
0.1 Question 1

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. *I performed trial and error with various words. I started by looking up spam emails to find common trigger words and then building from there. Once I hit a plateau with just words, I added on punctuation and code tags as well.*
2. *I did consider the issue that I was overfitting, but I realized that many keywords were in fact necessary to distinguish between spam and ham. I also observed that some words did not have as much of an impact as other words, so I removed those less significant keywords.*
3. *I also found that using root words instead of full words added on to the accuracy. For example, using "congrat" instead of "congratulations" or "remind" instead of "reminder."*

0.2 Question 2a

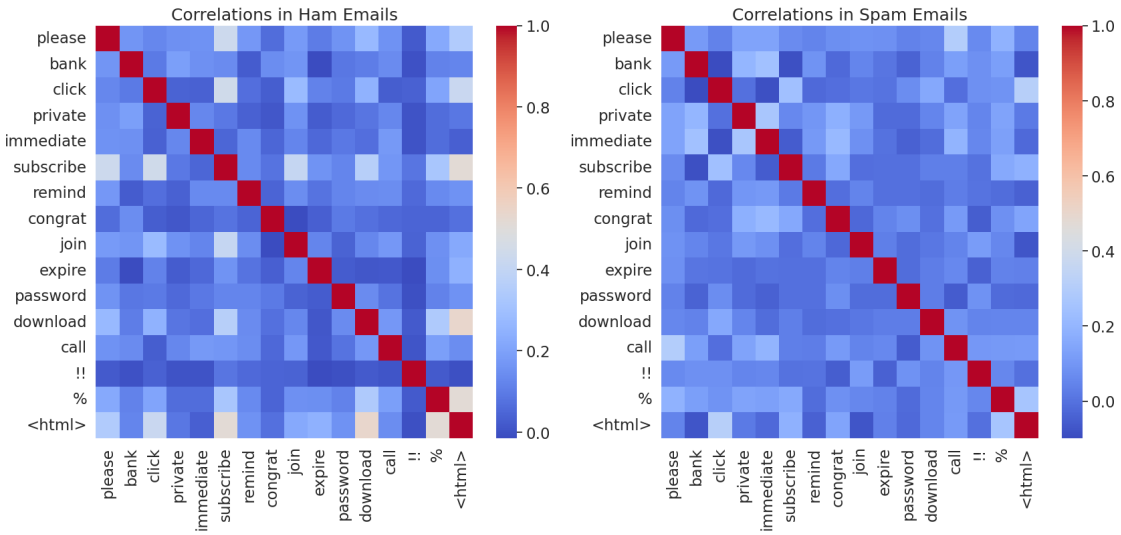
Generate your visualization in the cell below.

```
In [75]: chosen_d2_array = words_in_texts(chosen_words, train['email'])
         chosen_df = pd.DataFrame(chosen_d2_array,columns=chosen_words)
         chosen_df['type'] = train['spam'].map({1:'Spam',0:'Ham'})
         chosen_ham = chosen_df[chosen_df['type']=='Ham'].drop(columns='type')
         chosen_spam = chosen_df[chosen_df['type']=='Spam'].drop(columns='type')

         plt.figure(figsize=(8,6))
         fig,(ax1,ax2) = plt.subplots(ncols=2,figsize=(20,8))
         sns.heatmap(data=chosen_ham.corr(), ax=ax1, cmap='coolwarm')
         sns.heatmap(data=chosen_spam.corr(), ax=ax2, cmap='coolwarm')
         ax1.set_title('Correlations in Ham Emails')
         ax2.set_title('Correlations in Spam Emails')
         ;
```

```
Out[75]: ''
```

```
<Figure size 800x600 with 0 Axes>
```



0.3 Question 2b

Write your commentary in the cell below.

In order to choose which words we want as keywords to classify our emails, we want to see how significant each word is in helping distinguish spam. If multiple words are dependent on each other, then we don't necessarily need all of those keywords, we can just take one since the others depend on it. Thus, we want the correlation between different words to be very low so that they each have independent signficance on the model. Observe in the two plos above that, overall, most of the correlations are very low (indicated by blue), meaning I chose a good combination of keywords to classify spam emails. We see a slight pinkish tone in the cells of 'subscribe-<html>' and 'download-<html>', but those correlations are still quite weak so the model is good.

