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 second email is written in html, as opposed to the other one, which isnt and rather more ‘human’. Another indicator is that the link of the second one goes to some ip address rather than a website with “.com” like the first email.

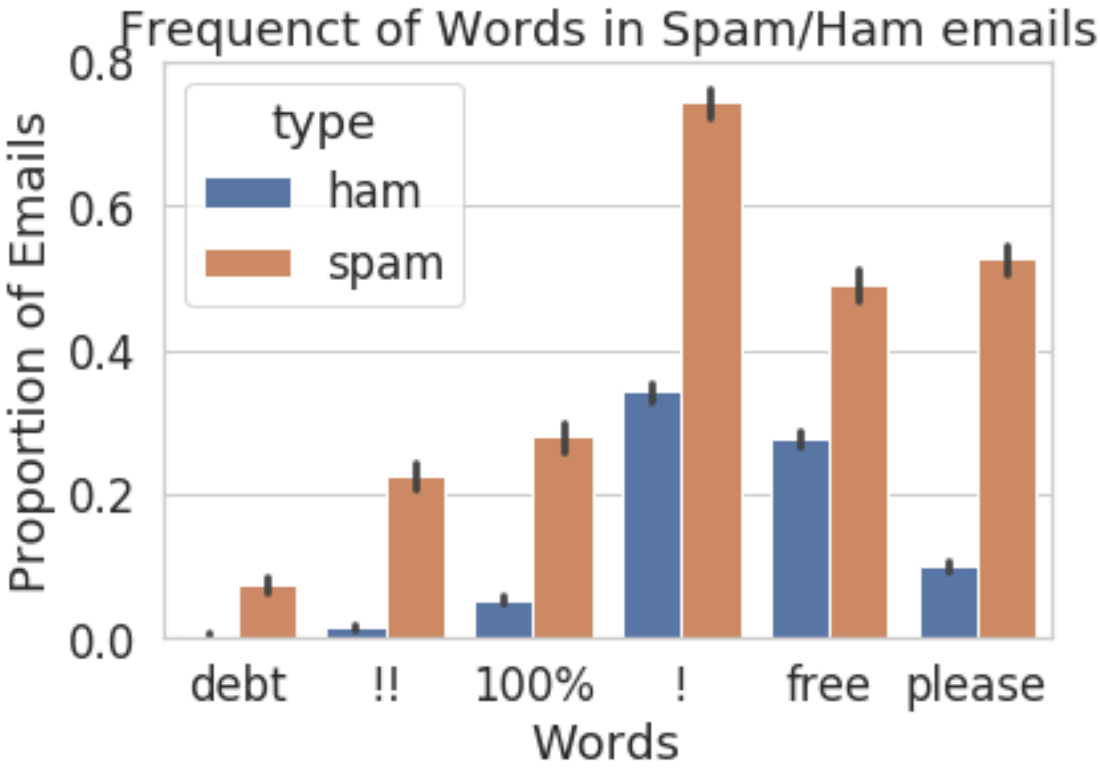
0.0.2 Question 3a

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`.

