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.

It would appear that the spam emails would have some includes some weird HTML formating, which ham does not seem to have.

0.0.2 Question 3a

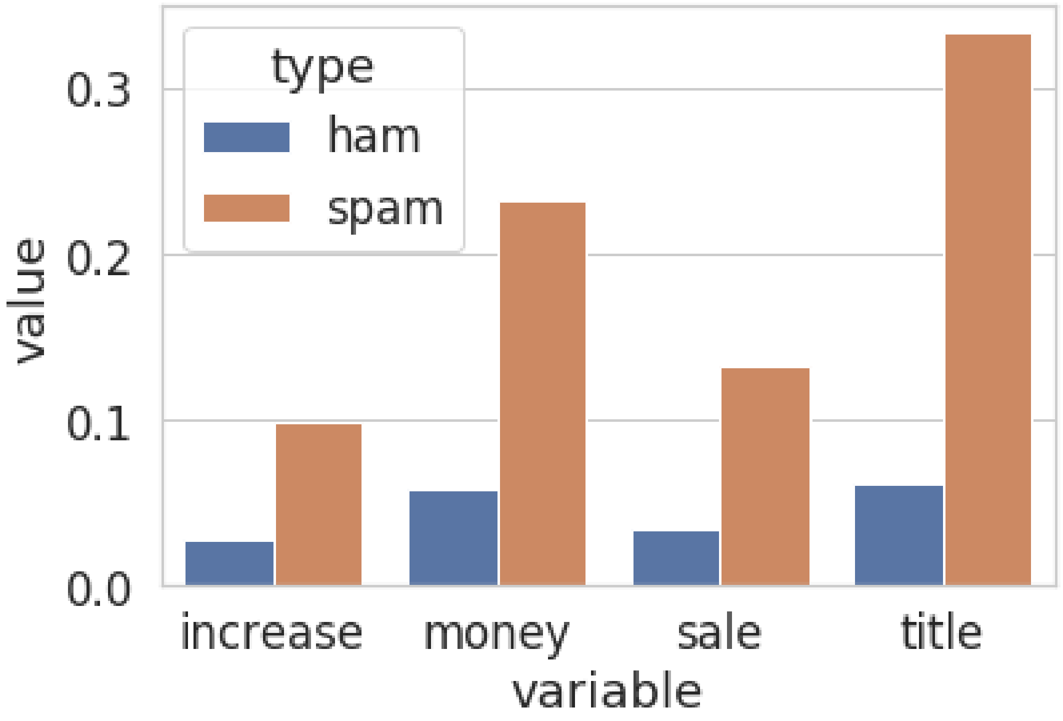
Create a bar chart like the one above comparing the proportion of spam and ham emails containing certain words. Choose a set of words that are different from the ones above, but also have different proportions for the two classes. Make sure to only consider emails from `train`.

```
In [16]: train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai

         #Getting the training emails and words.
         email_train = train['email']
         words = ['title', 'sale', 'increase','money']
         #Seeing what words are in the emails.
         zeros_ones = words_in_texts(words, email_train).astype(int)

         #Separating out the first word and second word for the columsn
         first_word = [item[0] for item in zeros_ones]
         second_word = [item[1] for item in zeros_ones]
         third_word = [item[2] for item in zeros_ones]
         fourth_word = [item[3] for item in zeros_ones]

         words = [first_word, second_word, third_word, fourth_word]
         #Looking at which of the emails are spam.
         spam_train = train['spam']
         spam_ham_frame = pd.DataFrame({
             'title' : first_word,
             'sale' : second_word,
             'increase' : third_word,
             'money' : fourth_word,
             'type' :  spam_train
         })
         spam_ham_frame['type'].replace({0: 'ham', 1: 'spam'}, inplace = True)
         #Creating a table seeing which email has the first and second word and if it's type.
         #sns.barplot(x = 'variable', y = 'value', hue = 'type', data = spam_ham_frame.melt('type').gro
         df = spam_ham_frame.melt('type').groupby(['type', 'variable']).mean().reset_index()
         df
```