0.4 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 threshold*: 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.

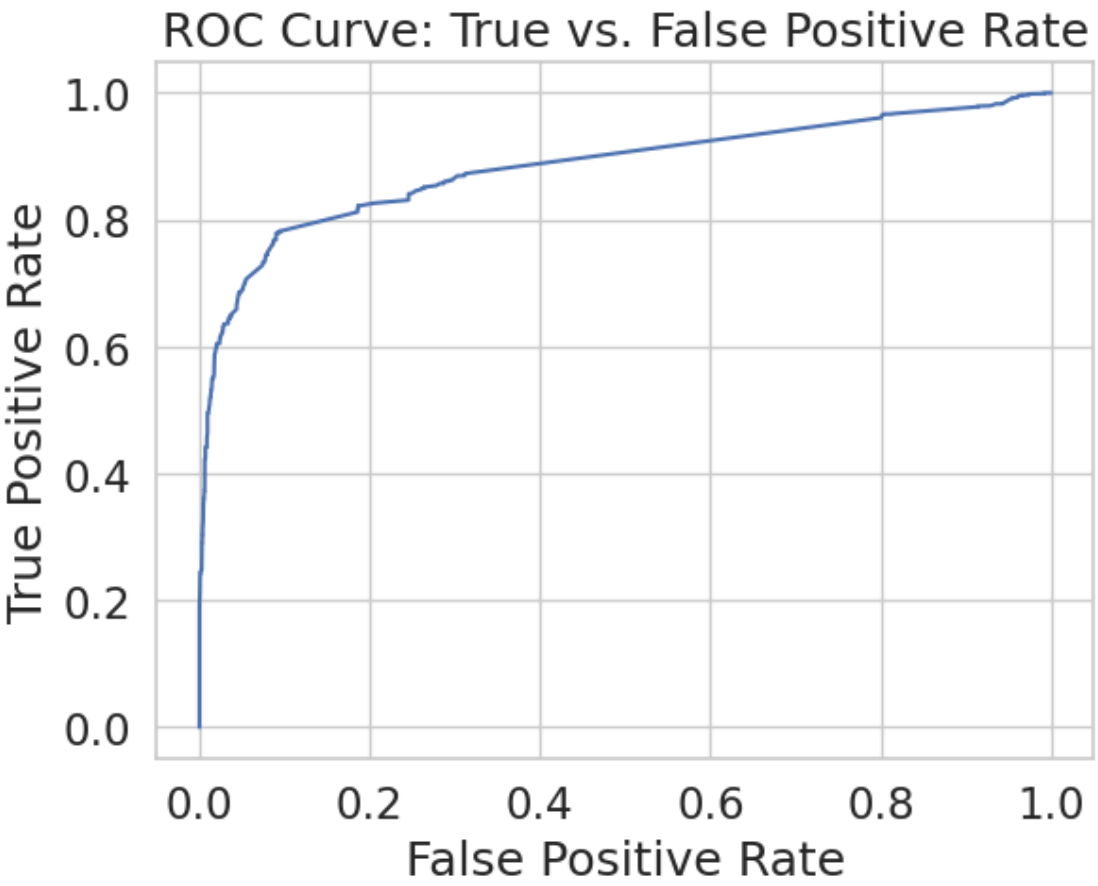
The Receiver Operating Characteristic (ROC) curve shows this trade off for each possible cutoff probability. In the cell below, plot a ROC curve for your final classifier (the one you use to make predictions for Gradescope) on the training data. Refer to Lecture 24 to see how to plot an ROC curve.

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

In [76]:
```
from sklearn.metrics import roc_curve

y_pred_proba = model.predict_proba(X_train)[::,1]
fpr, tpr, _ = roc_curve(Y_train,  y_pred_proba)

#create ROC curve
plt.plot(fpr,tpr)
plt.title('ROC Curve: True vs. False Positive Rate')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```


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