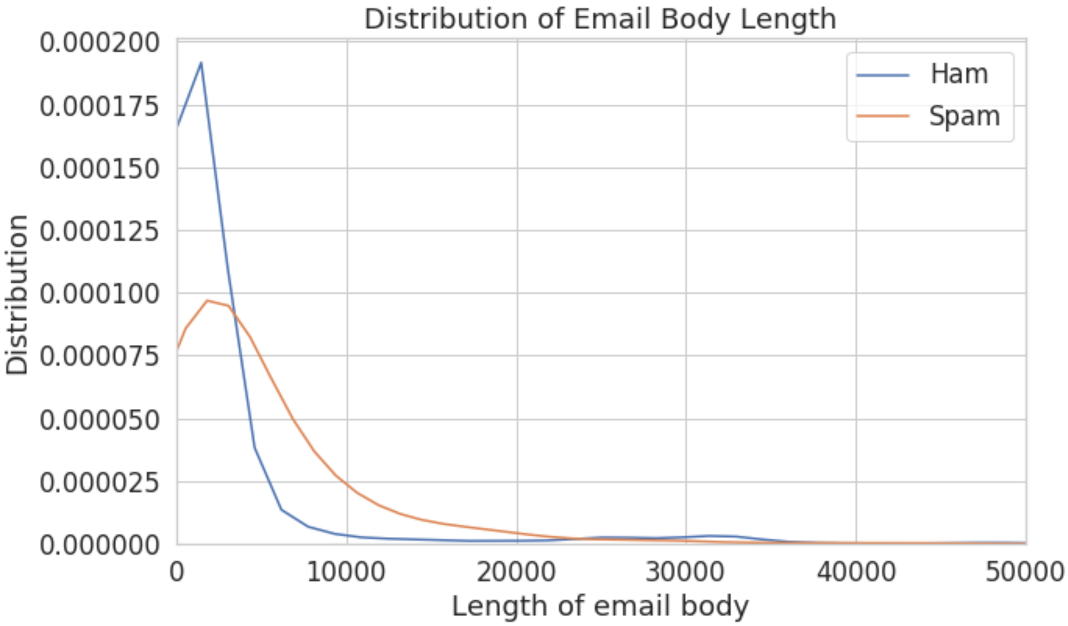
```
In [321]: train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of ema
          subdata = pd.DataFrame()
          w = words_in_texts(['debt', '!!', '100%', "!", 'free', "please"], train['email'])
          subdata['debt'] = [x[0] for x in w]
          subdata['!!'] = [x[1] for x in w]
          subdata['100%'] = [x[2] for x in w]
          subdata['!'] = [x[3] for x in w]
          subdata['free'] = [x[4] for x in w]
          subdata['please'] = [x[5] for x in w]
          subdata['type'] = train['spam'].apply(str).str.replace('0', 'ham').str.replace('1', 'spam')
          subdata = subdata.melt("type")
          sns.barplot(data = subdata, x = "variable", y = "value", hue = 'type')
          plt.title("Frequenct of Words in Spam/Ham emails")
          plt.ylabel("Proportion of Emails")
          plt.xlabel("Words");
          train.head(10)
```

```
Out[321]:      id                                            subject  \
          0  7657             Subject: Patch to enable/disable log\n
          1  6911        Subject: When an engineer flaps his wings\n
          2  6074  Subject: Re: [Razor-users] razor plugins for m…
          3  4376  Subject: NYTimes.com Article: Stop Those Press…
          4  5766  Subject: What's facing FBI's new CIO? (Tech Up…
          5  5247                                Subject: asap\n
          6  7410     Subject: [use Perl] Headlines for 2002-09-18\n
          7   576  Subject: Cost price Guinness, Budweiser and se…
          8  4398  Subject: Play by Play: Effective Memory Manage…
          9  3989  Subject: Sheila Lennon was interviewed for the…

                                                        email  spam    len  \
          0  while i was playing with the past issues, it a…     0   1641
          1  url: http://diveintomark.org/archives/2002/10/…     0   4713
          2  no, please post a link!\n \n fox\n ----- origi…     0   1399
          3  this article from nytimes.com \n has been sent…     0   4435
          4  <html>\n <head>\n <title>tech update today</ti…     0  32857
          5  --===_secatt_000_1fuklemuttfusq\n content-type…     1   1156
          6  use perl daily headline mailer\n \n subscribe …     0    631
          7  ------=_part_2067_1947928.1038245405702\n cont…     0  26233
          8  url: http://www.newsisfree.com/click/-0,861367…     0    513
          9  url: http://scriptingnews.userland.com/backiss…     0    215

             predictions  lenSubject
          0            0          37
