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.

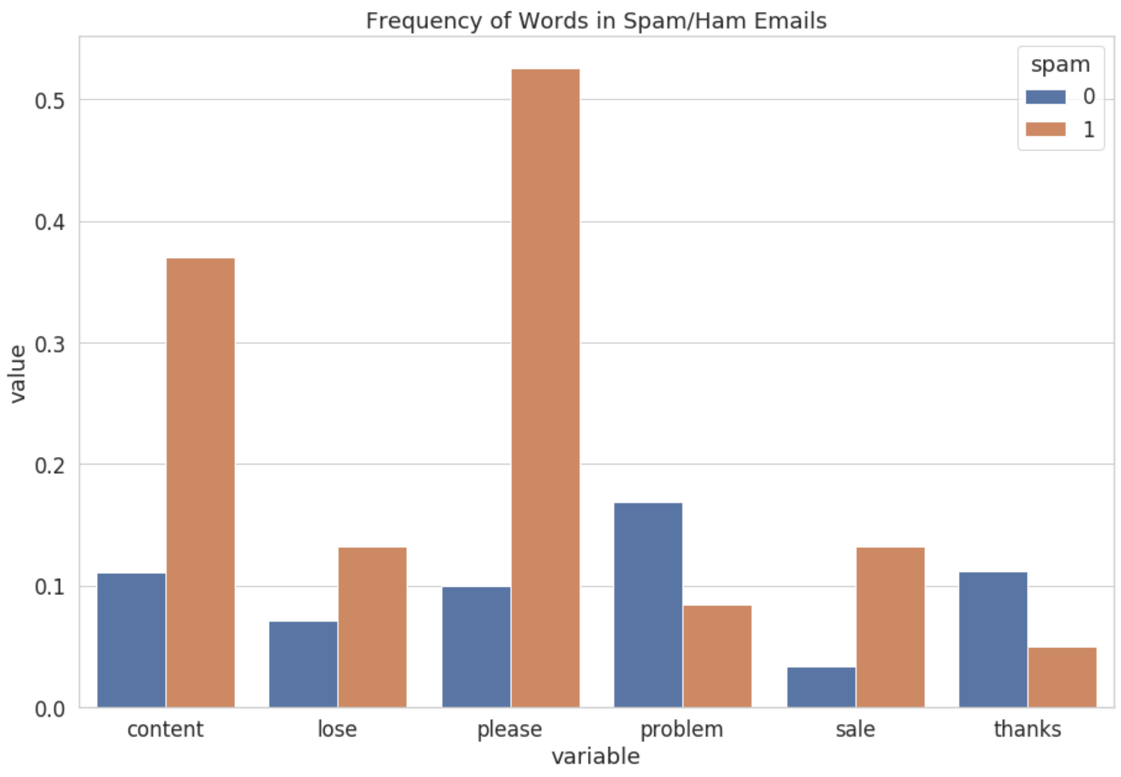
The ham email uses regular text, while the spam uses html body text for the email.

