Matthew O’Brien
CS445
HW4 Naïve Bayes Classification
2/25/2016

The purpose of this project was to create a Naïve Bayes Classifier. The data set was the same as the previous assignment, which is the spam email set. The spam data was separated by their positive and negative values. These pieces where then split in half and concatenated so that there were an even number of positive and negative in both the training and testing data sets. For no real reason, I chose to use the second half of the data pieces for my training set. This actually eliminated the need for the standard deviation check to see if it was zero. I left it within my code anyway, because it wasn’t doing any harm, and if more data is ever used, I won’t need to worry about possible standard deviation errors. The only other issue of underflow errors rounding to zero came within my Gaussian function. The np.exp() function will actually round to zero with larger negative numbers. This was only happening on a few of the training instances, but was still causing a runtime error because the probability would become zero. To fix this, within my Gaussian function, I check the inner probabilities for instances of zero, and set them to a very small number before taking the log base 10 of the inner probabilities. This seemed to fix the runtime error. It’s worth mentioning that I tried a few different routes to check for zeroes. One of the options was to split the Gaussian function apart, and check the exp() function for zeroes first. If there was a zero, set that to a very small number instead. Then before the rest of the Gaussian function. I actually found that this decreased my accuracy by roughly 0.5%. There wasn’t any real reason why I went with one route or the other, except that the final decision gave me a better accuracy for the test data. My findings are the following:

Accuracy: 84.96%
Precision: 73.69%
Recall: 96.14%

Confusion Matrix:

$$\begin{array}{llcc}
 & & \text{Expected} & \\
 & & 0 & 1 \\
\text{Guessed} & 0 & [1083 & 35] \\
 & 1 & [311 & 871]
\end{array}$$

The Classifier had a very interesting skew in the guessing. While it was very good at the 0 sets, it didn’t seem to have the same with the 1 sets. There is a very large piece of the testing set, where the classifier would guess 1’s, when it was actually a 0 set. This could be due to the amount of negative (0’s) examples given to the classifier compared to the positive (1’s). And it almost never guessed a 1 when it was expected to be a 0. This would explain why the precision is the lowest of the three values. And the reason why the precision is as high as it is. Again, this could be due to the higher amount of negative examples in the different sets compared to the positive examples.

I believe the Naïve Bayes did well on this problem even with the assumption of independence of attributes. It would be interesting to see more data given to the classifier. I believe that the majority of the attributes are indeed independent. I feel like it might be slightly cheating to the look at the names of the different attributes, but those alone sort of suggest independence. There are obviously a certain few that are not independent of each other, like the three for strings of capital letters. But I believe that some of the attributes are actually independent of one another, which explains why there was a decent outcome for the testing set. I could be possible that the Naïve Bayes Classifier could do better with other attributes. Possibly trying to figure out which of the given attributes are not independent, and excluding them. I'm not sure if the reduction in data would actually end up hurting the Classifier though. From a basic understanding of Naïve Bayes Classifiers, it seems like this Classifier did a rather good job at classifying the test set from this data.