0.0.1 Question 1c

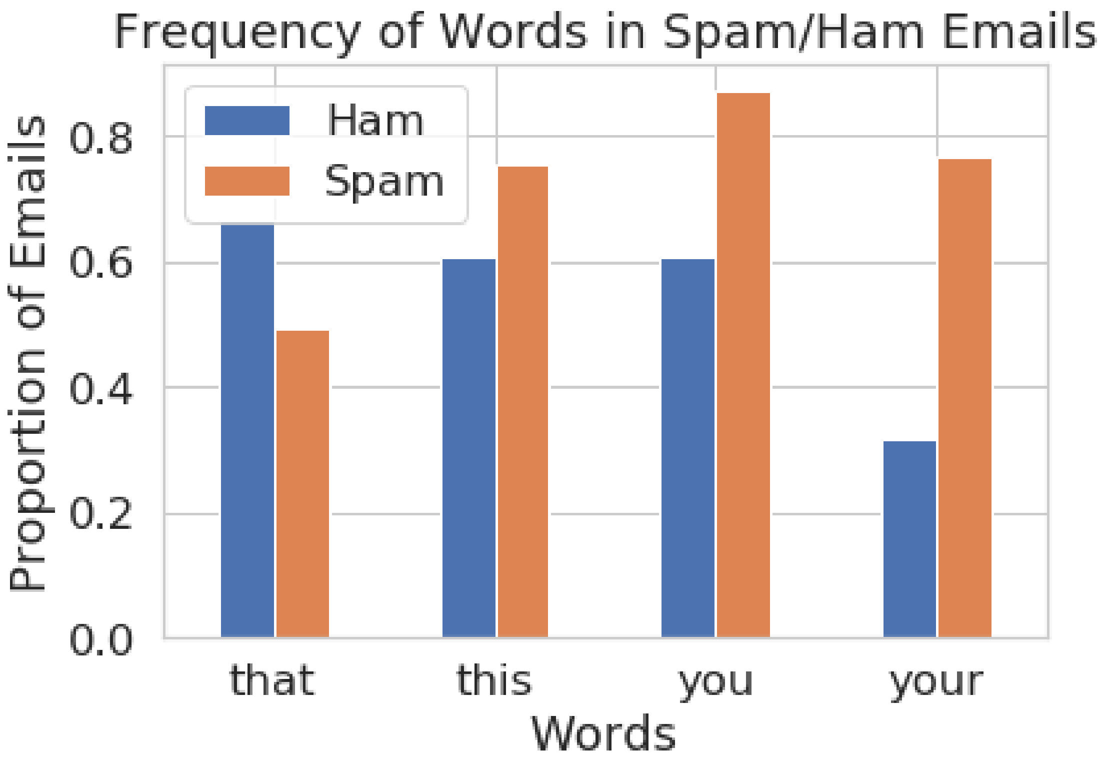
Discuss one thing you notice that is different between the two emails that might relate to the identification of spam.

One thing I notice is that the spam email is written in html while the ham email is not.

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 [100]: train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of ema
          num_spam = train['spam'].sum()
          num_ham = len(train['spam']) - train['spam'].sum()

          def convert_spam_ham(series):
              new_array = []
              for x in series:
                  if x == 0:
                      new_array.append('ham')
                  else:
                      new_array.append('spam')
              return new_array

          spam_indicator_text = ['you', 'your', 'this', 'that']

