0.0.1 Question 1c

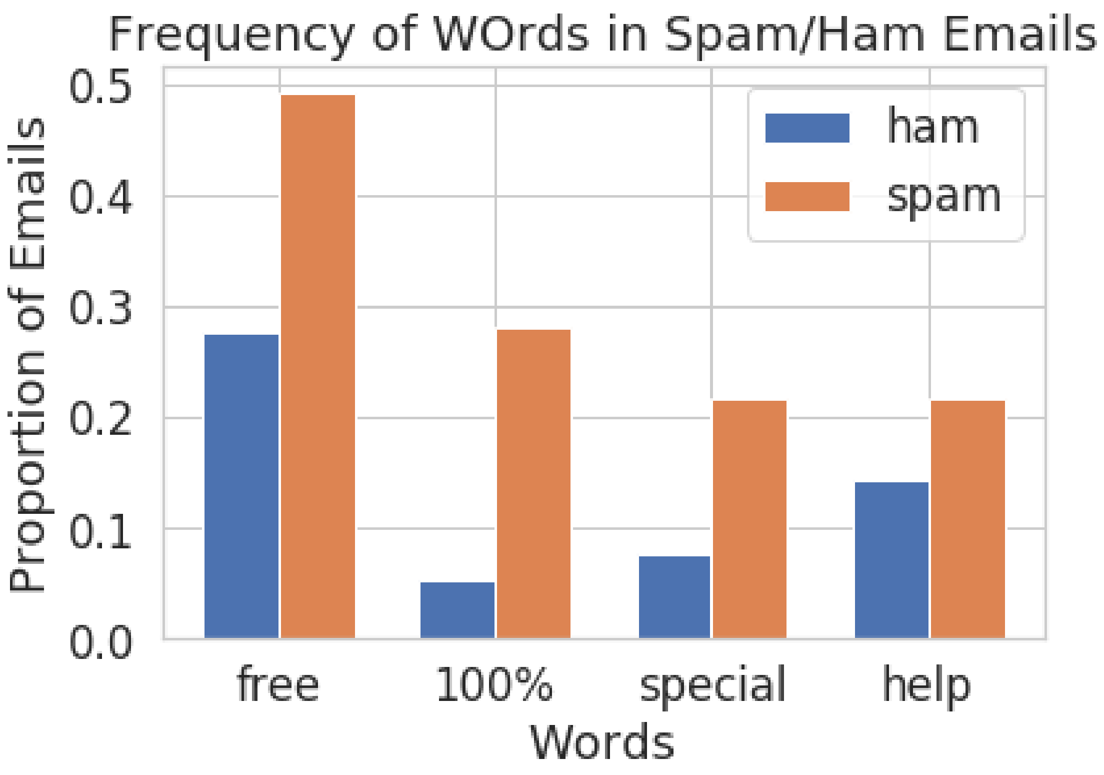
Discuss one thing you notice that is different between the two emails that might relate to the identification of spam.

The links in the spam email are not a formal website, but rather an IP address '209.163.187.47'. The ham email uses more reputable links to redirect its receiver.