0.0.2 Question 3a

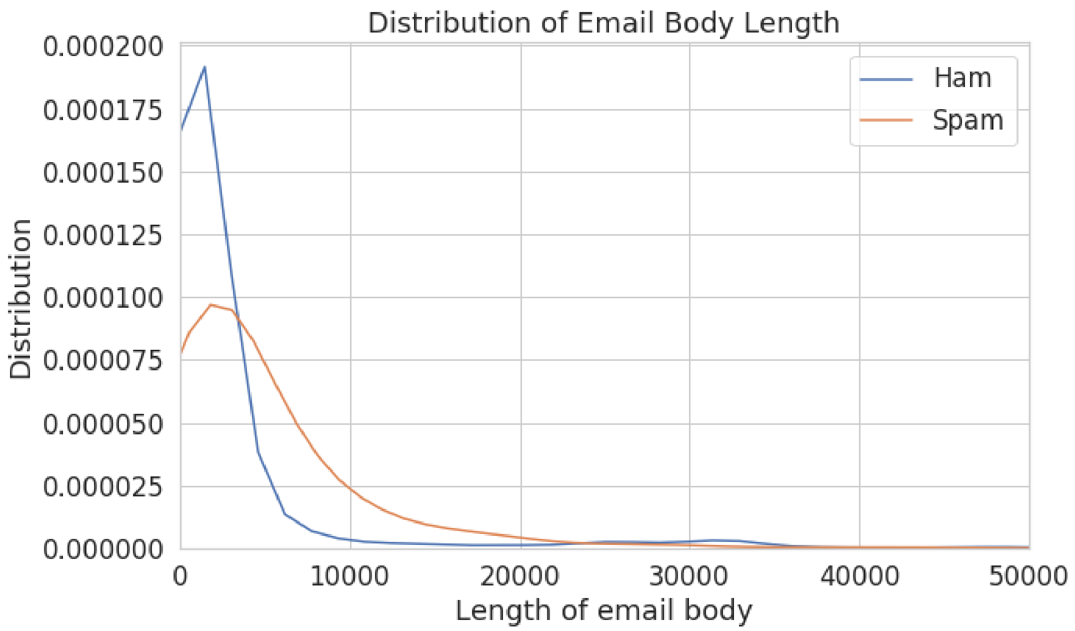
Create a bar chart like the one above comparing the proportion of spam and ham emails containing certain words. Choose a set of words that are different from the ones above, but also have different proportions for the two classes. Make sure to only consider emails from `train`.

```
In [44]: train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai
         words = ['problem', 'sale', 'thanks', 'please', 'lose',"content"]
         indicators_w = pd.DataFrame(words_in_texts(words, train['email']),columns= words)
         indicators_w['spam'] = train['spam']
         indicators_w= indicators_w.melt('spam')
         indicators_w = indicators_w.groupby(['spam','variable']).agg(np.mean).reset_index()
         plt.figure(figsize = (15,10))
         sns.barplot(data= indicators_w,x = 'variable', y='value',hue = 'spam')
         plt.title("Frequency of Words in Spam/Ham Emails")
```

```
Out[44]: Text(0.5, 1.0, 'Frequency of Words in Spam/Ham Emails')
```

