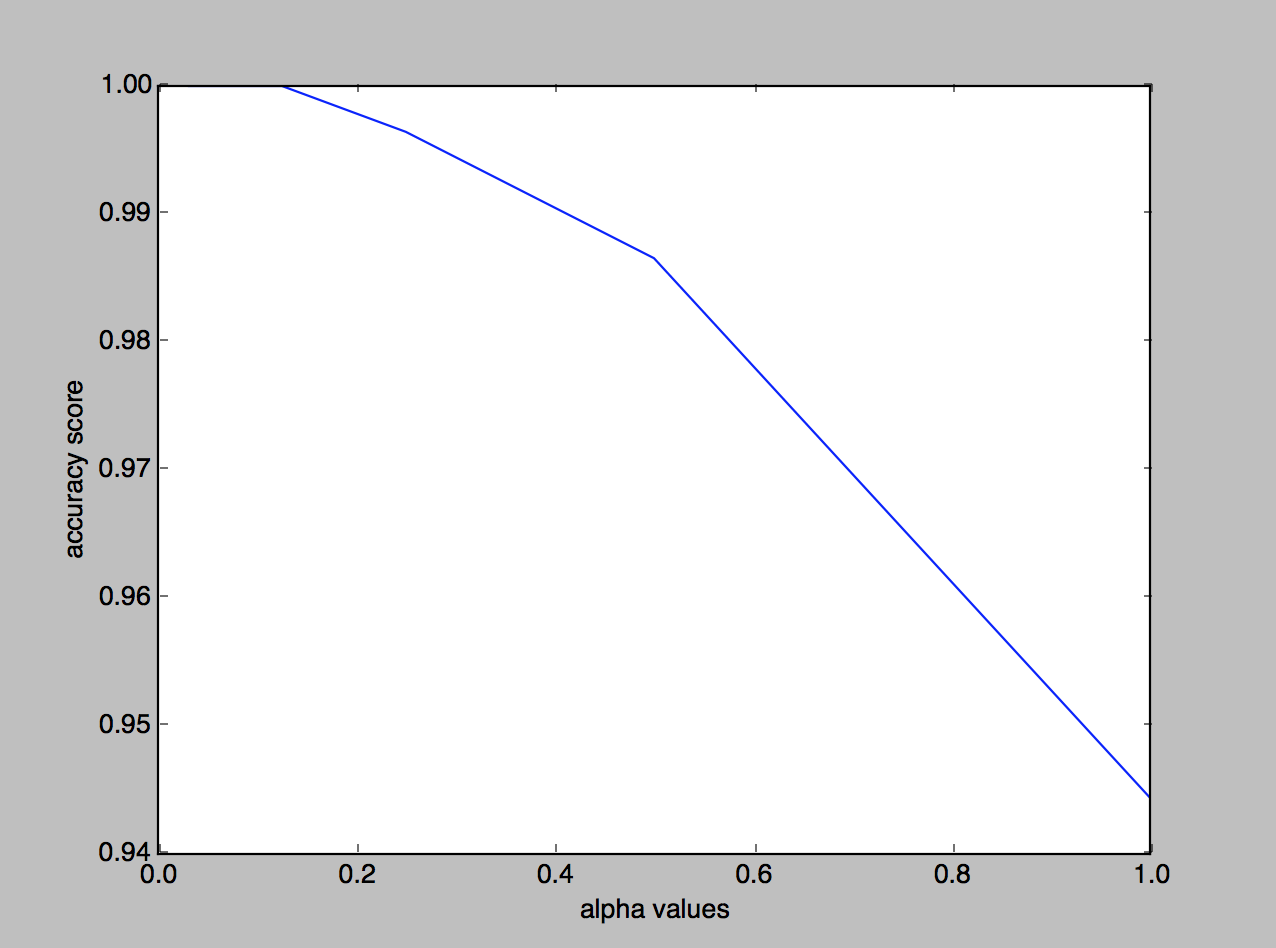
avg / total       0.95      0.95      0.94      1115

