Notebook

August 6, 2019

0.0.1 Question 1c

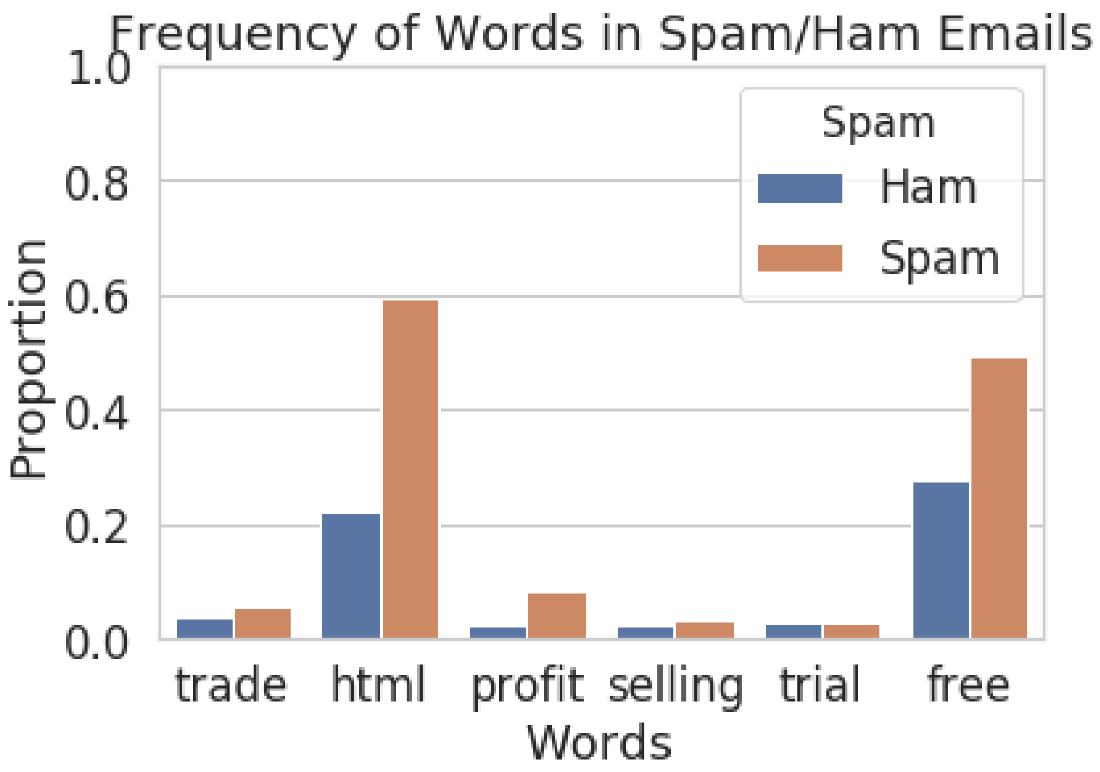
Discuss one thing you notice that is different between the two emails that might relate to the identification of spam.

There is html formatting and normally unusual symbols in the spam whereas the ham does not contain any and all the links are hyperlinks in the ham 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`.