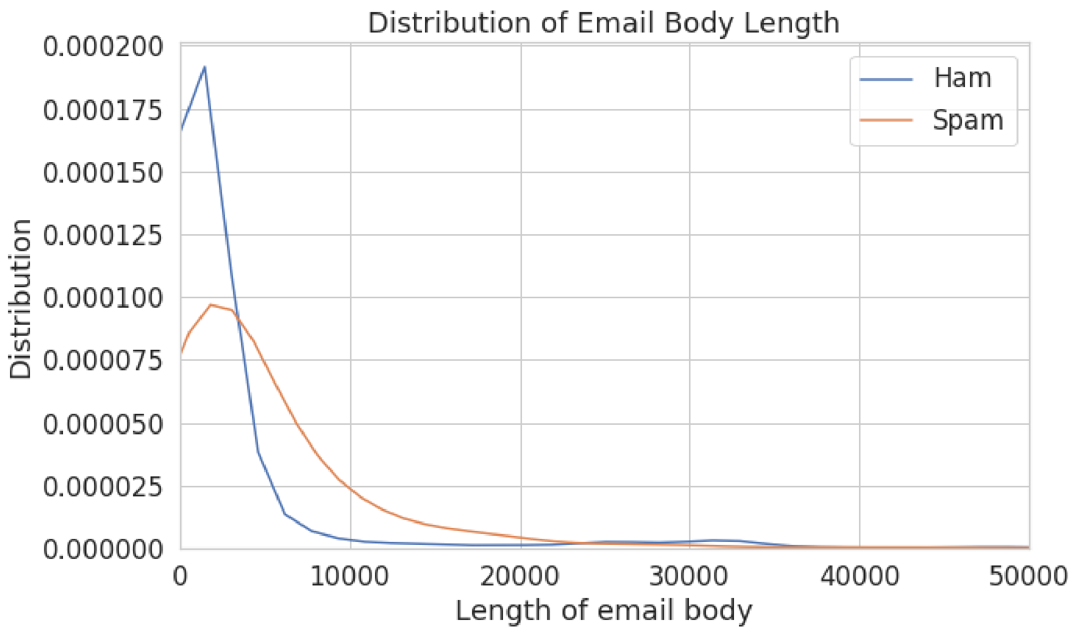
```
Out[16]:    type  variable     value
         0   ham  increase  0.027346
         1   ham     money  0.058266
         2   ham      sale  0.033780
         3   ham     title  0.061662
         4  spam  increase  0.098019
         5  spam     money  0.232534
         6  spam      sale  0.132430
         7  spam     title  0.333681
```

```
In [17]: sns.barplot(x = 'variable', y = 'value', data = df, hue = 'type')
```

```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7f73a5d94340>
```