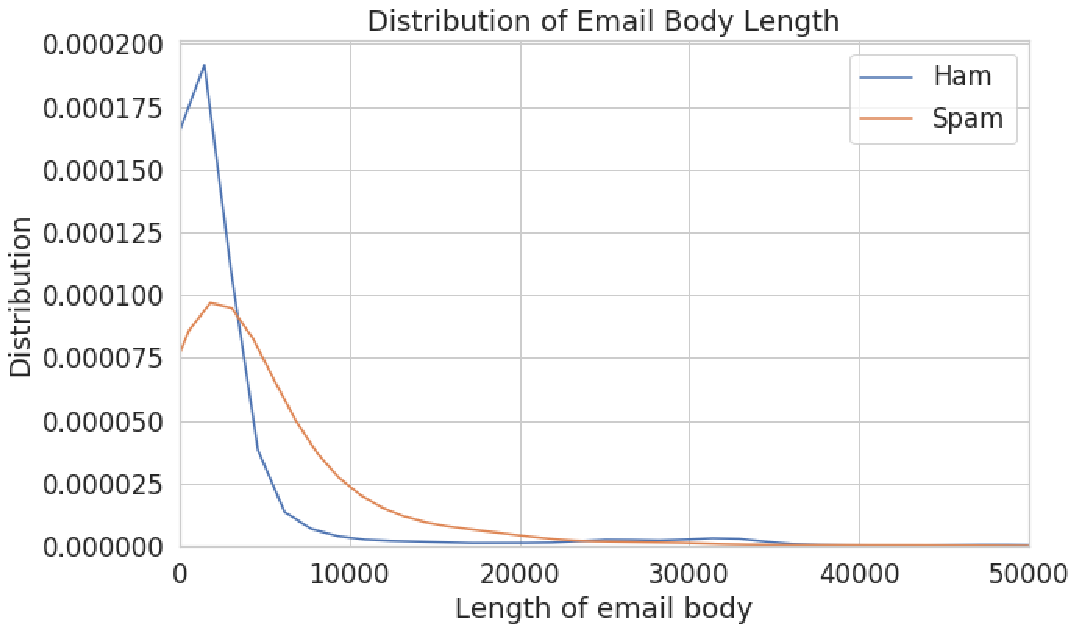
0.0.2 Question 3a

Create a bar chart like the one above comparing the proportion of spam and ham emails containing certain words. Choose a set of words that are different from the ones above, but also have different proportions for the two classes. Make sure to only consider emails from `train`.

```
In [13]: train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai

         words = ['free', '100%', 'special', 'help']
         ham = train[train['spam'] == 0]['email']
         spam = train[train['spam'] == 1]['email']
         ham_count = words_in_texts(words, ham)
         spam_count = words_in_texts(words, spam)

         plt.bar(x = words, align = 'edge', height = np.sum(ham_count, axis = 0)/len(ham_count), label =
         plt.bar(x = words, align = 'edge', height = np.sum(spam_count, axis = 0)/len(spam_count), label
         plt.legend()
         plt.xlabel('Words')
         plt.ylabel("Proportion of Emails")
         plt.title("Frequency of WOrds in Spam/Ham Emails")
         plt.show()
```

