0.0.1 Question 1: Feature/Model Selection Process

In this following cell, describe the process of improving your model. You should use at least 2-3 sentences each to address the follow questions:

1. How did you find better features for your model?
2. What did you try that worked or didn't work?
3. What was surprising in your search for good features?

1. To find better features for my model, I did an explored the emails in various ways. For one, I made a function that takes in a list of words, and shows how many emails that have that word are spam, and how many are not spam. Using this I was able to pinpoint specific words that "triggered" the spam filter, in addition to finding HTML tags that "triggered" the spam filter. Finally, I looked at emails that were marked as spam and found that they had lots of punctuation and capital letters, so I added those features to my model.

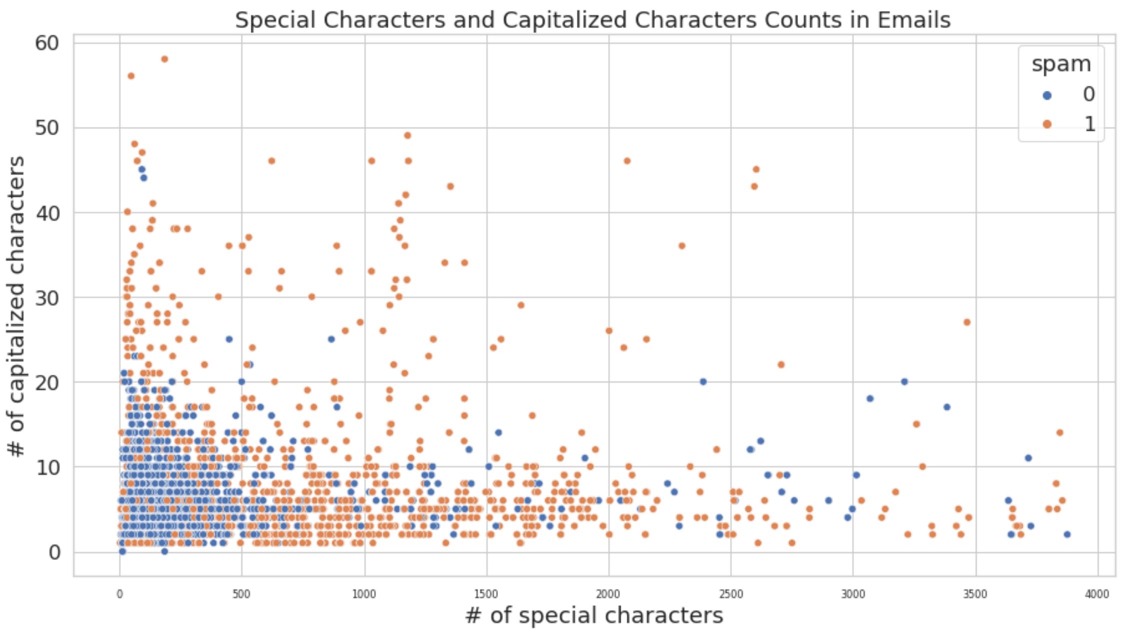
2. For my model I tried to make a decision tree to find good features for my model, but that didn't work. In addition, the words I initially used were not that good, which gave me a low accuracy for my model. Finally, I tried using string length of the emails and subjects as a feature, but they actually made my model less accurate. All the features that I mentioned in part 1 of this question worked out pretty well for me though.

3. In my search for good features, I was surprised that the length of emails didn't indicate that well whether or not an email was spam. You would think that exceptionally long emails were spam, but that wasn't the case all the time. In addition, I was surprised about the words that revealed themselves as useful during my exploratory data analysis, for example "100%" was a great indicator about whether or not an email was spam.

Question 2a Generate your visualization in the cell below.

```
In [174]: train_copy2 = train_copy.copy()
          train_copy2 = train_copy2[train_copy2["punctuation"] < 4000]
          plt.figure(figsize=(15,8))
          plot = sns.scatterplot(data=train_copy2, x="punctuation", y="capitalized", hue="spam", x_jitte
          plot.set_xlabel("# of special characters")
          plot.set_ylabel("# of capitalized characters")
          plot.set_title("Special Characters and Capitalized Characters Counts in Emails")
```

```
Out[174]: Text(0.5, 1.0, 'Special Characters and Capitalized Characters Counts in Emails')
```



Question 2b Write your commentary in the cell below.

The plot above tells us for each email the following: How many special characters does an email have, how many characters are capialized in an email, and is the email spam or not? Using this, we can see that emails with a large amount of special characters, for example “$” or “!” are more often spam than ham. Similarly, emails with many capital letters are more often spam than ham (These would be emails like “HeLlO FriENd!”, clearly not written by a human). Because of this plot, I decided to use the number of special characters and number of capitalized characters in an email as a predictor for my model.