type
ham
spam
value
0.0
0.1
0.2
0.3
increase
money
sale
title
variable

0.0.3 Question 3b



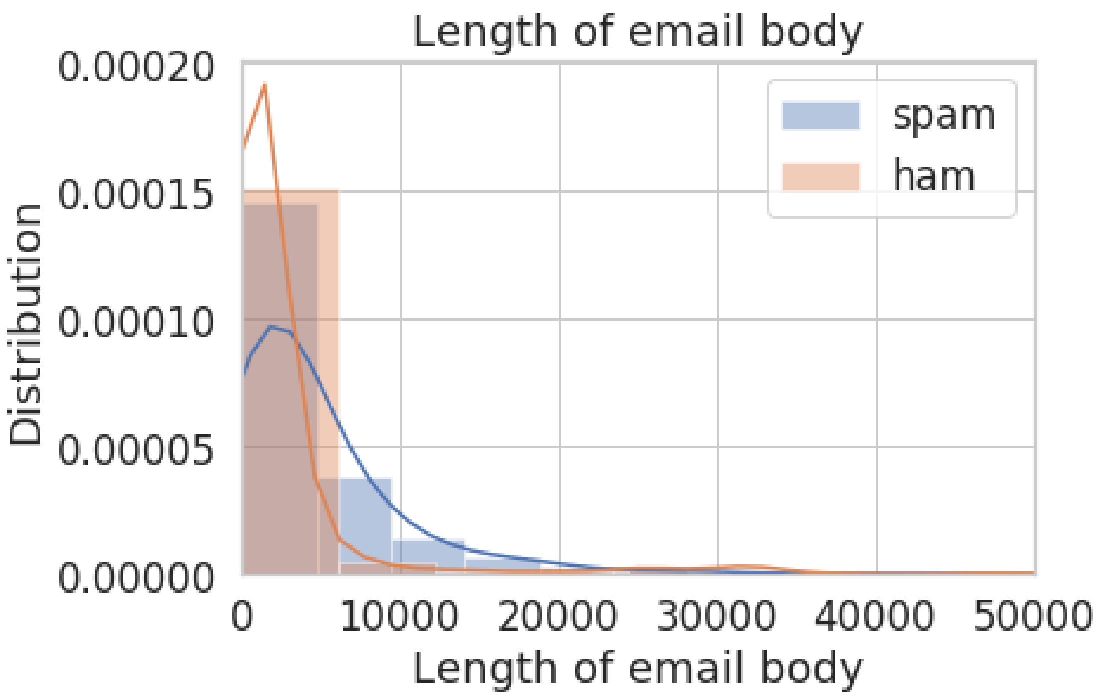
Create a *class conditional density plot* like the one above (using `sns.distplot`), comparing the distribution of the length of spam emails to the distribution of the length of ham emails in the training set. Set the x-axis limit from 0 to 50000.

```
In [18]: train[train['spam']==1].email
```

```
Out[18]: 5       --===_secatt_000_1fuklemuttfusq\n content-type…
         13      ####################################\n \n   fre…
         15      <table width="600" border="20" align="center" …
         18      <html>\n <head>\n </head>\n <body>\n <center>\…
         19      <html>\n <head>\n </head>\n <center>\n <h1>\n …
                                           …
         7502    <html>\n <body bgcolor=3d"#003300">\n <p align…
         7503    below is the result of your feedback form.  it…
         7505    <html>\n <table width="350" border="0" cellspa…
         7507    \n mr. ayanda maredi\n department of minerals …
         7509    \n dear consumers, increase your business sale…
         Name: email, Length: 1918, dtype: object
```

```
In [19]: spam_email_len = [len(i) for i in train[train['spam']==1].email]
         ham_email_len = [len(i) for i in train[train['spam']!=1].email]
         sns.distplot(x = spam_email_len, label = 'spam')
         sns.distplot(x = ham_email_len, label = 'ham')
```

```
plt.xlim(0,50000)
plt.title("Length of email body")
plt.xlabel("Length of email body")
plt.ylabel("Distribution")
plt.legend()
plt.savefig('training_conditional_densities.png')
```



```
In [20]: #...
         #plt.savefig('training_conditional_densities.png')
```

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)?

Because we predicted that all emails would be ham. This would mean that we 0 False postives because it would not be possible to get any predicted spam. However, because we predicted only ham, it would raise the amount of false negatives. It would appear accuracy is around 74 % meaning that 74% of the time it is indeed ham emails. For recall, it is 0 % because this model only shows ham emails.

0.0.5 Question 6e

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

Number 5: fp : 0 fn : 1918
All Zeros: fp : 122 fn : 1699

We can see that there are less false positives when we use the logistic regression in q5.