          1            0          42
```

```
2               0            54
3               0            73
4               0            52
5               0            14
6               0            45
7               0            73
8               0            51
9               0            60
```

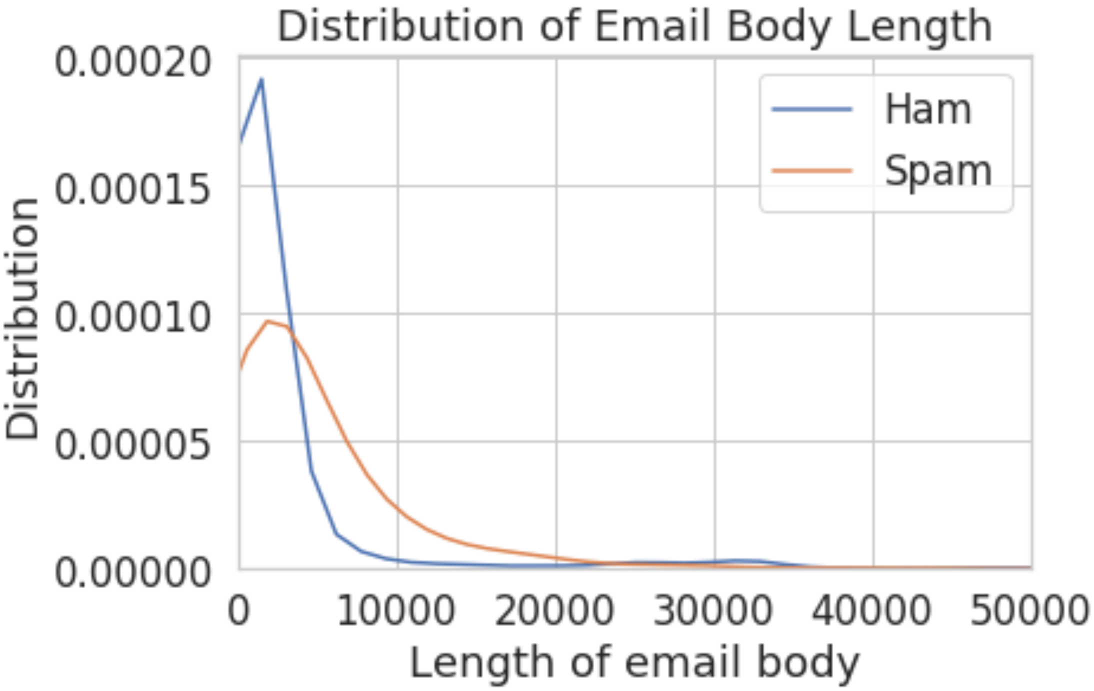


0.0.3 Question 3b



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 [124]: trainpl = train
          trainpl['len'] = [len(x) for x in trainpl['email']]
          trainpl
          sns.distplot(a = trainpl[trainpl["spam"] == 0]['len'], kde=True, hist = False)
          sns.distplot(a = trainpl[trainpl["spam"] == 1]['len'], kde=True, hist = False)
          plt.xlim((0, 50000))
          plt.xlabel('Length of email body')
          plt.ylabel('Distribution')
          plt.title("Distribution of Email Body Length")
          plt.legend(['Ham', "Spam"])
          plt.savefig('training_conditional_densities.png')
```


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

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)?

We observe these values to give us an idea of the shortcomings on our predictor, as if there was a common trait of mishaps. In our case (counting all as not spam and rather ham), we just get returned the proportion of ham to all values in our training set. Because of this, we really just reflected knowns about the data, while not being able to predict ANY of the correct spam emails.

0.0.5 Question 6e

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

There will definitly be a lot less false negatives when using logistic regression classifier over our Zero Classifier, due to the fact that we will ALWAYS predict negative if our value is positive, so that is the max amount of false negatives we can predict. The false positives will be more for sure. With our Zero Classifier, there is NO WAY we can predict spam, even by mistake, if we classify them all as ham. Our logistic regression classifier can predict better than this, so we expect that to be the lowest it will ever predict, and more than likely pick way beyond 0.

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. It predicted that it was in fact about 74% for our zero classifier, but the reason for this is that our test set is about 74% ham emails and 26% spam, so it will reflect that if we claim them all to be one or the other. The logisitic classifier was close, but this just means that the values we based it off were not very accurate and we were probably underfitting our data.