          text = train['email']
          indicator_array = words_in_texts(spam_indicator_text, text)
          df = pd.DataFrame(indicator_array, columns = spam_indicator_text).assign(type = convert_spam_h
          df_melted = df.melt('type')
          df_melted_sorted = df_melted.groupby(['variable', 'type']).sum().unstack()
          df_melted_sorted = df_melted_sorted['value'].assign(ham = df_melted_sorted['value']['ham']/nu
                                          spam = df_melted_sorted['value']['spam']/num_spam)
          df_melted_sorted.plot(kind = 'bar')
          plt.legend(['Ham', 'Spam'])
          plt.xticks(rotation=0)
          plt.xlabel('Words')
          plt.ylabel('Proportion of Emails')
          plt.title('Frequency of Words in Spam/Ham Emails')
          plt.show();
```

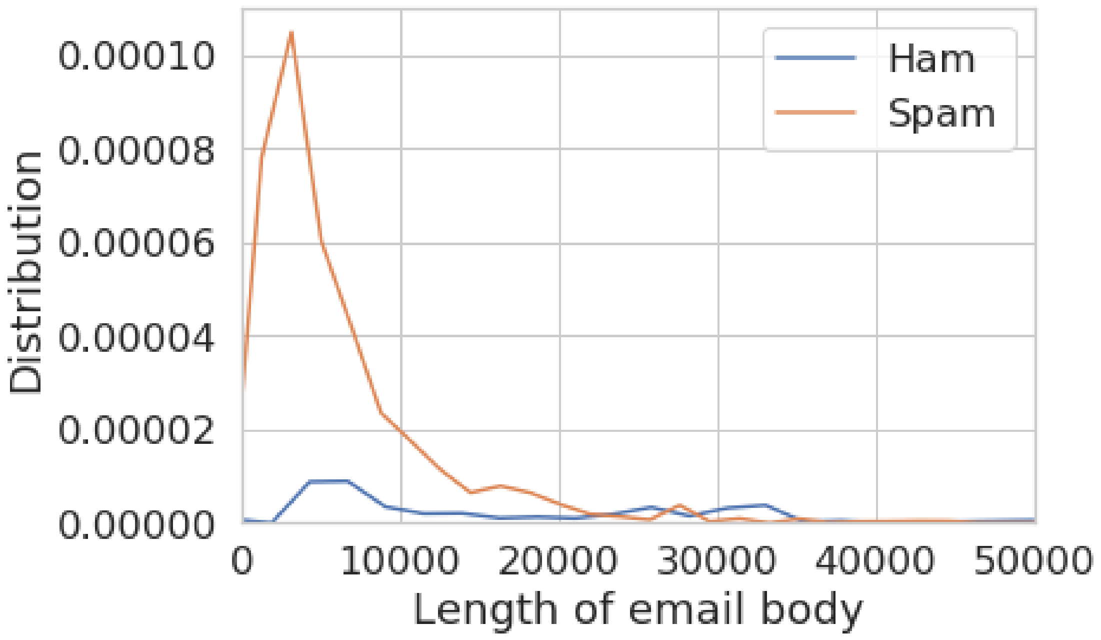

Frequency of Words in Spam/Ham Emails
Ham
Spam
Proportion of Emails
0.0
0.2
0.4
0.6
0.8
that
this
you
your
Words

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 [101]: ham = train.query("spam == 0")
          spam = train.query("spam == 1")
          ham_length = np.array([len(i) for i in ham["email"]])
          spam_length = np.array([len(i) for i in spam["email"]])
          sns.distplot(ham_length, hist = False, label = "Ham")
          sns.distplot(spam_length, hist = False, label = "Spam")
          plt.legend()
          plt.xlim(0, 50000)
          plt.xlabel("Length of email body")
          plt.ylabel("Distribution")
```

```
Out[101]: Text(0, 0.5, 'Distribution')
```



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)?

Since the zero predictor is predicting 0 for every email, this means that there are no positive (1) predictions. Thus, there will be neither false positive, nor true positives as there are no positive predictions at all. Instead, the number of true negatives will be equal to the number of actual ham emails, and the number of false negatives will be equal to the number of actual spam emails. Thus, accuracy will be the number of correct values over the total number of values in general which is just the number of true negatives, which we know is just the number of actual ham emails all over the number of total emails. For recall, we know that the number of true positives is 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 fewer false negatives and more false positives when using the logistic classifier than that in question 5 since the logistic regression classifier will identify some spam emails in comparison to the zero predictor classifier will miss the spam emails.