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 would appear that predicting 0 for every email is only around 0.7447091707706642, which is approximately one percent lower than our logistic regression classifier.
2) Our words are 'drug', 'bank', 'prescription', 'memo', 'private', which are words I barely see in my own emails. Thus, a reason that this classifier may be performing poorly is because these words appear rarely making it hard to use as indicators for ham/spam. 3)I would use the logistic regression classifier. Even though using all 0s give a similar accuracy, but I think it is just a coincidence that they have such similar accuracy of around 75 percent.

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) I followed your instructions on trying to use the lengths of the words, the number of punctuations such as ! and ?, and tried different words from spam.

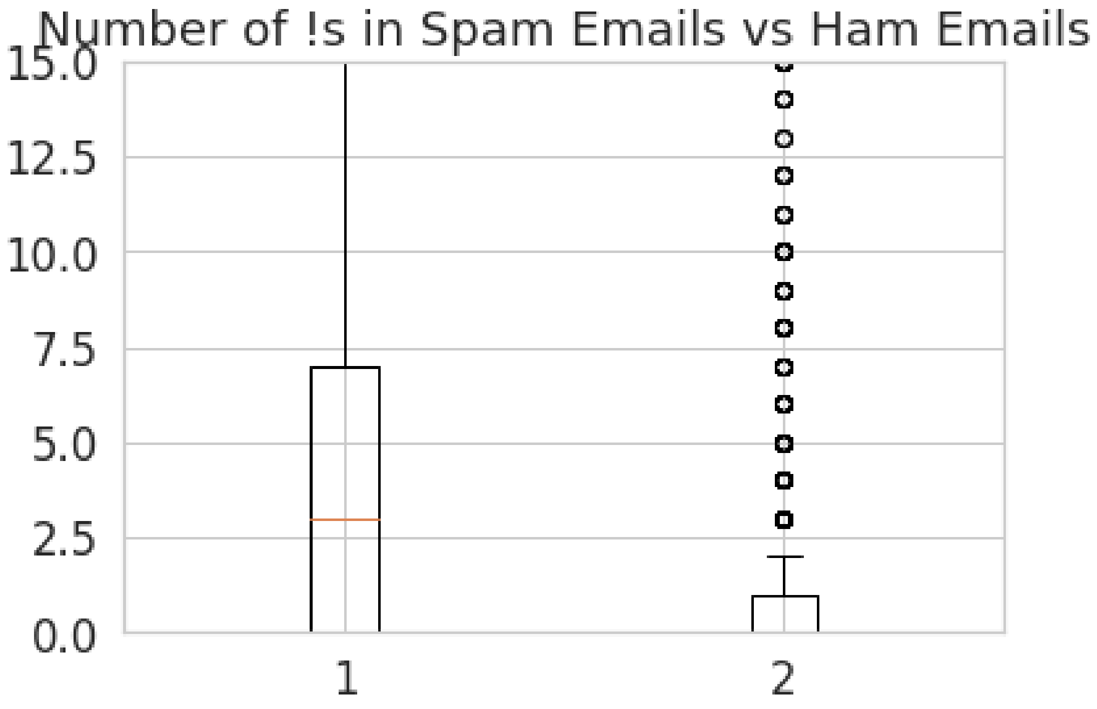
2) One thing that really worked for me was finding common words in spam that were not used in spam. Also getting rid of stop words and html tags helped a lot.

3) I was really surprised that we could use many "count"s of the "spam" words to create so many features. I felt that I could use more words, but they were probably going to have low frequency in spam and ham emails that it probably did not matter.