2. We have the words 'drug', 'bank', 'prescription', 'memo', and 'private' as our predictors. These words, for the most part don't much reflect what we would see from spam mail. In fact, if we worked at lets say a doctors office, we would have to mention the words drug, prescription, and private, but bank and memo might not make it in very well, and may reflect a spam mail, but a very niche area of spam mail, so it doesnt encompass it all. Better words that are more polarizing when it comes to classification (such as HTML or body), would make our data a lot more accurate.

3. Personally, although both are poor, I would choose the Logistic Classifier. I would definitely not choose the Zero Classifier since I would recieve all emails either way, and therefore would not be an accurate spam filter. It will bring the true positives up, and avoiding seeing emails in total. Definitely would prefer to change the data it is being filtered by however to include some more "spot on" examples (ie: body, html, money).

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. For my model, I looked at different emails (thus why theres a bunch of .head(n) with various numbers). I also realized that most had HTML attributes to them, so if I took common HTML words (such as body, div, href… etc), then I could pinpoint some as well. I noticed as well they tended to have a lot of symbols not normally used in conversations, such as ##, %%, $$, ==, and –. I found this out by just reading the emails and testing around with my
2. I thought that typical words, such as free, buy, and others arent strong enough words to consitute a spam mail. In fact, it was more likely that trigger words like that would be mispelled, but things such as html and symbols were fairly common and almost impossible to misspell, so those remained pretty reliable.
3. Honestly, me being more reliant on symbols and HTML attributes was pretty surprising. It didnt occur to me that this would be such a hit among the training set in accurately deciding which was which.

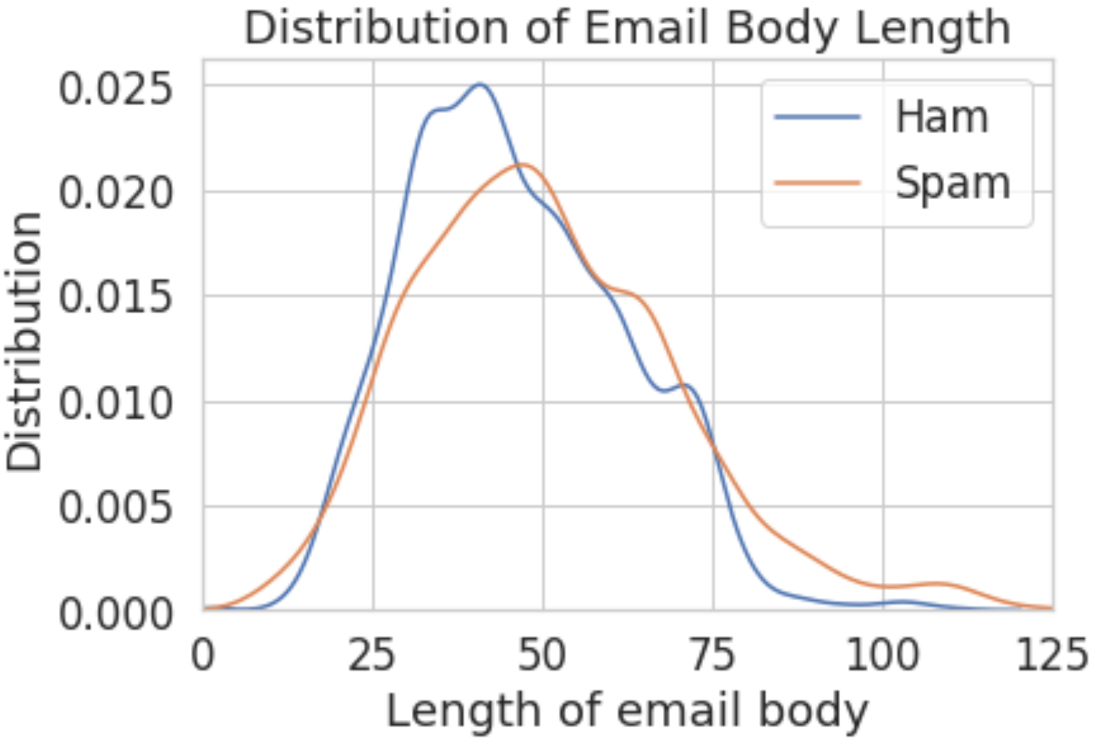
Generate your visualization in the cell below and provide your description in a comment.

