

Sydney Trout
Dr. Li
CPSC 4430
13 April 2022

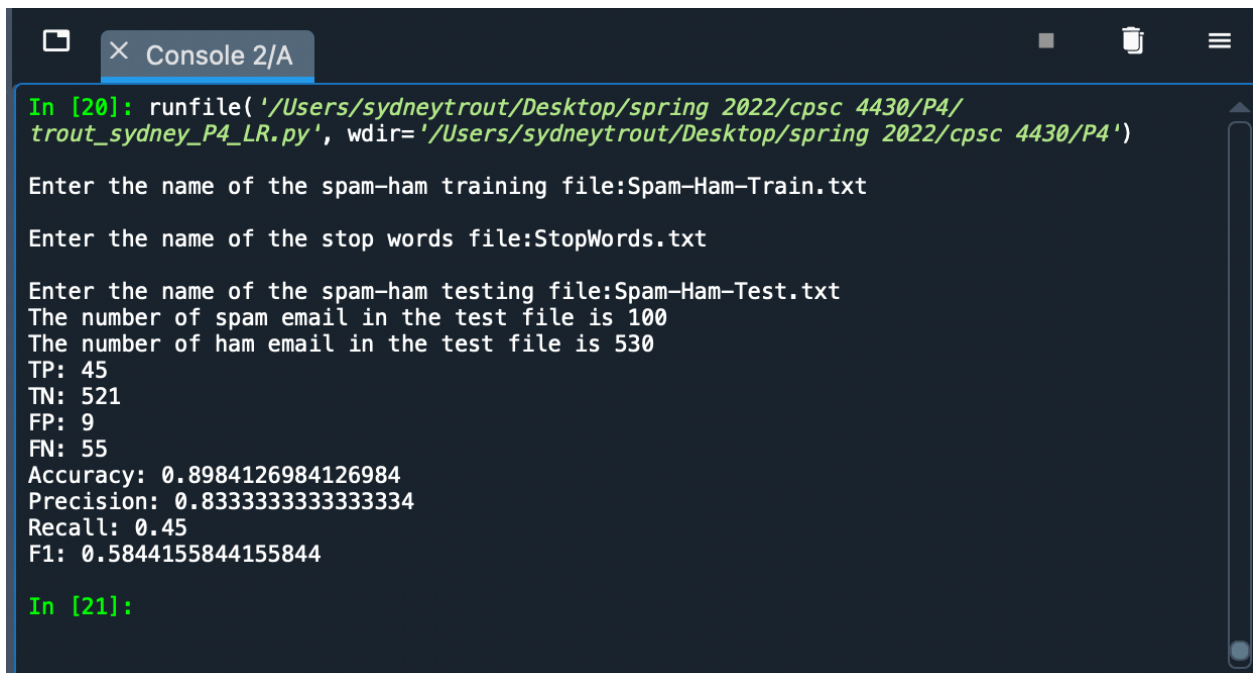
Project 4 Report

The model I used for this project was the Naïve Bayes Classifier. This classifier predicts on the basis of probability and within this project allowed for decisions to be made on whether an email was spam or ham based upon its subject line.

TP, TN, FP, FN, Accuracy, Precision, Recall, and F1 Score:

```
The number of spam email in the test file is 100
The number of ham email in the test file is 530
TP: 45
TN: 521
FP: 9
FN: 55
Accuracy: 0.8984126984126984
Precision: 0.8333333333333334
Recall: 0.45
F1: 0.5844155844155844
```

Python Console:



```
Console 2/A

In [20]: runfile('/Users/sydneytrout/Desktop/spring 2022/cpsc 4430/P4/
trout_sydney_P4_LR.py', wdir='/Users/sydneytrout/Desktop/spring 2022/cpsc 4430/P4')

Enter the name of the spam-ham training file:Spam-Ham-Train.txt

Enter the name of the stop words file:StopWords.txt

Enter the name of the spam-ham testing file:Spam-Ham-Test.txt
The number of spam email in the test file is 100
The number of ham email in the test file is 530
TP: 45
TN: 521
FP: 9
FN: 55
Accuracy: 0.8984126984126984
Precision: 0.8333333333333334
Recall: 0.45
F1: 0.5844155844155844

In [21]:
```

