Notebook

November 22, 2019

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.

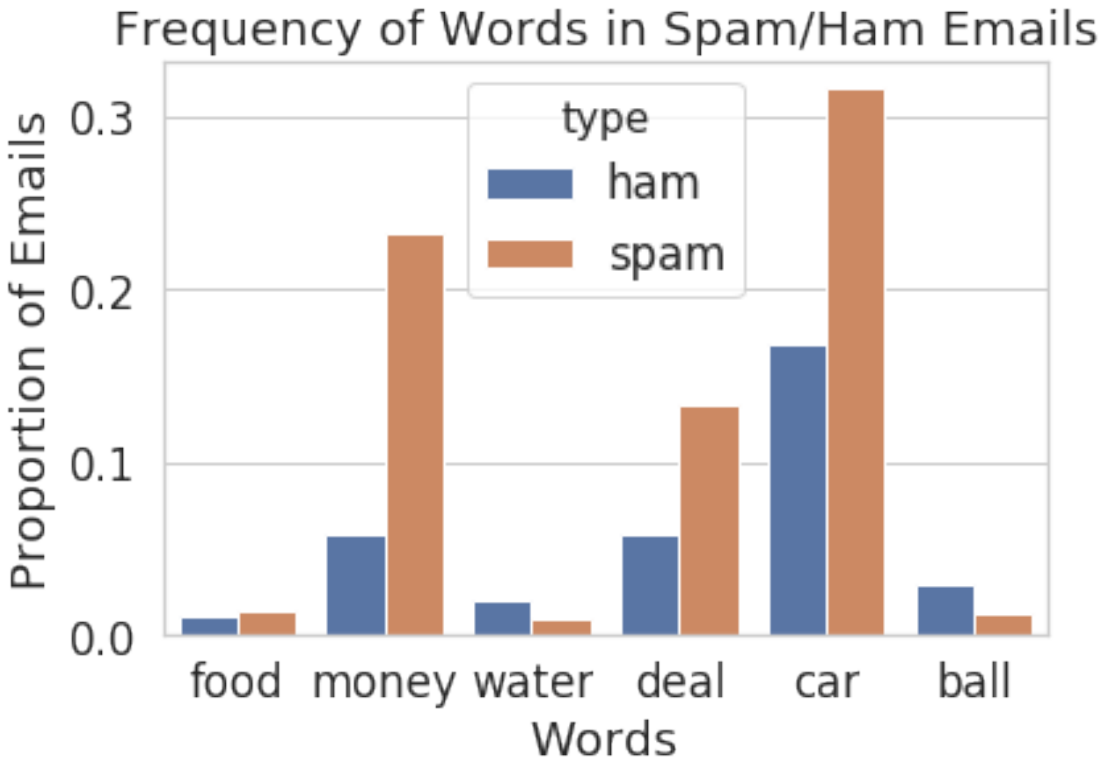
Comparing the two emails, I see that the ham email includes urls and text that is well organized. Looking at the spam email, the text is not well organized, and it includes html, body, font, and others.

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]:
```
import seaborn as sns

train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai

words = ['food', 'money', 'water', 'deal', 'car', 'ball']
words_text = words_in_texts(words, train['email'])
matrix = np.matrix(words_text)
df = pd.DataFrame(matrix).rename(columns={0:words[0],1:words[1],2:words[2],3:words[3],4:words[
df['type']=train['spam']
df=df.melt('type')
df['type']=df['type'].map({0:'ham',1:'spam'})
barplot = sns.barplot(x = 'variable', y = 'value', hue='type',data=df,ci=None)
barplot.set(xlabel = "Words", ylabel = "Proportion of Emails", title = "Frequency of Words in S
```

Out[13]:
```
[Text(0, 0.5, 'Proportion of Emails'),
 Text(0.5, 0, 'Words'),
 Text(0.5, 1.0, 'Frequency of Words in Spam/Ham Emails')]
```