0.0.2 Question 3: ROC Curve

In most cases we won't be able to get 0 false positives and 0 false negatives, so we have to compromise. For example, in the case of cancer screenings, false negatives are comparatively worse than false positives — a false negative means that a patient might not discover that they have cancer until it's too late, whereas a patient can just receive another screening for a false positive.

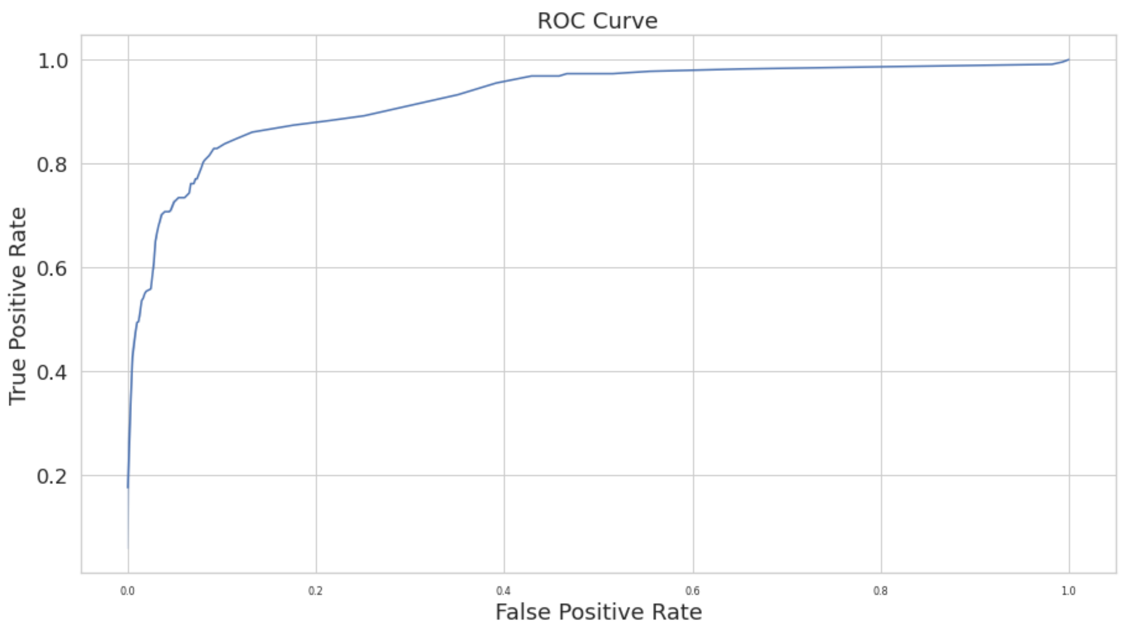
Recall that logistic regression calculates the probability that an example belongs to a certain class. Then, to classify an example we say that an email is spam if our classifier gives it ≥ 0.5 probability of being spam. However, *we can adjust that cutoff*: we can say that an email is spam only if our classifier gives it ≥ 0.7 probability of being spam, for example. This is how we can trade off false positives and false negatives.

The ROC curve shows this trade off for each possible cutoff probability. In the cell below, plot a ROC curve for your final classifier (the one you use to make predictions for Gradescope) on the training data. Refer to Lecture 20 to see how to plot an ROC curve.

Hint: You'll want to use the `.predict_proba` method for your classifier instead of `.predict` so you get probabilities instead of binary predictions.

```
In [176]: from sklearn.metrics import roc_curve

          bc_model = model
          def predict_threshold(model, X, T):
              prob_one = model.predict_proba(X)[:, 1]
              return (prob_one >= T).astype(int)

          def tpr_threshold(X, Y, T): # this is recall
              Y_hat = predict_threshold(bc_model, X, T)
              return np.sum((Y_hat == 1) & (Y == 1)) / np.sum(Y == 1)

          def fpr_threshold(X, Y, T):
              Y_hat = predict_threshold(bc_model, X, T)
              return np.sum((Y_hat == 1) & (Y == 0)) / np.sum(Y == 0)

          thresholds = np.linspace(0, 1, 100)
          tprs = [tpr_threshold(X_test, Y_test, t) for t in thresholds]
          fprs = [fpr_threshold(X_test, Y_test, t) for t in thresholds]

          plt.figure(figsize=(15,8))
          plot = sns.lineplot(x=fprs, y=tprs)
          plot.set_xlabel("False Positive Rate")
          plot.set_ylabel("True Positive Rate")
          plot.set_title("ROC Curve")
```

```
Out[176]: Text(0.5, 1.0, 'ROC Curve')
```


ROC Curve
True Positive Rate
False Positive Rate
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0