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 first split my data into spam and ham emails and obtain individual words from each email. I then made a dictionary for spam and ham words and found out the number of times each word appeared in each category of emails. I then found the top 20 words for spam and ham emails individually and figured out which top words were in spam but not in ham in order to find better spam words to distinguish spam emails.
2. I first tried to create a bar graph to visualize 50 words with the exact counts of each word, but I quickly realized that the visualization got messy with very high counts for certain words. Then I divided the counts by 2000 to get a cleaner graph with smaller counts for each word. I also plotted 20 words instead of 50.
3. I was surprised to see that 'nbsp' was in one of the top 20 words for spam emails as it didn't look like a real English word. After further research, I found out that html code uses 'nbsp' often, which means non-breaking space that prevents an automatic line break. Based on my original analysis in hw12, I mentioned that a pattern I found was spam emails seemed to be mostly coded in html, which makes sense that 'nbsp' is a common word in spam but not in ham.

Question 2a Generate your visualization in the cell below.

```
In [85]: spam = train[train['spam']==1]
         spam_split[:10]
```

```
Out[85]: 5     [secatt, 000, 1fuklemuttfusq, content, type, t…
         13    [free, adult, lifetime, membership, limited, t…
         15    [table, width, 600, border, 20, align, center,…
         18    [html, head, head, body, center, table, border…
         19    [html, head, head, center, h1, b, font, face, …
         20    [i, saw, your, email, on, a, website, i, visit…
         24    [html, body, p, align, center, b, font, size, …
         27    [dear, opportunities, seekers, i, thought, you…
         28    [this, is, a, multi, part, message, in, mime, …
         32    [html, body, onload, window, open, http, 202, …
         Name: email, dtype: object
```

```
In [86]: #Count size
         size_counter_per_email=[]
         for email in spam_split:
             counter=0
             for word in email:
                 if word == 'size':
                     counter+=1
             size_counter_per_email.append(counter)
         size_counter_per_email[:10]
```

```
Out[86]: [0, 0, 5, 23, 114, 0, 4, 0, 0, 8]
```

```
In [87]: #Count color
         color_counter_per_email=[]
         for email in spam_split:
             counter=0
             for word in email:
                 if word == 'color':
                     counter+=1
             color_counter_per_email.append(counter)
         color_counter_per_email[:10]
```

```
Out[87]: [0, 0, 5, 21, 36, 0, 1, 0, 0, 0]
```