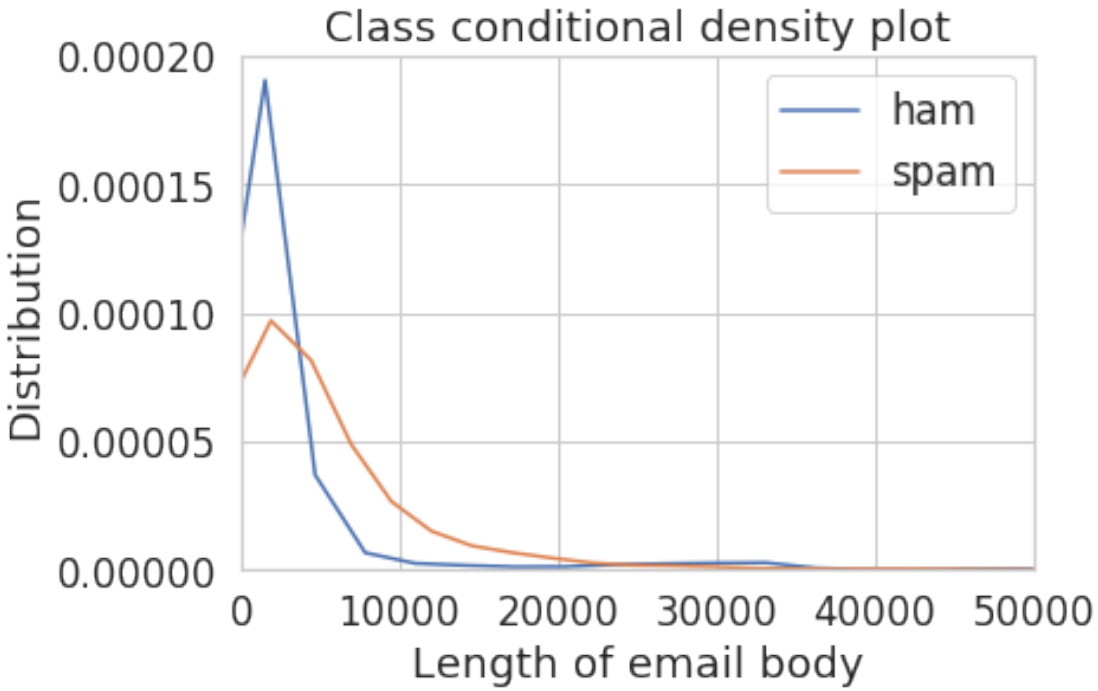


0.0.3 Question 3b

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]: length = train['email'].apply(len)
         df2 = pd.DataFrame({'Length':length,"Type":train['spam']})
         df2 = df2.melt('Type')
         df2['Type']=df2['Type'].map({0:'ham',1:'spam'})
         plt.xlim(0,50000)
         spam = df2[df2['Type']=='spam']
         ham = df2[df2['Type']=='ham']

         distplot1 = sns.distplot(ham['value'],label='ham',hist=False)
         distplot2 = sns.distplot(spam['value'],label='spam',hist=False)
         distplot1.set(xlabel = "Length of email body", ylabel = "Distribution", title = "Class conditi
```