In [218]:

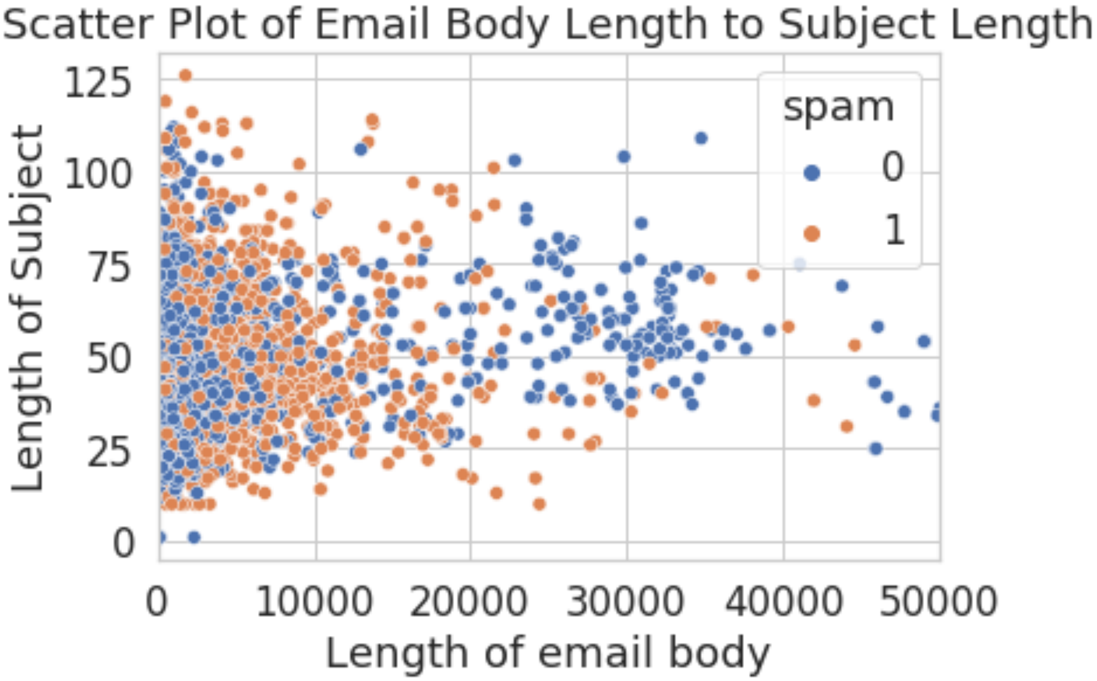
```
# Now I made two, since I knew there was not allowed to copy the same thing we did from probl
# reflect the subject to see if there was anything to note there as well. The second visualiz
# to see if there was any correlation between the two. From the first visualazation, the dist
# and ham is fairly the same, which indicates to me that there is no real differentiation bet
# subjects between the two. When I made the scatterplot distribution, it confirmed to me that
# correlation between them both as well, as the values have no clear trend, and are rather al
# of this data, I decided that in terms of length, subject is not an indicator of spam or not

# Write the code to generate your visualization here:
train['lenSubject'] = [len(x) for x in train['subject']]
sns.distplot(a = trainpl[trainpl["spam"] == 0]['lenSubject'], kde=True, hist = False)
sns.distplot(a = trainpl[trainpl["spam"] == 1]['lenSubject'], kde=True, hist = False)
plt.xlim((0, 125))
plt.xlabel('Length of email body')
plt.ylabel('Distribution')
plt.title("Distribution of Email Body Length")
plt.legend(['Ham', "Spam"]);
```

Out[218]:

```
<matplotlib.legend.Legend at 0x7f3c0e93aa90>
```



```
In [219]: sns.scatterplot(x = train['len'], y = train['lenSubject'], hue = train['spam'])
          plt.xlim((0, 50000))
          plt.xlabel('Length of email body')
          plt.ylabel('Length of Subject')
          plt.title("Scatter Plot of Email Body Length to Subject Length");
```