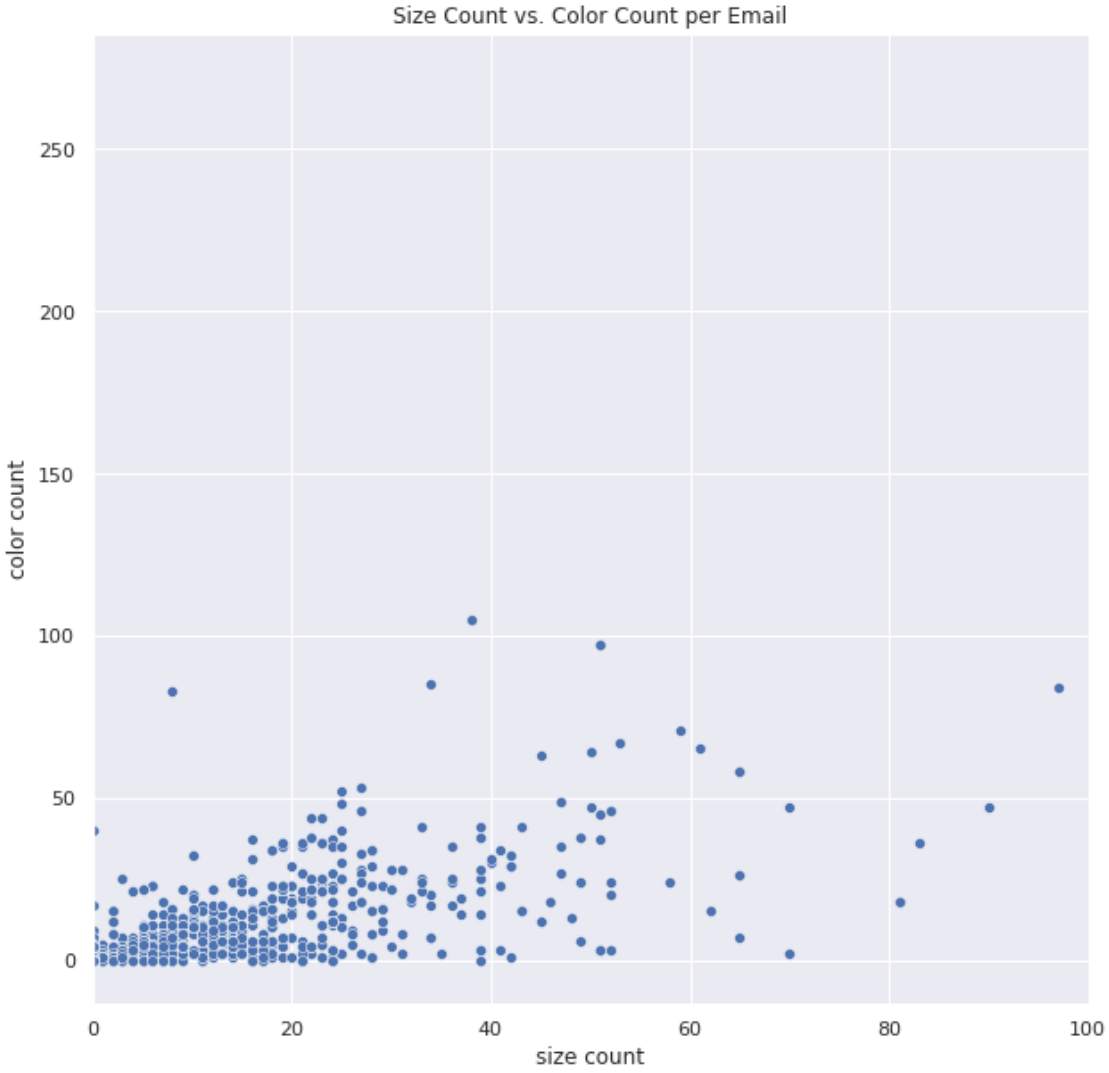
```
In [88]: spam['size count'] = size_counter_per_email
         spam['color count'] = color_counter_per_email
```

```
/tmp/ipykernel_377/148579859.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.
  spam['size count'] = size_counter_per_email
/tmp/ipykernel_377/148579859.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.
  spam['color count'] = color_counter_per_email
```

```
In [89]: my_visualization = sns.scatterplot(data=spam, x='size count', y='color count').set(title='Size
         plt.xlim(0, 100)
         my_visualization;
```


Size Count vs. Color Count per Email
color count
size count
0
50
100
150
200
250
0
20
40
60
80
100

Question 2b Write your commentary in the cell below.

I used the 'Size Count vs. Color Count per Email' scatterplot to see if there is an association between the count of the word 'size' and the word 'color' for each of the spam emails. Earlier I found out that the top most common spam email words include 'size' and 'color'. 'size' and 'color' are both words commonly used in html code and since spam emails are generally coded with html I thought there would be an association with the count of the two words per email. Based on the scatterplot there doesn't seem to be a linear association between 'size' and 'color' counts.

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 [90]: x_train=words_in_texts(spam_not_ham, train['email'])
         y_train= train['spam']

         my_model=LogisticRegression().fit(x_train,y_train)
         y_pred = my_model.predict(x_train)

