Sharice Mayer
Naive Bayes Report
5/18/20

Program Description:

- The dataset used for this program was the UCI ML dataset(SpamBase).
- Gaussian Naïve Bayes was used to classify the SpamBase dataset

First I read in the spamdata file and placed the data into a matrix. I then shuffled the rows to mix the data. Next, I read the spamdata instances row by row, and placed each instance into one of three data sets: either a spam training set, ~spam training set, or a test set(combined spam/~spam). I kept track of the total instances of spam or ~spam seen, and used mod2 on that value to make sure that every other spam got placed into spam training or test data, and every other ~spam was placed in ~spam ~spam training or test data. I separated the spam and ~spam training sets for ease of handling the data.

I computed the prior probability by checking the number of spam instances compared to total instances in the spam training set divided by the spam + ~spam training set total. It was roughly 40%(~39.5%) for P(spam), and 60%(~60.5%) for P(~spam). For the mean and the standard deviation for each feature(all 57), I transposed the training matrices, so that features were the rows, and instances were the columns, and then used built in mean and stdev python functions on each feature row for every instance in the set for both spam and ~spam classes, which gave me 4 vector rows(spam_mean, _notspam_mean, spam_stdev, notspam_stdev) so that I could use these for the Naive Bayes computation in the next step.

For the Naive Bayes Algorithm Computation, I rearranged the algorithm by taking the log of the function, allowing me to use the sum of the logs of each function, and comparing the prediction(spam) value compared to prediction(~spam) for each instance, and used the greater value of the two as the class for that instance. As I did this, I tracked what the value should have been using the label column in order to update TruePositive, TrueNegative, FalsePositive, and FalseNegative counts for the class.

Results:

The accuracy overall was ~81%, which is pretty good. The precision was relatively low(<70%), for the spam class, but really high(~97%) for the ~spam class, while the recall rate was really high(~96%) for the spam class, and comparatively low for ~spam(~70%). Unfortunately, the most error was found in the false positive rate: while most actual spam instances were correctly identified, far too many ~spam instances were classified incorrectly as spam.

Questions:

(1) Do you think the attributes here are independent, as assumed by Naïve Bayes?

(2) Does Naïve Bayes do well on this problem in spite of the independence assumption?

(3) Speculate on other reasons Naïve Bayes might do well or poorly on this problem.

Summary:

I don't think the attributes are entirely independent, because pieces of language found in text are generally not completely independent. I actually thought that Naive Bayes did surprisingly well on this problem considering the independence assumption, but I think that it probably did poorly in terms of recognizing things that weren't spam because there were so many instances with a near zero value for a particular feature. This would cause the standard deviation to grow much larger than normal, which would certainly affect the classification of positive examples as negative. Considering the independence assumption, I think it probably did decently well because it was able to use the collection of independent features for classification, which means that several features that may be present in a certain class may be enough to overcome the error caused by other features that may give values not representative of their class.

Figure 1
Test Set Accuracy, Precision, Recall and Confusion Matrix Results

```
(base) sharices-air:~/Desktop/CS445-MachineLearning/HW/prog2-nbayes-spam$ python naivebayes.py

Prior Probability of spam = 0.394176
Prior Probability of not spam = 0.605824


Confusion Matrix for a given class spam:
|_____________________________________________________________________________
|  Actual Test Instances    |      Predicted(or 'classified')
|  2300                     |  Positive            Negative
|                           |  (in class spam)     (not in class spam)
|                           |---------------------------------------------
|  Positive(in class spam)  |  TruePositive        FalseNegative
|                           |  872                 34
|                           |
|  Negative(not in class)   |  FalsePositive       TrueNegative
|                           |  405                 989
|                           |
|-----------------------------------------------------------------------------

Percent spam predicted after test = 0.5552173913043478

Total accuracy = 0.8091304347826087
(spam)precision = 0.682850430696946
(spam)recall = 0.9624724061810155
(spam)false positive rate = 0.29053084648493543


Confusion Matrix for a given class ~spam:
|_____________________________________________________________________________
|  Actual Test Instances    |      Predicted(or 'classified')
|  2300                     |  Positive             Negative
|                           |  (in class ~spam)     (not in class ~spam)
|                           |---------------------------------------------
|  Positive(in class ~spam) |  TruePositive         FalseNegative
|                           |  989                  405
|                           |
|  Negative(not in class)   |  FalsePositive        TrueNegative
|                           |  34                   872
|                           |
|-----------------------------------------------------------------------------

Percent ~spam predicted after test = 0.44478260869565217

Total accuracy = 0.8091304347826087
(~spam)precision = 0.9667644183773216
(~spam)recall = 0.7094691535150646
(~spam)false positive rate = 0.037527593818984545

(base) sharices-air:~/Desktop/CS445-MachineLearning/HW/prog2-nbayes-spam$
```