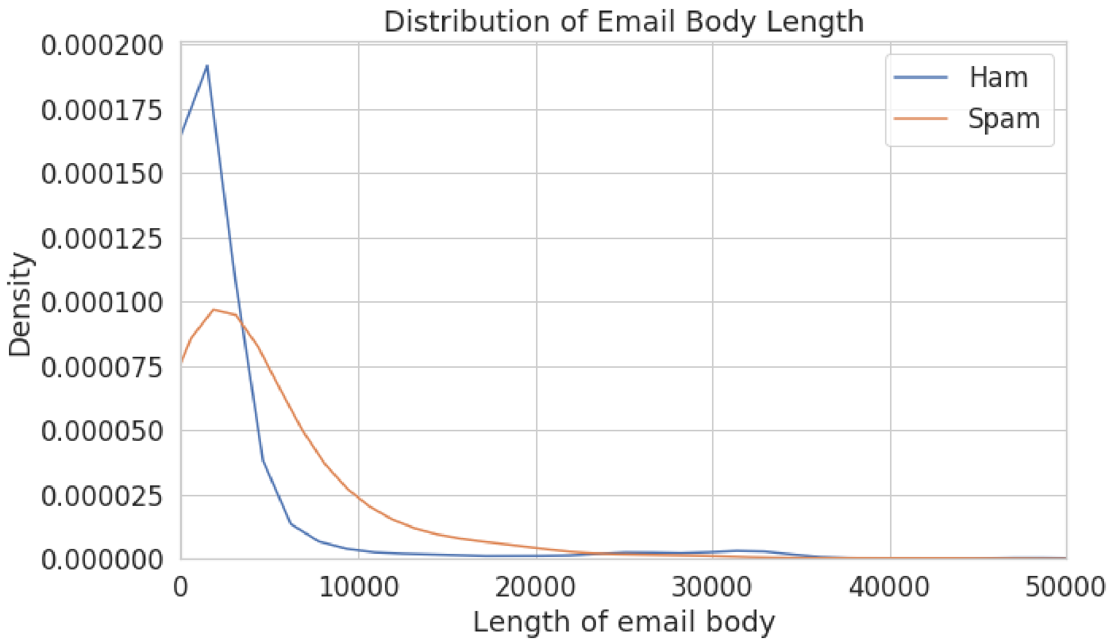


0.0.3 Question 3b



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]: plt.figure(figsize = (10,6))
         sns.distplot(train[train["spam"]==0]['email'].str.len(),hist = False, label = 'Ham')
         sns.distplot(train[train["spam"]==1]['email'].str.len(),hist = False, label = 'Spam')
         plt.xlabel('Length of email body')
         plt.xlim(0,50000)
         plt.title("Distribution of Email Body Length")
         plt.legend()
         plt.savefig('training_conditional_densities.png')
```


Distribution of Email Body Length
Density
Length of email body
Ham
Spam
0.000200
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
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)?

1. We have 0 false postives because false postives are values that are predicted positive, but actually negative. Since our model never predicts values as positive, there are no false postives.
2. We have 1918 false negatives because a false negative is any value that is postive, but is predicted as negative. Since the models predicts only negative values, so all the positive values will be false negatives. Since there are 1918 postive values, there are 1918 false negatives.
3. The accuracy is about 0.745 or 74.5%. This is the number ham emails that are actually predicted as ham emails. There are 5595 ham emails that are predicted as ham and a total of 7513 emails in the training set. 5595/7513 esitimated to 0.745
4. True positive values are values that are spam or positive and actually predicted as positive. Since no values are predicted as positive, there are no true postives, so recall 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 122 false positives and 1699 false negatives. There are more false negatives than false postives.

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. The logistic model is a better classifier than predicting 0, but the logistic model is still a pretty bad model. It's accuracy of 75.76% is pretty close to the zero predicter accuracy of 74.5 %.
2. This classifier is poor because some of the words like 'drug' , 'memo' and 'presciption' don't show up in a lot emails, since they aren't that commomnly used words.
3. I prefer the logistic model because it has a better accuracy of 75.76% and a better recall of 11.4%. This means the proportion of emails that are spam that are classified of spa

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. How did you find better features for your model?