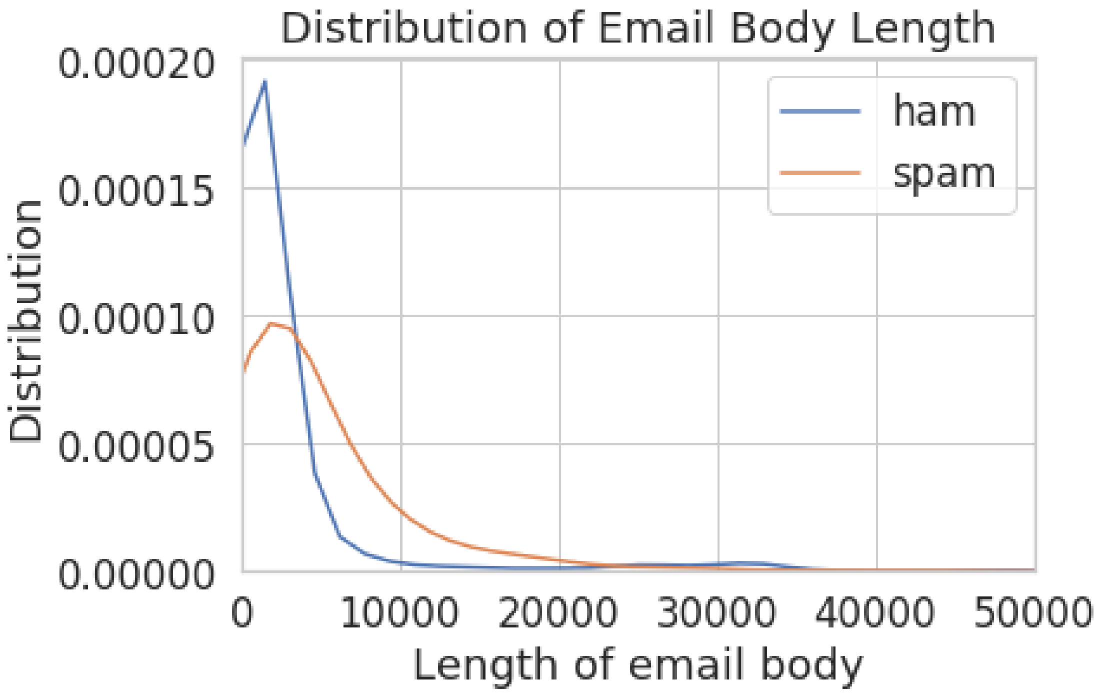


0.0.3 Question 3b



Create a *class conditional density plot* like the one above (using `sns.distplot`), comparing the distribution of the length of spam emails to the distribution of the length of ham emails in the training set. Set the x-axis limit from 0 to 50000.

```
In [14]: len_ham = [len(x) for x in ham]
         len_spam = [len(x) for x in spam]

         plt.xlim(0, 50000)

         sns.distplot(len_ham, label = 'ham', hist = False)
         sns.distplot(len_spam, label = "spam", hist = False)

         plt.legend()
         plt.xlabel("Length of email body")
         plt.ylabel("Distribution")
         plt.title("Distribution of Email Body Length")
         plt.savefig('training_conditional_densities.png')
```



Distribution of Email Body Length
Distribution
Length of email body
ham
spam
0.00020
0.00015
0.00010
0.00005
0.00000
0
10000
20000
30000
40000
50000

0.0.4 Question 6c

Provide brief explanations of the results from 6a and 6b. Why do we observe each of these values (FP, FN, accuracy, recall)?

False Positive: a ham email that gets flagged as spam. With a zero predictor, the hame will always be flagged as ham, thus this will result in 0.

False Negative: all spam emails will get flagged as ham and thus is the legnth of actual spam emails in data.

accuracy: number of accurate responses. Since we are always predicting 0 it will the length of ham emails divided by the total number of emails.

recall: recall would be the proportionl of spam emails acurrately flagged spam. As we dont flag spam, we will get a rate of 0.

0.0.5 Question 6e

Are there more false positives or false negatives when using the logistic regression classifier from Question 5?

There are more False Negatives (1699 of them). (Only 122 FP)

0.0.6 Question 6f

1. Our logistic regression classifier got 75.76% prediction accuracy (number of correct predictions / total). How does this compare with predicting 0 for every email?
2. Given the word features we gave you above, name one reason this classifier is performing poorly. Hint: Think about how prevalent these words are in the email set.
3. Which of these two classifiers would you prefer for a spam filter and why? Describe your reasoning and relate it to at least one of the evaluation metrics you have computed so far.

1. Our zero predictor has a 74.47% prediction. Our predictor at 75.76% is slightly better than the zero predictor.

2. The words may be a good classifier in only some emails but may not be a great indicator for emails that dont contain those words.