0.0.6 Question 6f

1. Our logistic regression classifier got 75.6% 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.

Our logistic regression classifier gets only slightly better compared with predicting 0 for every email (with 72.4% accuracy of predicting all zeros). The "some_words" list is loosely relevant to the prediction of whether the email is spam or not, as spam emails such as advertisements would also use words such as "drug", "banks" and "prescription". In terms of prediction accuracy, I would prefer the logistic regression classifier since it will actually catch some of the spam emails versus the zero-predictor classifier which will let all of the spam emails through. We can see a direct numerical result of this in the recall value of the two sets, where the zero-predictor classifier has a recall of 0 meaning it didn't catch any of the spam emails versus the logistic regression classifier which managed to catch about 11% of the actual spam emails.

0.0.7 Question 7: Feature/Model Selection Process

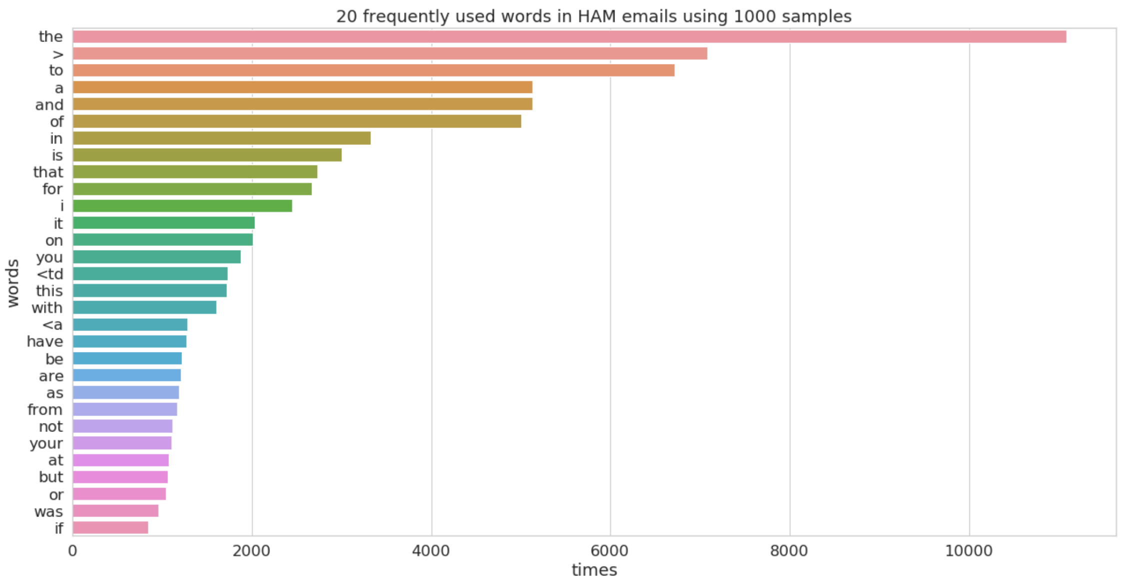
In this following cell, describe the process of improving your model. You should use at least 2-3 sentences each to address the follow questions:

