1. I decided to go with the method of adding more words to train the set on. I attempted a variety of things. Before really even starting anything I wanted to play around with where my intuition took me, use words that I know or think to be common in spam. One thing I did was look up lists of extremely common words and phrases in spam emails in addition to looking through parts of the test data set to see what kind of wording was in those. Given that info I tactically added different words that came across to me as highly associated with spam or not, ranging from different types of greetings to header text to actual content. This successfully got me up to around 93% or so pretty quickly. After that I moved onto some slightly more sophisticated analysis to try and find words to add in a more targeted way. You can see how that process went by reading through the descriptions in my EDA cells below.

2. One interesting thing I tried that did not work was running a test that looped through the entire list I had up to one point, around 200 or so, and removed any that lowered the score value. This took nearly an hour to run, only resulted in a couple words being removed, didn't increase the score by much, and had literally zero effect on the kaggle submission. I realized after that this was a bit of a silly attempt. Testing how removing things results in being able to fit the model on the training data doesn't matter that much past a certain point. You can get super focused on getting that perfect but really that just results in overfitting the model to the training data, which doesn't really help with predictions. I found much more success in focusing on the general idea of finding spam and ham in a more generalized sense during this project than to focus too closely on minor discrepencies in the training data.

3. I'm surprised by how effective just doing a word based regression prediction can be. Getting up to that first 85% correct or so isn't really that hard and getting up to even 95%+ is fairly doable just by adding words as features. I'm sure that getting towards 99/100% requires some more complicated analysis but getting pretty good isn't that hard with some basic data science and regressions. My code got up to 98.2% on Kaggle without doing any more than training my model on a bunch of words. Cleverly chosen words, but words nonetheless.