3. Our predictor has a higher recall. Thus it classifies spam a lot better than the 0 preditor.

0.0.7 Question 7: 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:

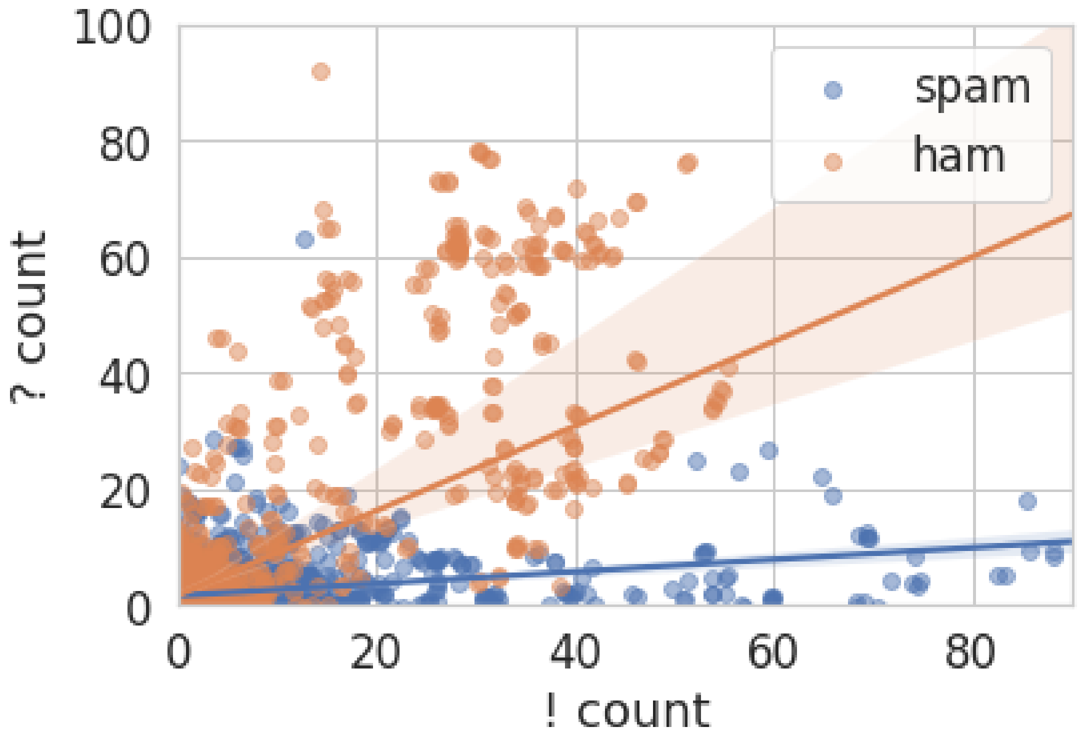
1. How did you find better features for your model?
2. What did you try that worked or didn't work?
3. What was surprising in your search for good features?

1. In order to find better features for the model, I started with looking at the feature suggestions in the beginning of this section. Measuring the features, I had looked for a major difference in a distribution between spam or ham. If there was a major difference, I kept the feature.
2. I tried many differnt features that did not work out such as I did not notice a difference in character length of a spam email compared to a ham email. On the other hand, some worked such as Reply emails, where most Re: emails were ham emails and more Spam emails contained IP addresses.
3. I noticed that I was not able to properly locate capital letters in emails but that was because we used .lower() in the beginning of the project. Something else that I found surprising was that emails with "Re:" were more likely ham rather than spam.

Generate your visualization in the cell below and provide your description in a comment.

```
In [143]: # Write your description (2-3 sentences) as a comment here:
          # The exclamation points (!) and question mark(?) often occur together very frequently.
          # These punctuation marks follow a similar trend. In ham emails, ?s range from 0 to 80 counts
          # On the other hand, !s occur from 0 to 90 in spam emails but in ham emails range from 0 to 6
          # This makes a lot of sense as ham emails tend to be asking a lot more questions while spam e
          # When it comes to (!)s there is generally higher rates in spam as they are looking to excite

          # Write the code to generate your visualization here:

          def exclTag(df):
              return df['email'].str.findall('!').str.len()

          def quesTag(df):
              return df['email'].str.findall('\?').str.len()

          ax = sns.regplot(x = exclTag(spam), y = quesTag(spam),
                            x_jitter= 0.5, y_jitter = 0.5, label = "spam",
                           scatter_kws = {"alpha" : 0.5})

          ax = sns.regplot(x = exclTag(ham), y = quesTag(ham),
                            x_jitter= 0.5, y_jitter = 0.5, label = "ham",
                           scatter_kws = {"alpha" : 0.5})

          ax.legend()
          plt.xlabel('! count')
          plt.ylabel('? count')
          plt.xlim(0,90)
          plt.ylim(0,100)
```

```
Out[143]: (0, 100)
```