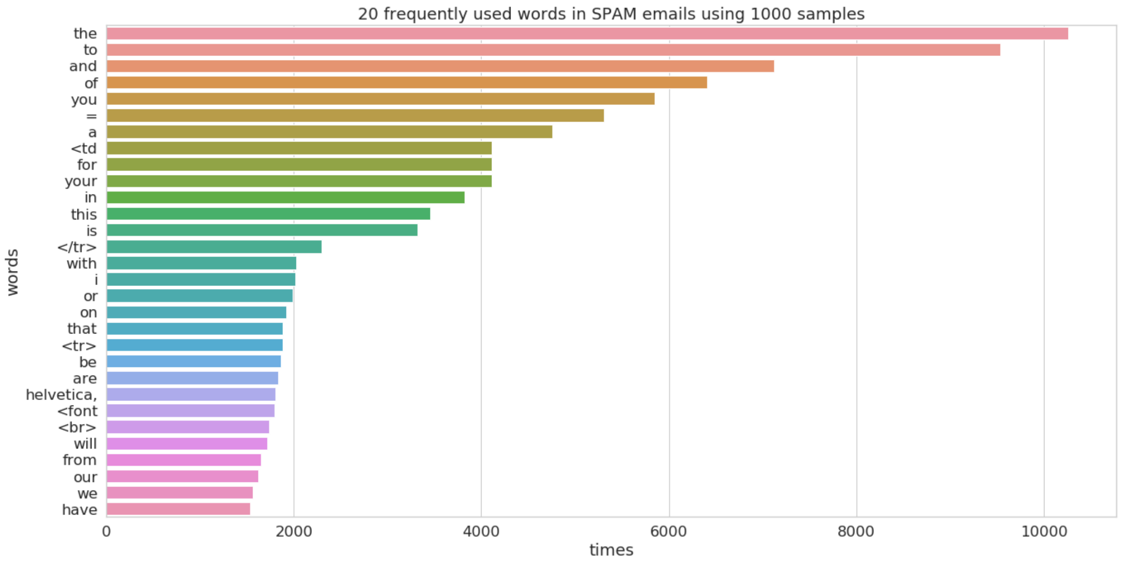
1. How did you find better features for your model?
2. What did you try that worked / didn't work?
3. What was surprising in your search for good features?

I found the features for my model by observing some patterns that I saw in the data and then running these features through 10-fold cross-validation. First, I noticed that many spam emails have lots of capital letters, and so created a method which returns an array of the number of capital letters in each email. I also took into account the total body length of the email itself and whether or not the email is a reply and contains the string 'Re:' in the subject line. Then, I explored how the punctuation would be a good indicator of whehter of not it is a spam email. At the end, the 10-fold cross validation results have made me believe that this model, despite having quite a few features, has not terribly overfit the data and will yield results of 88% or above accurac.

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

```
In [117]: # Write your description (2-3 sentences) as a comment here:
          # In this cell I am ploting the 20 most frequently used words in spam and ham emails using 10
          # From the figure we can find out that although there is a big overlap between words both kin
          # it shows a tendency for some word choice that can help us make distinguishment, for intance
          # use of "you", "your" and "this" in spam email than ham email, and ham emails use "that" mor

          # Write the code to generate your visualization here:

          # split the words from 1000 spam and ham emails
          splited_ham = train.query("spam == 0").iloc[0:1000]['email'].str.split()
          splited_spam = train.query("spam == 1").iloc[0:1000]['email'].str.split()

          # get rid of the "list" seperation between words
          ham_word_list = sum(splited_ham, [])
          spam_word_list = sum(splited_spam, [])

          # count the 10 most frequently used words in both lists
          ten_ham = pd.Series(ham_word_list).value_counts(ascending=False).head(30)
          ten_spam = pd.Series(spam_word_list).value_counts(ascending=False).head(30)

          # create two seperate bar chart of visualization
          plt.figure(figsize = (20, 10))
          plt.ylabel('words')
          plt.xlabel('times')
          plt.title('20 frequently used words in HAM emails using 1000 samples')
          sns.barplot(ten_ham.values, ten_ham.index)

          plt.figure(figsize = (20, 10))
          plt.ylabel('words')
          plt.xlabel('times')
          plt.title('20 frequently used words in SPAM emails using 1000 samples')
          sns.barplot(ten_spam.values, ten_spam.index)


Out[117]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbca0d68d90>
```


20 frequently used words in HAM emails using 1000 samples
words
times
the
>
to
a
and
of
in
is
that
for
i
it
on
you
<td
this
with
<a
have
be
are
as
from
not
your
at
but
or
was
if
0
2000
4000
6000
8000
10000