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

```
In [42]: # Write your description (2-3 sentences) as a comment here:
         # I figured that spam emails will usually want to capture their audience and that is usually d
         # punctuation (mainly exclamation marks). Thus, I set out to figure out how big of a differenc
         # exclamation marks in spam and non spam emails. And as you can tell from the boxplot on the r
         # ! marks, but spam emails tend to use on average around 3.

         # Write the code to generate your visualization here:

         x=train[train.spam == 1]['num_of_!s']
         y=train[train.spam != 1]['num_of_!s']
         plt.boxplot ([x,y])

         plt.ylim(0,15)
         plt.title("Number of !s in Spam Emails vs Ham Emails")
```

```
Out[42]: Text(0.5, 1.0, 'Number of !s in Spam Emails vs Ham Emails')
```



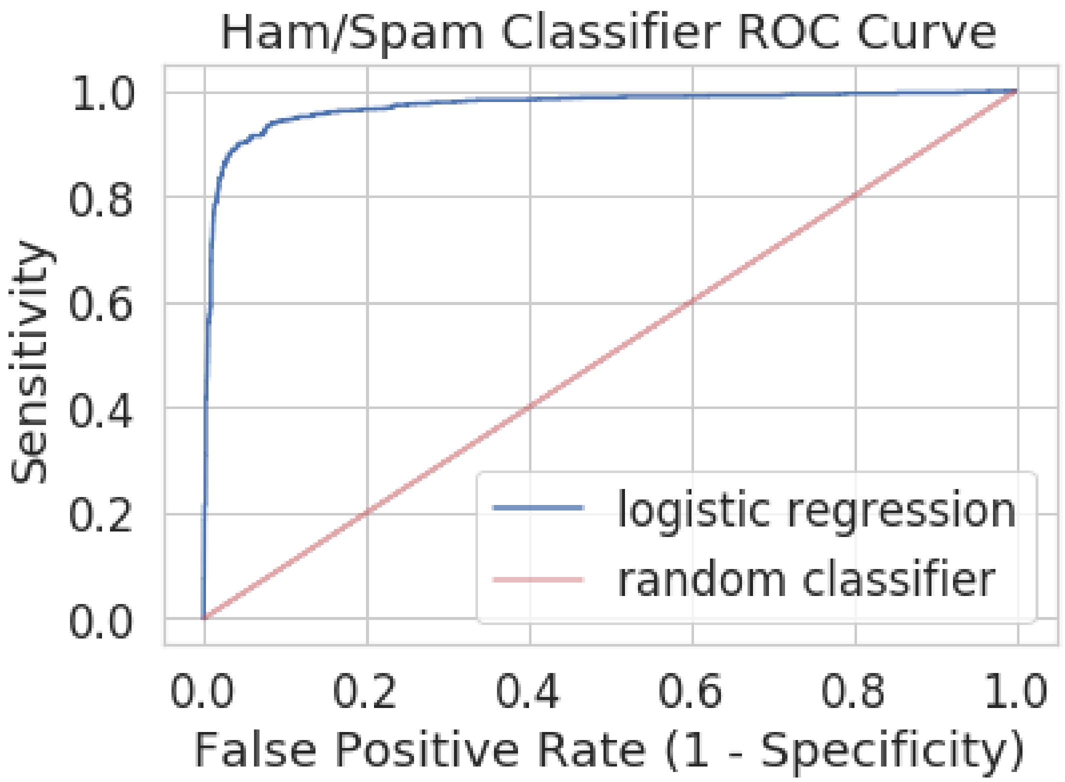
0.0.8 Question 9: ROC Curve

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 [43]: 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

         model_probs = model2.predict_proba(X_train)[:, 1]
         false_positive_rate_values, sensitivity_values, thresholds = roc_curve(y_train, model_probs, po

         plt.step(false_positive_rate_values, sensitivity_values, color='b',
                  where='post', label = 'logistic regression')
         plt.xlabel('False Positive Rate (1 - Specificity)')
         plt.ylabel('Sensitivity')
         plt.title('Ham/Spam Classifier ROC Curve')
         plt.step(np.arange(0, 1, 0.001), np.arange(0, 1, 0.001), color='r', alpha=0.5,
                  where='post', label = 'random classifier')
         plt.legend();
```


Ham/Spam Classifier ROC Curve
Sensitivity
False Positive Rate (1 - Specificity)
0.0
0.2
0.4
0.6
0.8
1.0
logistic regression
random classifier