I first combined the subject with the email, so I can also use the words in the subject aand the words in the email as possible features. Then I got the spam emails from the training dataframe and then counted how many times each word shows up in all of the emails. I created a spam dictionary where I mapped the word to the number of times in showed up in the email category. Then I sorted these dictionaries by count, so I had the words with highest counts in the beginning. This way I could use the words that were the most frequent as my features. Then I cross validated using five folds the set with a different number of features each time to find the best number of features to use. I looked at a range of 100-1500 in intervals of a 100. I found out I was getting the best accuracy between 1100 and 1300 features. I ran the cross validating process again to narrow the interval of the number of features I needed. Since I used intervals of 100, previously I wanted to see if I could capture better accuracy somewhere in between the 100 jump. I ran cross validation on a smaller interval with more frequent jumps. I used the interval of 1100-1300 in intervals of 10. I plotted this and I found out that using the top 1160 most used words in our spam emails as features for my model was the best number of features to use.

2. What did you try that worked or didn't work?

I intially just googled the nost used words in spam emails and started adding these words as my features. I kept on adding more words until I reached the 88% accuracy. This worked, however it was rather tedious becuase I would some words and then check my accuracy over and ove again. This also wasn't giving me that high accuracy as I only reached about 89%. After that, it was hard to increase the accuracy.I tried adding about 15 more words and my accuracy was still at 89%, so I decided to try another way to make my model.

3. What was surprising in your search for good features?

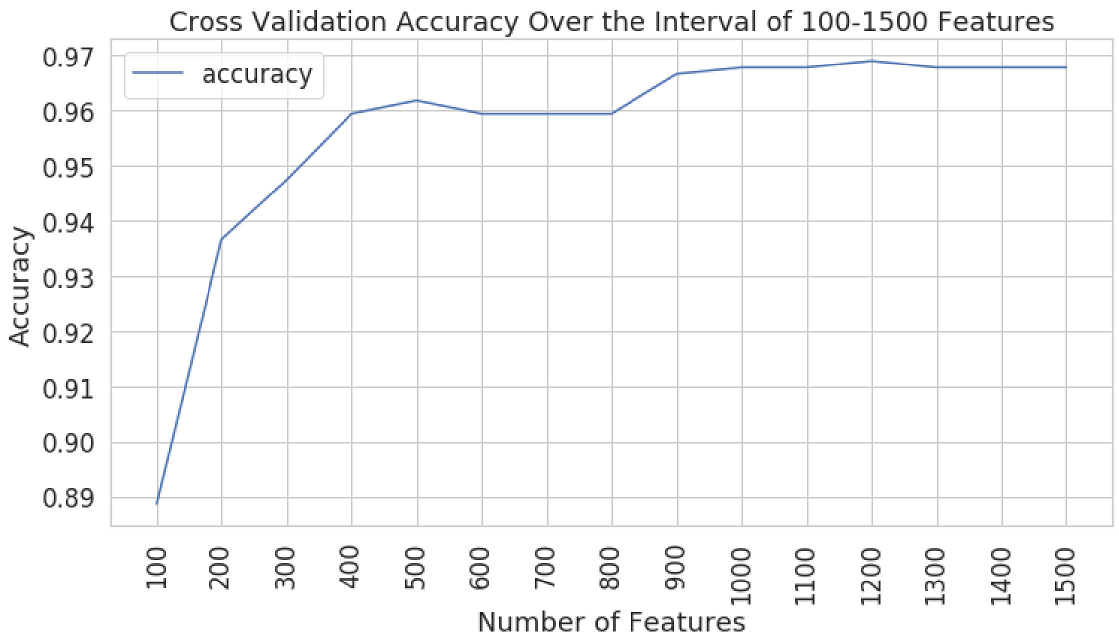
A lot of the features are actually not real words like 'ffffff' or "aaaaaaaaaaaaaaaa". A lot of the good features were also part of the html style like "font" or "border". I expected more real words, rather than made up words or html language. I intially took out words that occured in both ham and spam emails. I thought this would increase my accuracy, beacuse removing words that show up in both would give me distinct features that easily distinguish an email from ham and spam. However, this actually reduced my accuracy slightly.

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

