Introduction

The amount of spam texts have increased over time. With various techniques being utilized to trap the average cell phone user. From asking for sensitive information in a casual way to encourage the user to accidentally reveal too much. To convincing the user that they should definitely click on this link, which will definitely take you to this thing that needs your quick and eager attention. Anything is on the table if it means that user information is getting spread.

But how do we keep the users safe from such an attack? We tell them which texts are likely to be spam! If they are aware that the person who sent them their most recent package is being held warning is not telling the truth, then they would be less likely to click on that link! Especially if we move the possible spam from the main message inbox. This way they would be less likely to accidentally pocket click on said malicious links.

But in order for this to be effective, we need to be accurate. If we keep wrongly detecting spam, then the users would be less likely to trust us. So we need as accurate of a spam detector model as we can get. In this analysis we will be attempting to create such an accurate model. It doesn't need to be perfect, but it has to be decently close.

Project 5 Presentation
Project 5

Dataset

For our dataset we used one called SMS Spam Collection Dataset from the UC Irvine Machine Learning Repository. This was a set of 5,572 correctly identified and labeled messages. And several features that made it perfect for our purposes. It was a clean dataset that didn't need any replacement for missing or misused data entries. The only adjustments we had to use was to use CountVectorizer to convert the messages to numerical features. It had clear labels, so it was easy for us to utilize it effectively. And lastly, it was balanced. So we would ensure that our model had the best chance of a clear representation of both spam and not spam.

Analysis Technique

Our analysis technique was using the Multinomial Naive Bayer Classifier, which is very well-suited for text and word frequency analysis, to train a model to identify spam texts among none spam text. Since the texts are already labeled as such, we can be sure to get accurate training results. We utilize a training data split of 80% to a 20% test data. Then we need to show how effective our newly trained model is at accurately spotting spam messages from non-spam, and vice versa. We will do so by first determining the accuracy of identifying both by finding the accuracy, precision, recall, and f1 scores for both. Next we'll show the spread of message types that are included in the database through a bar chart. This will show what is available and expected for our model. And finally we will use a confusion matrix with heatmap to show how successful our model was at prediction!

Results

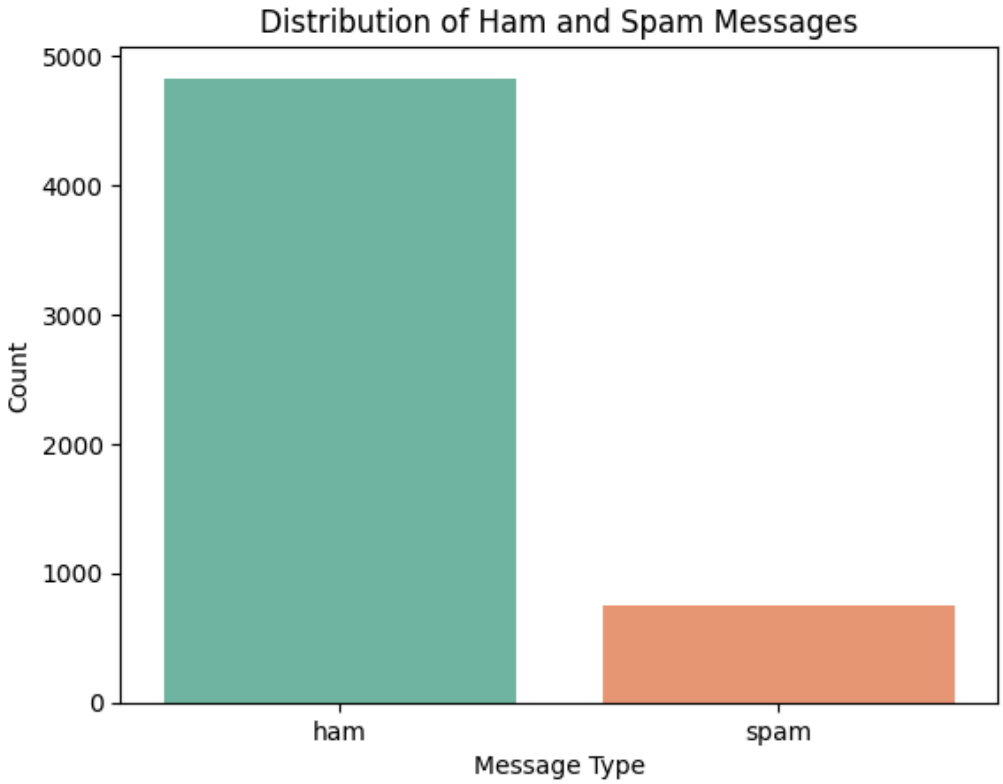
First we will start with the scores for predicting spam, predicting non-spam, and the overall model accuracy. The results are as follows:

Score Type	Ham	Spam
Precision	99%	100%
Recall	100%	94%
F1	100%	97%
Model Accuracy	99%	

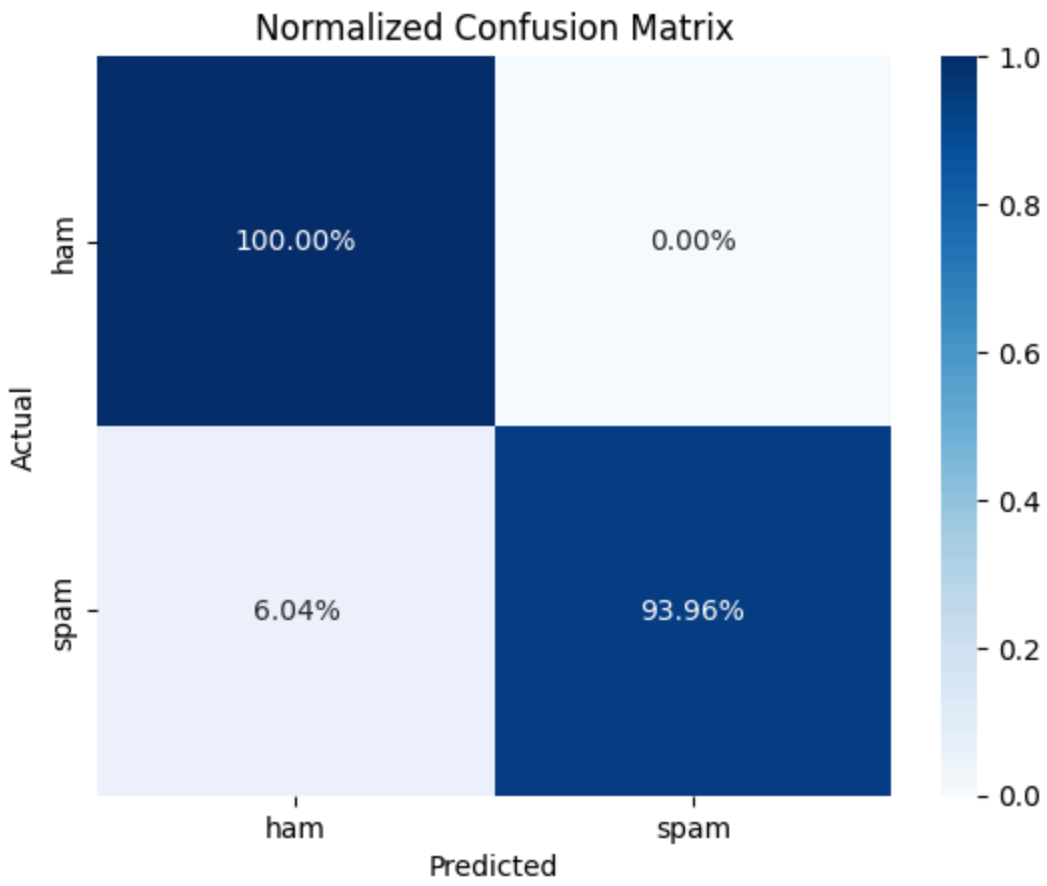
These results are all incredibly good! It shows that it is almost perfect at determining that a message is not spam, and almost just as good at detecting spam. Though these numbers are most likely due to the database we used. Because the database we used had a higher amount of Ham (Good messages) than it had Spam (Spam), it would have a higher chance of being right because it most likely would not be spam.

Speaking about amounts of Ham to Spam, let's look at the distribution. As mentioned before, the amount of spam makes up a very small percentage of the entire

set of available messages. In fact it makes up less than 1/5th, or 13.4% of the total available messages to look at. But it's still a more significant amount of spam messages than the average person would receive.



And finally, let's look at the results of our model through a normalized confusion matrix. Here we can see that if a message is not spam, it pretty much never gets tagged as such by this particular model. While it gets tagged properly as not spam pretty much every time. The spam messages are a little less certain, with about 6% accidently making it through the filter. But this is more than enough of a catch to ensure the best possible protection against phishing scams for our users. So I would call our model a success!


Normalized Confusion Matrix
100.00%
0.00%
6.04%
93.96%
ham
spam
Actual
ham
spam
Predicted
1.0
0.8
0.6
0.4
0.2
0.0