```
Out[14]: [Text(0, 0.5, 'Distribution'),
          Text(0.5, 0, 'Length of email body'),
          Text(0.5, 1.0, 'Class conditional density plot')]
```



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)?

In 6a), we get 0 for zero_predictor_fp, because the predictor predicts everything is 0, so we don't have any false positives. Because it predicts everything as 0, zero_predictor_fn is when the spam = 1. For 6b), the accuracy is the ratio of how many observations = 0, compared to all the observations. zero_predictor_recall = 0, because we have no positives, so TP=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. Thus, there are more false negatives.

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.

1. Predicting 0 for every email, we get an accuracy of 74.47%. Thus, this accuracy is less than the accuracy of our logistic regression classifier. However, the accuracy values are very close because there are a lot of 0's in the data.
2. The logistic regression classifier is performing poorly because the words we are using to train our model could be found in both ham and spam emails.
3. I would prefer the logistic regression classifier for our spam filter because it received a higher accuracy value of 75.6%. Additionally, our logistic regression classifier got a false alarm rate of 2.18%, which I believe is very low.

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:

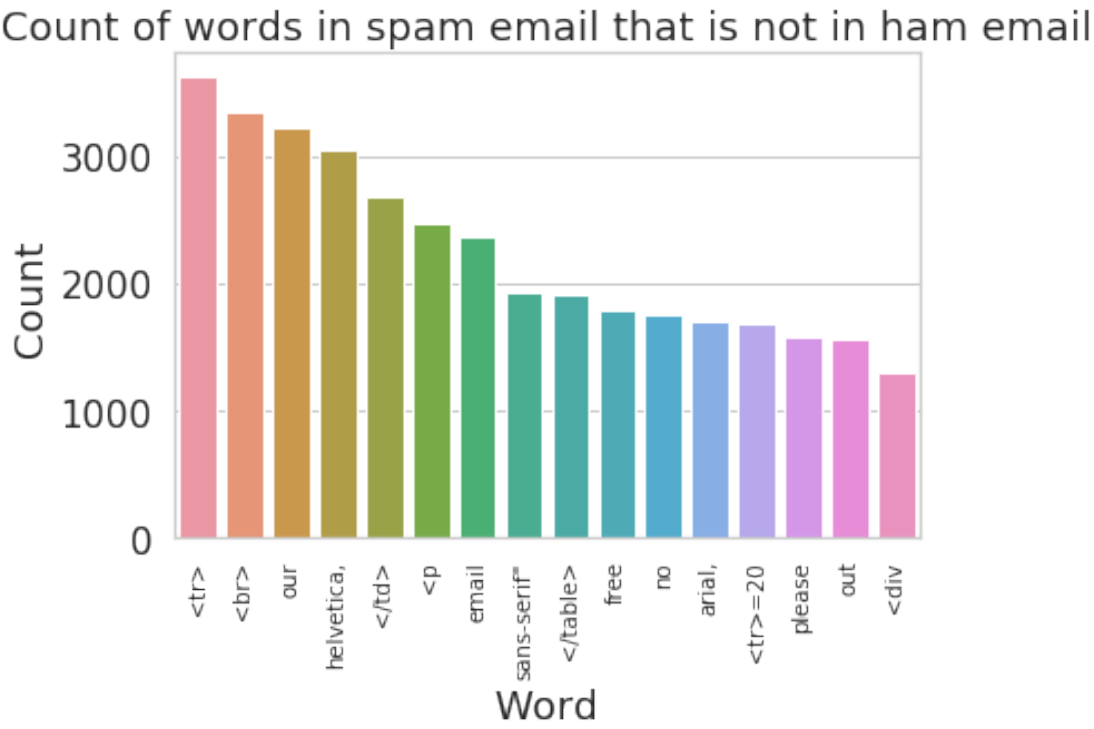
1. How did you find better features for your model?
2. What did you try that worked / didn't work?
3. What was surprising in your search for good features?

1. To find better features for my model, I used the code shown in the cell below to find the 60 most frequent words in emails that were classified as spam. Of course, the most frequent words were generic words like "the, to, and, of", so I chose to pick words that aren't as generic. Additionally, I looked at the most frequent words for emails that were classfied as ham, to make sure that the words I picked were not overlapping between the two. The specific words I found are shown in the cell below.
2. At first, what didn't work was I was looking for the 60 most frequent words that appeared in spam emails that didn't show up for the 60 most frequent words in the ham emails. For example one of those words were "free". However, "free" is still a very frequent word in the ham emails, I did not notice it because there are a lot more ham emails than there are spam emails. Thus, to handle this, I used the code "ham_email[ham_email.str.contains("free")]", and compared it to "spam_email[spam_email.str.contains("free")]" to see the frequency of words found in both ham and spam emails. This worked for me because, from repeating this step multiple of times for different words, I was able to see which words were frequent in spam emails that were not frequent in ham emails.
3. What was surprising for me was that a lot of the good features were words that had to do with fonts. For example, "helvetica, arial, sans-serif" were found predominantly in spam emails. What was also surprising was that I was expecting words like "free","coupon","discount" to be predominant in spam emails, but these words were found in both spam and ham emails.

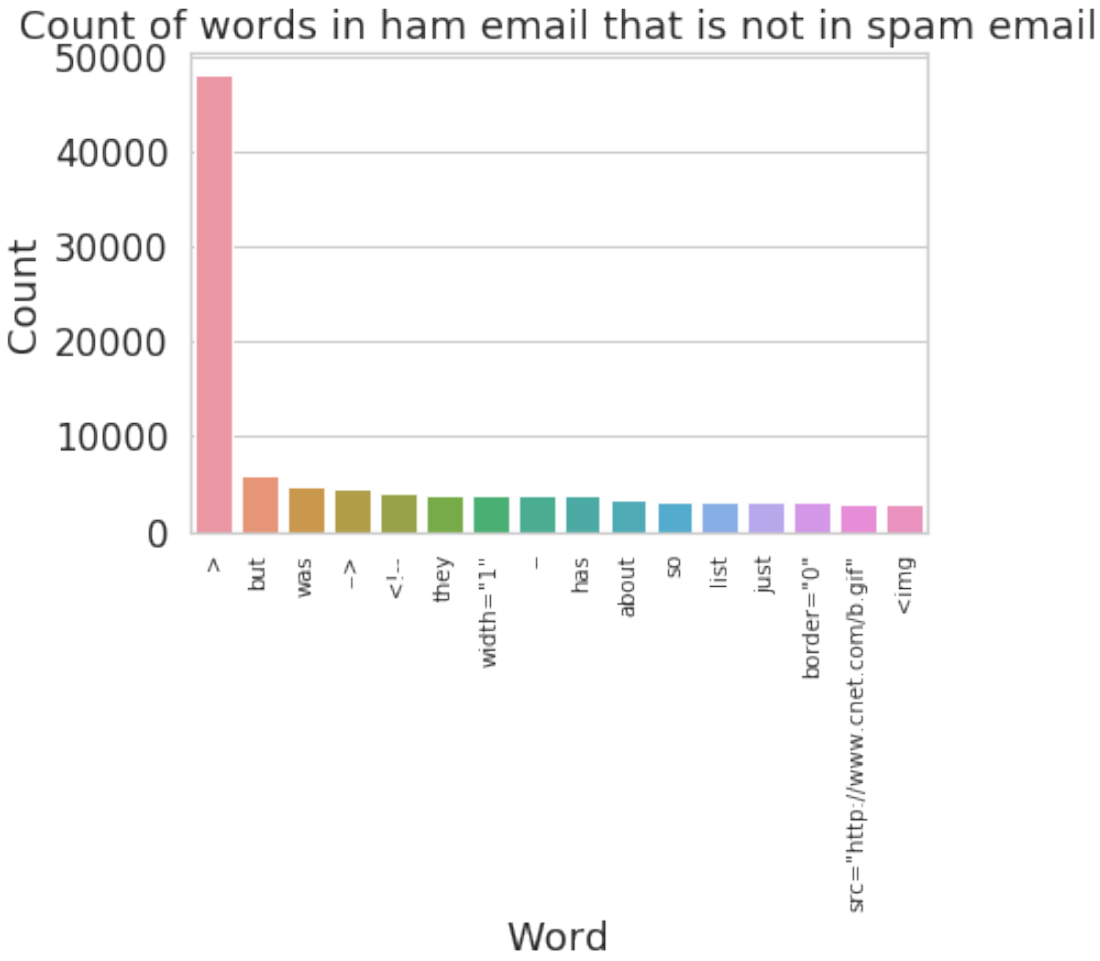
Generate your visualization in the cell below and provide your description in a comment.

In [364]:
```
# Write your description (2-3 sentences) as a comment here:
# From the code shown below, I have created four arrays. "spam_is_in" is an array of the coun
# of words in the spam email that also show up in the ham email. "ham_is_in" is the count
# of words in the ham email that also show up in the spam email. "spam_is_not_in" is the coun
# of words in the spam email that does not show up in the ham email. Finally, "ham_is_not_in"
# is the count  of words in the ham email that does not show up in the spam email. The two pl
# below are for "spam_is_not_in" and "ham_is_not_in", and they show me the most frequent word
# that don't show up for either case. This way, I could see the most frequent features, that
# show up only for the spam and ham emails.

