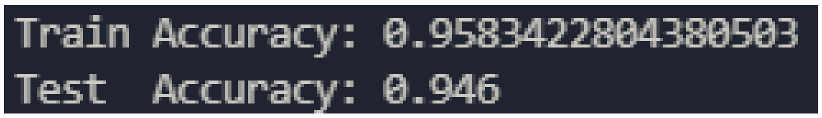
Name: Darren Lee
Contact information: danalee@ucsc.edu
CSE107

Project II Report

Instructions to compile/run my program

In the command line terminal, run the following:
python naive_bayes.py

Screenshot of expected output



```
Train Accuracy: 0.9583422804380503
Test  Accuracy: 0.946
```

Explanation of why my program works

Overview of program

- The naive_bayes program uses a set of training data to calculate ham and spam probabilities for lists of files such as emails. The NaiveBayes class is the spam filter and works by predicting if a given test file is ham or spam based on the training data.

Implementation

- My get_counts function returns the length of the lists of ham and spam emails so that I can count the total number of each type of email.
- The fit function gets the set of words in each email and sets the dictionary value of each word to the number of times the word appears in the emails (0 if the word has not been encountered before). It does this separately for the lists of ham and spam emails, using two separate dictionaries.
- The predict function first gets the probabilities of the ham emails by dividing the number of ham emails by the total number of all training emails. Then it does the same thing for the spam emails.
    - I take the logarithm using np.log of the ham and spam probabilities to avoid underflow because conditional probabilities will get very close to 0 when multiplying.
    - I use Laplace smoothing by adding 1 to the numerator of the conditional probabilities of the words in the training/testing email and 2 to the denominator to avoid getting any conditional probabilities of 0. Conditional probabilities of 0 will make entire probabilities 0 which could cause problems like division by 0.

- Since logarithms have the property log(xy)=log(x)+log(y), my program just adds the logarithms of all the conditional probabilities for the words given ham or spam after using Laplace smoothing to the logarithms of P(ham) or P(spam).
- I do not use Laplace smoothing for P(ham) and P(spam) since those will not be 0 anyways.
- If the sum of logarithms of the spam probabilities is greater than the sum of the logarithms of the ham probabilities, then my program predicts that the email is spam.
- If there is a tie in the sums or the ham sum is greater, then my program predicts that the email is ham.