         training_accuracy = sum(y_pred==y_train)/len(y_train)
         training_accuracy
```

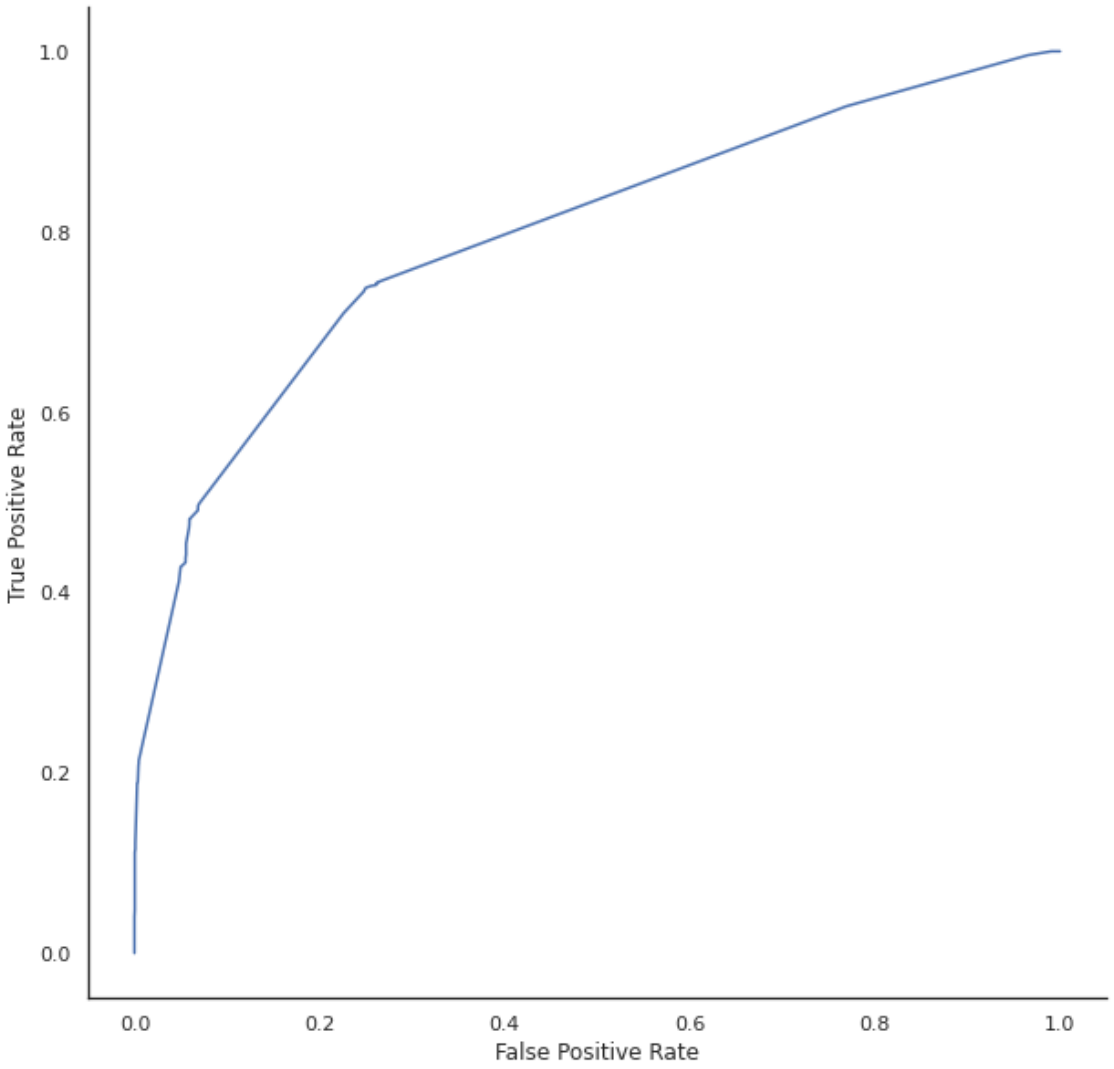
```
Out[90]: 0.8221748968454679
```

```
In [91]: from sklearn.metrics import roc_curve

         y_hat = my_model.predict_proba(x_train)[:,1]
         y_hat

         fpr, tpr, thresholds = roc_curve(y_train, y_hat)
         with sns.axes_style("white"):
             plt.plot(fpr, tpr)

         sns.despine()
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate');
```


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