# Write the code to generate your visualization here:
spam_is_in = spam_email_count[spam_email_count.keys().isin(ham_email_count.keys())]
ham_is_in = ham_email_count[ham_email_count.keys().isin(spam_email_count.keys())]
spam_is_not_in = spam_email_count[np.logical_not(spam_email_count.keys().isin(ham_email_count
ham_is_not_in = ham_email_count[np.logical_not(ham_email_count.keys().isin(spam_email_count.k


plot_spam_is_not_in = sns.barplot(x=spam_is_not_in.keys(),y=spam_is_not_in)
plot_spam_is_not_in.set(xlabel = "Word", ylabel = "Count", title = "Count of words in spam ema
plot_spam_is_not_in.tick_params(axis="x", labelsize=10)
plot_spam_is_not_in.set_xticklabels(plot_spam_is_not_in.get_xticklabels(), rotation=90)
plt.subplots()
plot_ham_is_not_in = sns.barplot(x=ham_is_not_in.keys(),y=ham_is_not_in)
plot_ham_is_not_in.tick_params(axis="x", labelsize=10)
plot_ham_is_not_in.set_xticklabels(plot_ham_is_not_in.get_xticklabels(), rotation=90)
plot_ham_is_not_in.set(xlabel = "Word", ylabel = "Count", title = "Count of words in ham emai
```

Out[364]:
```
[Text(0, 0.5, 'Count'),
 Text(0.5, 0, 'Word'),
 Text(0.5, 1.0, 'Count of words in ham email that is not in spam email')]
```


Count of words in spam email that is not in ham email
Count
3000
2000
1000
0
<tr>

our
helvetica,
</td>
<p
email
sans-serif"
</table>
free
no
arial,
<tr>=20
please
out
<div
Word


Count of words in ham email that is not in spam email
Count
50000
40000
30000
20000
10000
0
>
but
was
->
<!--
they
width="1"
–
has
about
so
list
just
border="0"
src="http://www.cnet.com/b.gif"
<img
Word

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 an ROC curve for your final classifier (the one you use to make predictions for Kaggle). Refer to the Lecture 22 notebook or Section 17.7 of the course text to see how to plot an ROC curve.

In [366]:
```
ham_email_count2 = pd.Series(' '.join(ham_email).lower().split()).value_counts()[:150]
spam_email_count2 = pd.Series(' '.join(spam_email).lower().split()).value_counts()[:150]

spam_is_not_in2 = spam_email_count2[np.logical_not(spam_email_count2.keys().isin(ham_email_co

spam_is_not_in2
```

Out[366]:
```
helvetica,              3054
<p                      2475
sans-serif"             1922
… Omitting 49 lines …
go                       565
internet                 562
dtype: int64
```