```
In [35]: # Write your description (2-3 sentences) as a comment here:
         #This plot shows us the best number of features to use from a range of 100-1500 with a jump
         #of 100 and with each number of features, a five fold cross validation is run and then the mea
         #accuracy is plotted.As the number of features increase, the accuracy rate increases in genera
         #At 600-800 features,the accuracy rate stays constant and then after that it increases sharply
         #from 800-900 and slowly increases from 900-1200, and then decreases after.At around 1200
         #features the accuracy rate is the highest.

         # Write the code to generate your visualization here:
         w_count = np.arange(100,1600,100)
         acc_df = pd.DataFrame(arr,index= w_count, columns = ["accuracy"])
         plt.figure(figsize=(12,6))
         sns.lineplot(data = acc_df)
         plt.xlabel('Number of Features')
         plt.ylabel('Accuracy')
         plt.title('Cross Validation Accuracy Over the Interval of 100-1500 Features')
         plt.xticks(np.arange(100,1600,100), rotation = 90)
```

```
Out[35]: ([<matplotlib.axis.XTick at 0x7fdaf3eede50>,
           <matplotlib.axis.XTick at 0x7fdaf3eede20>,
           <matplotlib.axis.XTick at 0x7fdaf3f021f0>,
           <matplotlib.axis.XTick at 0x7fdae5a57130>,
           <matplotlib.axis.XTick at 0x7fdaf3ed9610>,
           <matplotlib.axis.XTick at 0x7fdae5a57760>,
           <matplotlib.axis.XTick at 0x7fdae5a57c70>,
           <matplotlib.axis.XTick at 0x7fdae5a611c0>,
           <matplotlib.axis.XTick at 0x7fdae5a616d0>,
           <matplotlib.axis.XTick at 0x7fdaf3eed850>,
           <matplotlib.axis.XTick at 0x7fdae5a73a60>,
           <matplotlib.axis.XTick at 0x7fdae5a73f70>,
           <matplotlib.axis.XTick at 0x7fdae5a7d4c0>,
           <matplotlib.axis.XTick at 0x7fdae5a6d850>,
           <matplotlib.axis.XTick at 0x7fdae5a612e0>],
          <a list of 15 Text xticklabel objects>)
```


Cross Validation Accuracy Over the Interval of 100-1500 Features
accuracy
Accuracy
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.90
0.89
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
Number of Features

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.

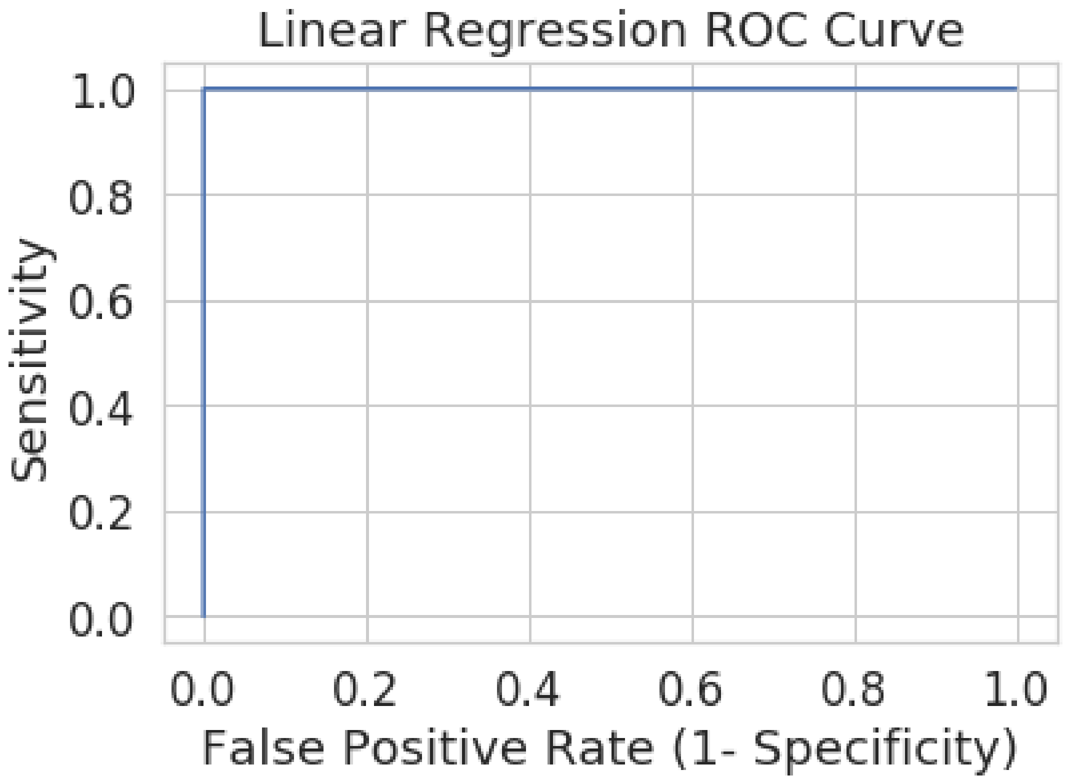
Recall that logistic regression calculates the probability that an example belongs to a certain class. Then, to classify an example we say that an email is spam if our classifier gives it ≥ 0.5 probability of being spam. However, *we can adjust that cutoff*: we can say that an email is spam only if our 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 [36]: 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, threshold = roc_curve(y_train, model_real.predict_proba(x_train)[:, 1], pos_label=1)
```

```
In [37]: plt.plot(fpr, tpr)
         plt.xlabel('False Positive Rate (1- Specificity)')
         plt.ylabel('Sensitivity')
         plt.title("Linear Regression ROC Curve")
```

```
Out[37]: Text(0.5, 1.0, 'Linear Regression ROC Curve')
```


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