Code Copy:

```
import numpy as np
```

```
#puts every letter in lower case and removes punctuation and strips out white space
```

```
def cleantext(text):  
    text = text.lower()  
    text = text.strip()  
    for letters in text:  
        if letters in ""[!.,"-!—@;':#$%^&*()+/?""":  
            text = text.replace(letters, " ")  
    return text
```

```
#removes duplicate words
```

```
def countwords(words, is_spam, counted):  
    for each_word in words:  
        if each_word in counted:  
            if is_spam == 1:  
                counted[each_word][1]=counted[each_word][1] + 1  
            else:  
                counted[each_word][0]=counted[each_word][0] + 1  
        else:  
            if is_spam == 1:  
                counted[each_word] = [0,1]  
            else:  
                counted[each_word] = [1,0]  
    return counted
```

```
#records each word as it appears in spam or ham
```

```
def make_percent_list(k, theCount, spams, hams):  
    for each_key in theCount:  
        theCount[each_key][0] = (theCount[each_key][0] + k)/(2*k+hams)  
        theCount[each_key][1] = (theCount[each_key][1] + k)/(2*k+spams)  
    return theCount
```

```
#naive bayes approach function
```

```
def bayes(goodwords, vocab, probspam, probham):  
    numinspam = 0  
    numinham = 0  
    for i in vocab:  
        if(i in goodwords):
```

```

        numinspam += np.log(vocab[i][1])
        numinham += np.log(vocab[i][0])
    else:
        numinspam += np.log(1-vocab[i][1])
        numinham += np.log(1-vocab[i][0])
    numinspam = np.e**(numinspam)
    numinham = np.e**(numinham)
    spam = numinspam * probspam
    ham = numinham * probham
    nbvalue = (spam)/(spam + ham)
    return nbvalue

```

#parameter initialization

```

spam = 0
ham = 0
tspam = 0
tham = 0
testresults = []
counted = dict()

```

#loads training and stop word file

```

fname = input("Enter the name of the spam-ham training file:")
fin = open(fname,"r")
sname = input("Enter the name of the stop words file:")
sin = open(sname,"r")

```

#creates list of stop words from the file

```

stextline = sin.readline()
stopwords = []
while stextline != "":
    stextline = stextline.strip()
    stopwords.append(stextline)
    stextline = sin.readline()

```

#processes the training data

```

textline = fin.readline()
while textline != "":
    is_spam = int(textline[:1])
    if is_spam == 1:

```

```

        spam = spam + 1
    else:
        ham = ham + 1
    textline = cleantext(textline[1:])
    words = textline.split()
    words = set(words)
    goodwords = []
    for i in words:
        if(i not in stopwords):
            goodwords.append(i)
    counted = countwords(goodwords, is_spam, counted)
    textline = fin.readline()
vocab = (make_percent_list(1,counted,spam,ham))

#probability of spam and ham file calculated from training file
probspam = spam / (spam + ham)
probham = ham / (spam + ham)

#process the testing data
tname = input("Enter the name of the spam-ham testing file:")
tin = open(tname,"r")
ttextline = tin.readline()
while ttextline != "":
    is_spam = int(ttextline[:1])
    if is_spam == 1:
        tspam = tspam + 1
    else:
        tham = tham + 1
    ttextline = cleantext(ttextline[1:])
    twords = ttextline.split()
    tgoodwords = []
    twords = set(twords)
    for i in twords:
        if(i not in stopwords):
            tgoodwords.append(i)
    test = bayes(tgoodwords, vocab, probspam, probham)
    if(test >= 0.5):
        testresults.append([is_spam, 1])
    else:

```

```

        testresults.append([is_spam, 0])
    ttextline = tin.readline()

print("The number of spam email in the test file is",tspam)
print("The number of ham email in the test file is",tham)

#calculates number of TP,TN,FP,and FN based on test results
TP = 0
TN = 0
FP = 0
FN = 0
for i in range(len(testresults)):
    if(testresults[i] == [0,0]):
        TN = TN + 1
    elif(testresults[i] == [1,0]):
        FN = FN + 1
    elif(testresults[i] == [0,1]):
        FP = FP + 1
    else:
        TP = TP + 1

#calculations for accuracy, precision, recall, and f1
accuracy = (TP + TN) / (TP + TN + FP + FN)
precision = TP / (TP + FP)
recall = TP / (TP + FN)
f1 = 2 * (1 / ((1/precision) + (1/recall)))

#prints the data
print("TP:", TP)
print("TN:", TN)
print("FP:", FP)
print("FN:", FN)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1:", f1)
sin.close()
fin.close()
tin.close()

```