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. I found better features by focusing first on the words I was going to pass through the `words_in_texts` *function. To do this, I created the* `best_words` *function, which when given a dataframe as input with a column of string values named* `email` *and a column of 0's and 1's named* `spam`*, returns a data frame with every unique word in the* `email` *column, sorted in descending order by a column named* `spam total count` *which corresponds to the respective number of times each word appears in an email corresponding to a value of 1 in the* `spam` *column. Additionally, I created my own features which capture things such as the number of capital letters in an email, the number of question marks in an email, the length of an email, the number of dollar signs in an email, and the number of times the word 'free' appears in an email; I tested each of these features out by making a plot of the feature and the* `spam` *column to see the distribution of the features' values for 0's and 1's in the* `spam` *column of* `train`*.*

2. Initially, I tried running my `best_words` *function on the entirety of* `train`*, but my kernel crashed. Therefore, instead I decided to run the function on a random sample of 50 rows from* `train` *many times manually, to capture different words that were prevelant among emails corresponding to a value of 1 in the* `spam` *column.*

3. Something surprising in my search for good features were that individual letters in the alphabet correspond to spam emails. Additionally, when plotting the distribution of my other features to see whether they had different values for different types of emails, I saw that most of them had similar values for both spam and ham emails, with the exception of outliers. I believe this might be because most spam emails are generated in a way where the number of occurences of certain things my features captured (such as the number of dollar signs, or the word 'free' for instance) are the same; therefore, they appear as outliers, when they actually are just overlap due to multiple observations of the same number. Regardless, these features still proved to be useful in my model.

Question 2a Generate your visualization in the cell below.

```
In [23]: g = sns.JointGrid(arr_ques, train['spam'])
         plt.suptitle('Distribution of the Feature `arr_ques` for Spam/Ham Emails', size = 15)
         plt.subplots_adjust(top=0.9)
         g.plot_marginals(sns.histplot)
         g.plot_joint(sns.stripplot,
                      orient='h', order=[1, 0],
                      color=sns.color_palette()[0])
         g.ax_joint.set_xlabel("Number of question marks in each email", size = 13)
         plt.show()
```

Question 2b Write your commentary in the cell below.

For my features that were seperate from the `words_in_texts` *function, I plotted a jointgrid plot of the distribution of the feature for spam and ham emails. By doing so, I was able to analysis the linear seperability of the data and whether or not I observed different values of the feature for spam vs ham emails, so that I was able to tell whether the feature would be useful or not. As we can see with the plot above, we observe many higher values for ham emails for the feature which counts the number of questions marks per email. Additionally, the plot tells us that the data with this feature aren't linearly seperable, so it is a good feature to use in our 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.

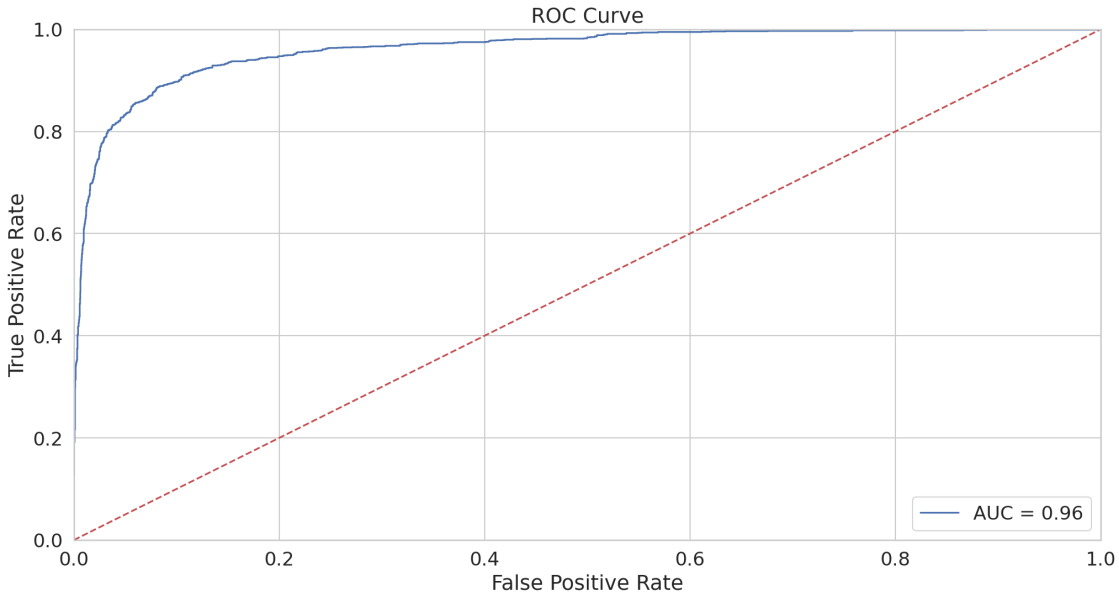
Hint: You'll want to use the `.predict_proba` method for your classifier instead of `.predict` so you get probabilities instead of binary predictions.

```
In [24]: from sklearn.metrics import roc_curve
         from sklearn.metrics import auc

         probs = modell.predict_proba(X_scaled)
         predictions = probs[:, 1]


         fpr, tpr, thresholds = roc_curve(train['spam'], predictions)
         roc_auc = auc(fpr, tpr)

         plt.title('ROC Curve')
         plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
         plt.legend(loc = 'lower right')
         plt.plot([0, 1], [0, 1],'r--')
         plt.xlim([0, 1])
         plt.ylim([0, 1])
         plt.ylabel('True Positive Rate')
         plt.xlabel('False Positive Rate')
         plt.show()
```


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 0.96