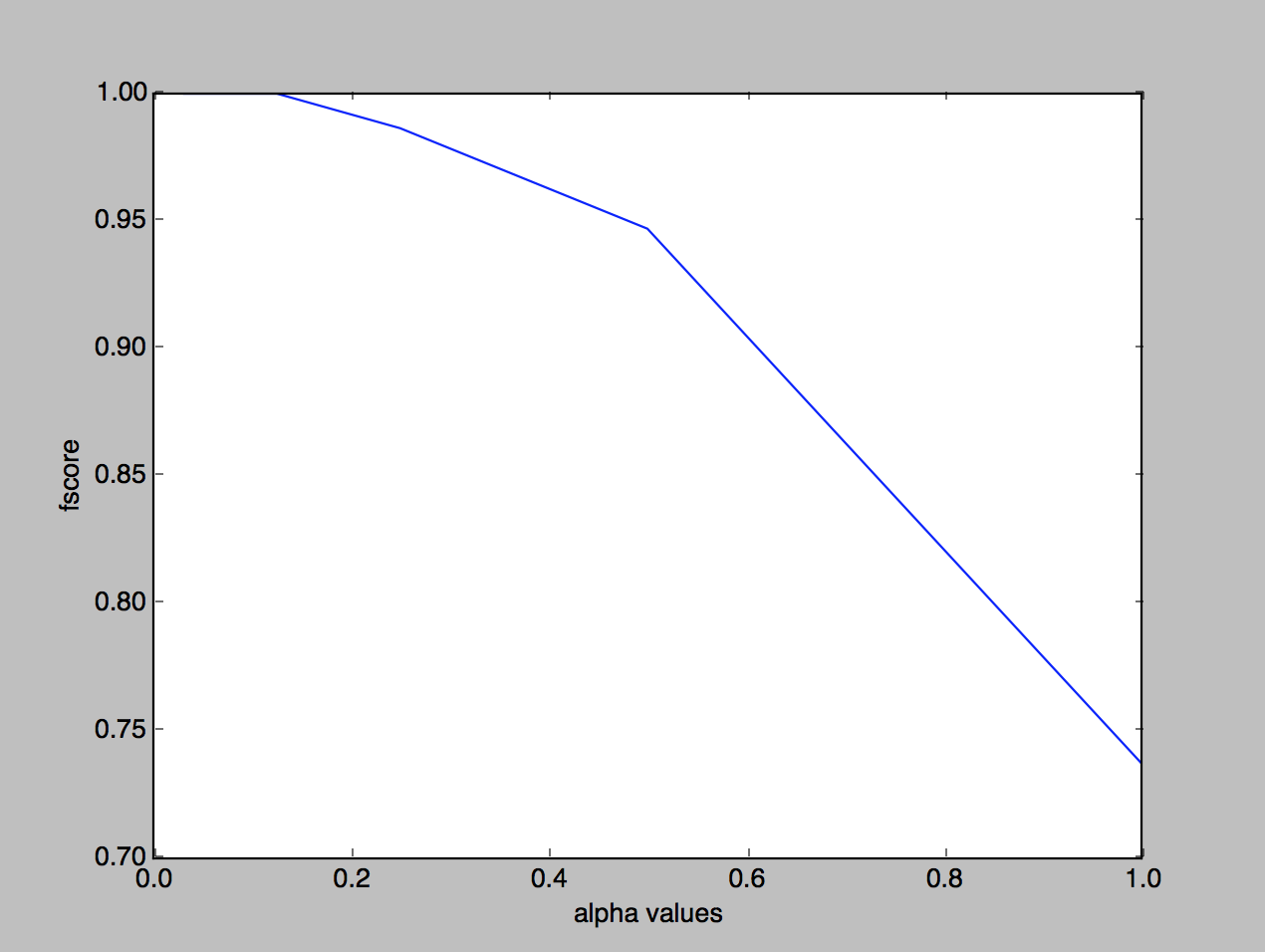
Confusion Matrix

[964   0]

 [ 57  94]

B) Both F-score and accuracy score seem to have an inverse correlation with alpha. I.E. as alpha increases in value, both tend to decrease





Source Code

(python)

"""

    Octave and julia doesn't work on my mac for some reason

    so I did the whole program in python. I used scikit learn

    for most all of the bayes and score calculations

"""

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn import metrics

import matplotlib.pyplot as plt

with open('SMSSpamCollection', 'r') as f:

    dataFull = f.read()

    lines = dataFull.split("\n")

    lines = list(map(lambda line: line.split("\t"), lines))

    del lines[-1]

    targets = map(lambda line: line[0], lines)

    targets = map(lambda target: 1 if target == "spam" else 0, targets)

    data = map(lambda line: line[1], lines)

    df = pd.DataFrame(

    {'target': list(targets),

     'data': list(data)

     })

    df['data'] = df['data'].astype(str)

    train,test = train\_test\_split(df, test\_size = 0.2)

    count\_vect = CountVectorizer()

    X\_train\_counts = count\_vect.fit\_transform(test.data)

    tfidf\_transformer = TfidfTransformer()

    X\_train\_tfidf = tfidf\_transformer.fit\_transform(X\_train\_counts)

    f1Score=[None]

    accuracyScore=[None]

    clf = MultinomialNB().fit(X\_train\_tfidf, test.target)

    docs\_test = test.data

    predicted = clf.predict(X\_train\_tfidf)

    print(metrics.classification\_report(test.target, predicted))

    print('Confusion Matrix ')

    print(metrics.confusion\_matrix(test.target, predicted))

    i1=0.03125

    i2= 0.0625

    i3=0.125

    i4=0.25

    i5=.5

    i6=1

    x = [i1,i2,i3,i4,i5,i6]

    for i in x:

        clf = MultinomialNB(alpha=i).fit(X\_train\_tfidf, test.target)

        docs\_test = test.data

        predicted = clf.predict(X\_train\_tfidf)

        fscore = metrics.f1\_score(test.target, predicted)

        accuracy = metrics.accuracy\_score(test.target, predicted)

        f1Score.append(fscore)

        accuracyScore.append(accuracy)

    i0=None

    x = [i0,i1,i2,i3,i4,i5,i6]

    plt.plot(x,f1Score)

    plt.ylabel('fscore')

    plt.xlabel('alpha values')

    plt.show()

    plt.plot(x,accuracyScore)

    plt.ylabel('accuracy score')

    plt.xlabel('alpha values')

    plt.show()