spam
ham
? count
! count
0
20
40
60
80
100

0.0.8 Question 9: ROC Curve

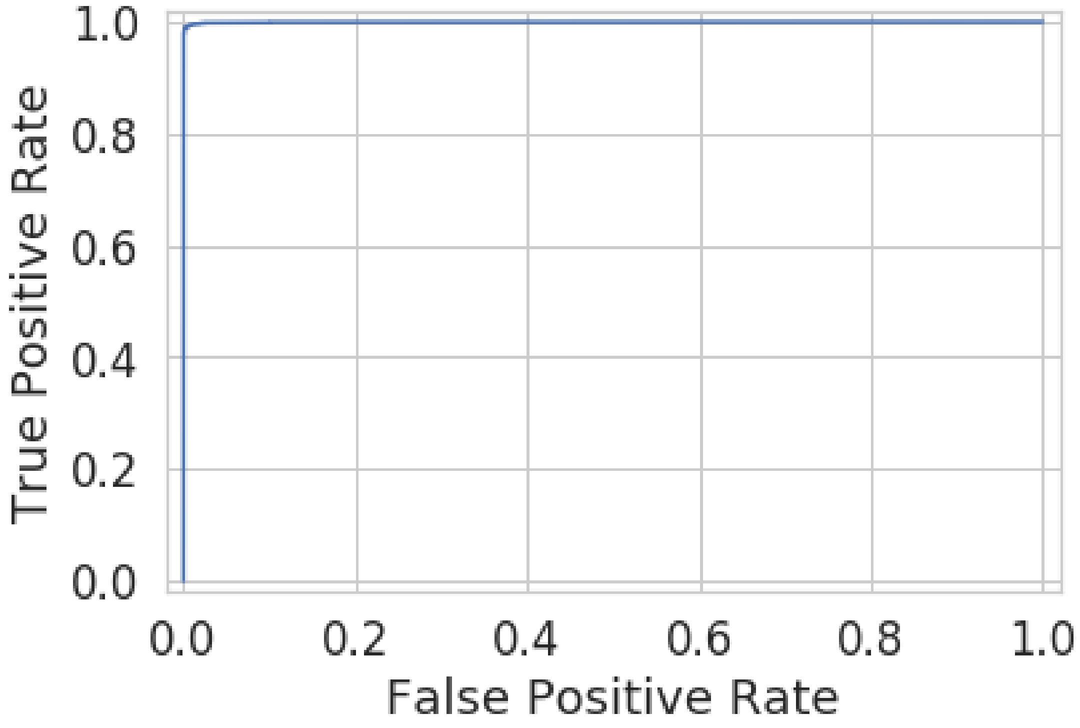
In most cases we won't be able to get 0 false positives and 0 false negatives, so we have to compromise. For example, in the case of cancer screenings, false negatives are comparatively worse than false positives — a false negative means that a patient might not discover that they have cancer until it's 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 19 or Section 17.7 of the course text to see how to plot an ROC curve.

```
In [144]: from sklearn.metrics import roc_curve

          # Note that you'll want to use the .predict_proba(...) method for your classifier
          # instead of .predict(...) so you get probabilities, not classes

          fpr, tpr, _ = roc_curve(y_train, [i[1] for i in model.predict_proba(x_train)])

          plt.step(fpr, tpr)

          plt.xlabel("False Positive Rate")
          plt.ylabel("True Positive Rate")
          plt.ylim([-0.02, 1.02])
          plt.xlim([-0.02, 1.02])
```

```
Out[144]: (-0.02, 1.02)
```


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