20 frequently used words in SPAM emails using 1000 samples
words
times
the
to
and
of
you
=
a
<td
for
your
in
this
is
</tr>
with
i
or
on
that
<tr>
be
are
helvetica,
<font

will
from
our
we
have
0
2000
4000
6000
8000
10000

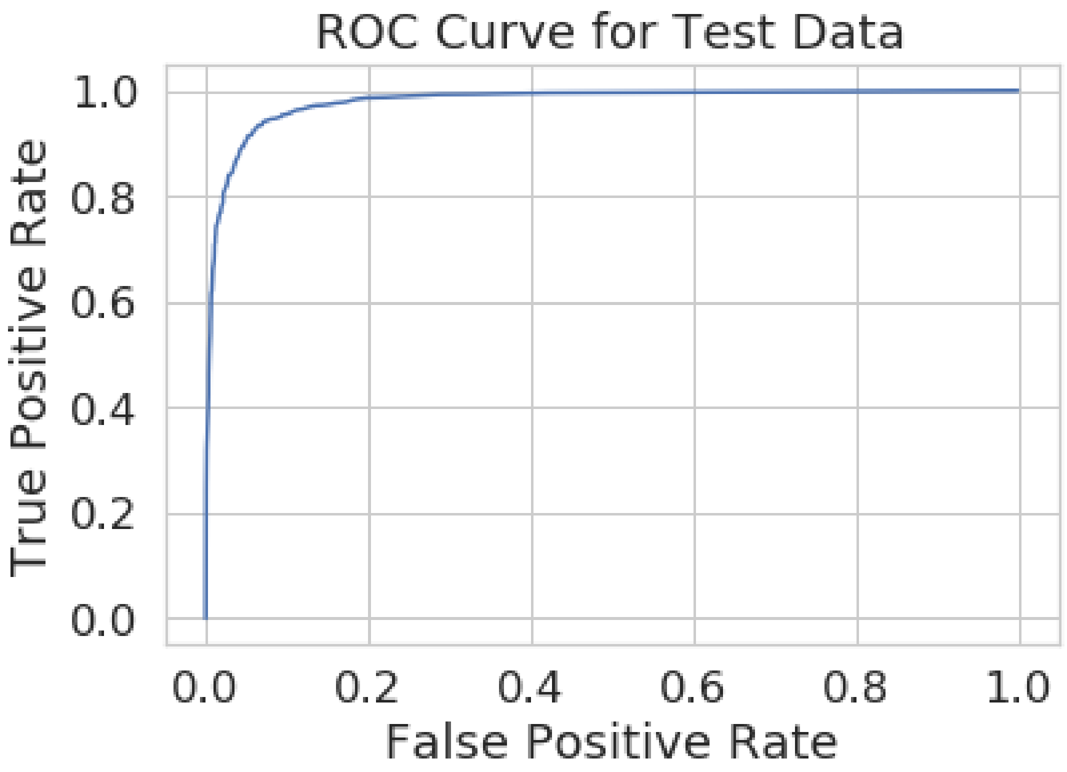
0.0.8 Question 9: ROC Curve

In most cases we won't be able to get no false positives and no 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 a disease until it's too late to treat, while a false positive means that a patient will probably have to take another screening.

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 [122]: from sklearn.metrics import roc_curve

          thresholds = np.linspace(0, 1, 101)
          def compute_TPR_FPR(y_obs, y_hat):
              TP = sum((y_obs == y_hat) & (y_obs == 1))
              FN = sum((y_obs != y_hat) & (y_obs == 1))
              TPR = TP / (TP + FN)

              TN = sum((y_obs == y_hat) & (y_obs == 0))
              FP = sum((y_obs != y_hat) & (y_obs == 0))
              FPR = FP / (FP + TN)

              return TPR, FPR


          TPRs_best_model = []
          FPRs_best_model = []
          y_hat = final_model.predict_proba(X_final)

          for threshold in thresholds:
              new_yhat = y_hat[:,:1].reshape(1, 7513)[0]
              new_yhat = np.where(new_yhat < threshold, 1, 0)
              TPR, FPR = compute_TPR_FPR(Y_train.values, new_yhat)
              TPRs_best_model.append(TPR)
              FPRs_best_model.append(FPR)

          plt.plot(FPRs_best_model, TPRs_best_model)
          plt.xlabel('False Positive Rate')
          plt.ylabel('True Positive Rate')
          plt.title('ROC Curve for Test Data');
```


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