0.0.8 Question 9: ROC Curve

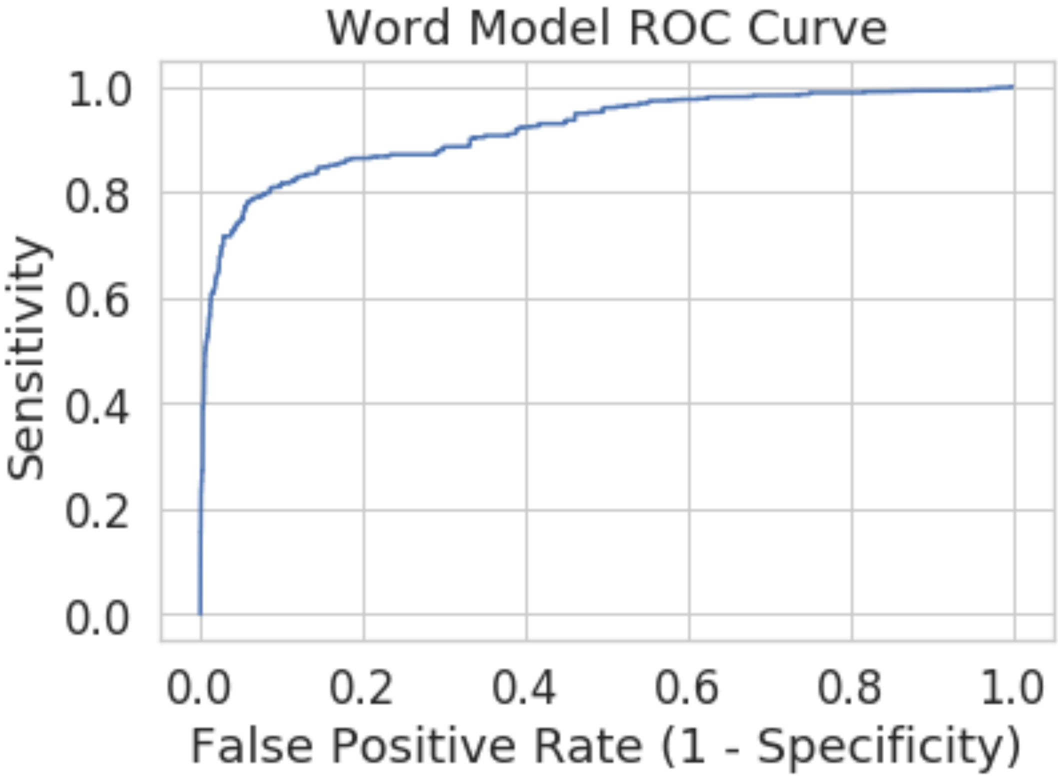
In most cases we won't be able to get 0 false positives and 0 false negatives, so we have to compromise. For example, in the case of cancer screenings, false negatives are comparatively worse than false positives — a false negative means that a patient might not discover that they have cancer until it's 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 [420]: 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
          my_words = ['html', 'body', 'dear', 'font', 'align', 'href', 'style', 'title', 'br', '!!!', '

          my_X_train = words_in_texts(my_words, train['email'])
          my_Y_train = train['spam']

          my_model = LogisticRegression().fit(my_X_train, my_Y_train)
          tot1 = my_model.predict(my_X_train)
          #train['my predictions'] = tot1

          my_training_accuracy = accuracy_score(train['spam'], tot1)
          print("Training Accuracy: ", my_training_accuracy)


          #source-- Section 17.7 of our textbook--
          probabilities = my_model.predict_proba(my_X_train)[:, 1]
          fp_rate, sensitivity, thresholds = roc_curve(Y_train, probabilities, pos_label=1)
          plt.step(fp_rate, sensitivity)
          plt.xlabel('False Positive Rate (1 - Specificity)')
          plt.ylabel('Sensitivity')
          plt.title('Word Model ROC Curve');
```

```
Training Accuracy:  0.9057633435378677
```


Word Model ROC Curve
Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (1 - Specificity)