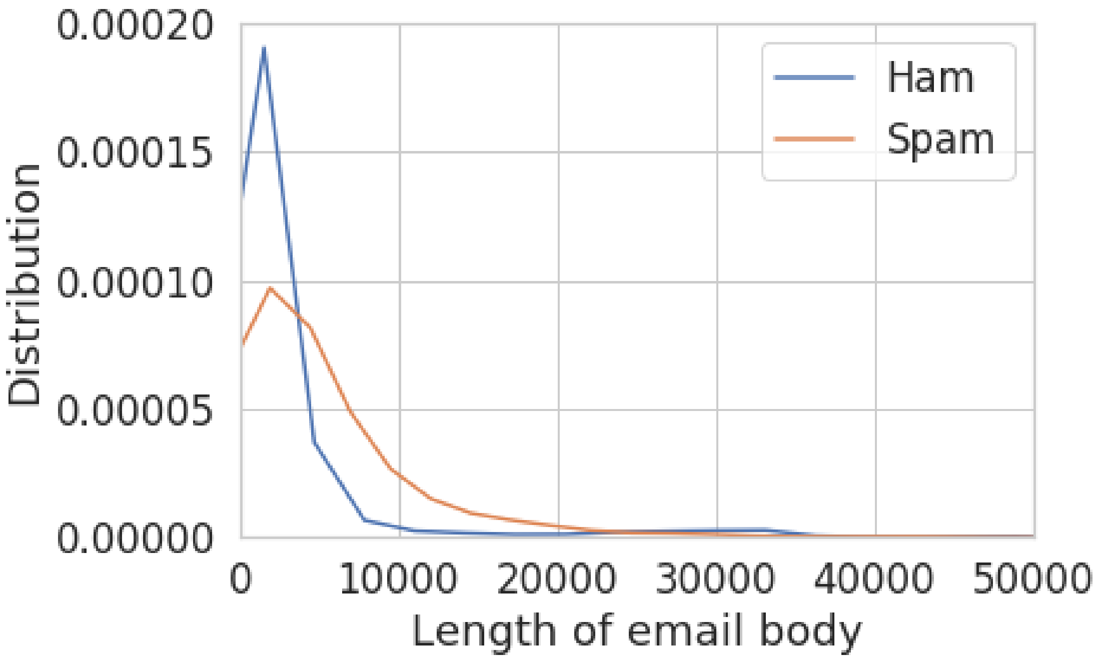
```
In [60]: train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai

         words = ['trade', 'html', 'profit', 'selling', 'trial', 'free']
         ham = train[train['spam'] ==0]
         spam = train[train['spam']==1]
         ham_T = words_in_texts(words, ham['email']).T
         spam_T = words_in_texts(words, spam['email']).T
         hams_proportion = [sum(HAM_T) / len(ham) for HAM_T in ham_T]
         spams_proportion = [sum(SPAM_T) / len(spam) for SPAM_T in spam_T]
         Certain_Words = pd.DataFrame(data = {'Words': words + words,
                                   'Proportion': np.append(hams_proportion, spams_proportion),
                                   'Spam': np.append(np.repeat('Ham', len(words)), np.repeat('Spam', len
         plt.ylim(0,1)
         plt.title('Frequency of Words in Spam/Ham Emails')
         sns.barplot(x='Words', y='Proportion', hue='Spam', data = Certain_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 [61]: train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai

         ham_length = train[train['spam'] ==0]['email'].apply(len)
         spam_length = train[train['spam']==1]['email'].apply(len)

         sns.distplot(ham_length, label = 'Ham', hist=False)
         sns.distplot(spam_length, label = 'Spam', hist= False)
         plt.xlim(0,50000)
         plt.xlabel('Length of email body')
         plt.ylabel('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)?

We see that there are no false positives while there are 1918 false negatives. Since accuracy computes the true positives and true negative overall the total, we note that there are more false negatives. For recall, it computes our true positives over false positives and true negatives indicating that it's not very good at predicting the email as spam or ham.

0.0.5 Question 6e

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

There are more false negatives when using the logistic regression classifier from Question 5

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.

Question 1, it is about the same as predicting 0 for every email

Question 2, one reason this classifier is performing poorly may be the frequency that these words appear meaning there are cases where the classifier could falsely predict a word as spam, i.e. a false alarm

Question 3, even though there are more false negatives when using the logistic regression classifier, I would prefer that for spam filter because it is better to predict something that is actually spam as not spam (False Negative) than False Positives because we do not want important emails to be lost in a spam folder. Thus a 75% accuracy is acceptable for a spam filter.

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 the model, I looked at the subject line as well to see which words could perform better. Also comparing that to the number of words in the subject line and body of the email. Also I wanted to see whether punctuation seemed to be important from earlier when we noticed that some emails were in HTML format so I wanted to take a look at that as well.

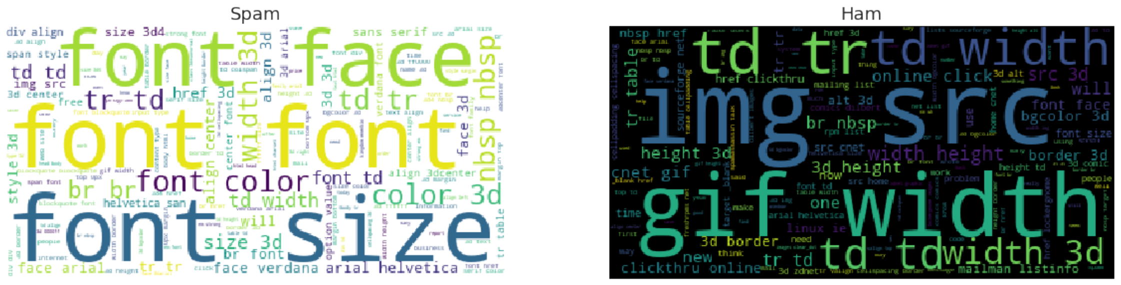
2. The emails with the HTML formatting were not able to improve the model well and some words were that not seem to affect the accuracy by much because they might words that are commonly used in both ham and spam emails, thus it could be a frequent issue in some emails. Also finding other words after finding the frequency that some words appear in the subject and body helped to improve the model.

3. Surprisingly, the subject line words have a big impact on the accuracy when I was finding good features. Typically people tend to identify spam from the contents of an email but knowing what the subject line contains helped my model perform better than picking words from the email itself.

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

```
In [72]: # Write your description (2-3 sentences) as a comment here:
         # The wordcloud helps to visualize which words have high or low values
         # We can see that for spam most frequent words are html related
         # While for ham the most frequent words are gif, width, img, src

         # Write the code to generate your visualization here:
         !pip install WordCloud
         import numpy as np
         import pandas as pd
         from os import path
         from PIL import Image
         from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
         spams = val[val['spam'] ==1]['email']
         hams = val[val['spam'] ==0]['email']
         wordcloud_spams=WordCloud(background_color='white').generate(' '.join(spams))
         wordcloud_hams=WordCloud().generate(' '.join(hams))

         plt.figure(figsize=(20,12))
         plt.subplot(1,2,1)
         plt.imshow(wordcloud_spams)
         plt.title('Spam')
         plt.axis('off')
         plt.subplot(1,2,2)
         plt.imshow(wordcloud_hams)
         plt.title('Ham')
         plt.axis('off');
```

```
Requirement already satisfied: WordCloud in /srv/conda/envs/data100/lib/python3.6/site-packages
Requirement already satisfied: pillow in /srv/conda/envs/data100/lib/python3.6/site-packages (from Word
Requirement already satisfied: numpy>=1.6.1 in /srv/conda/envs/data100/lib/python3.6/site-packages (fro
You are using pip version 9.0.1, however version 19.2.1 is available.You should consider upgrading via
```



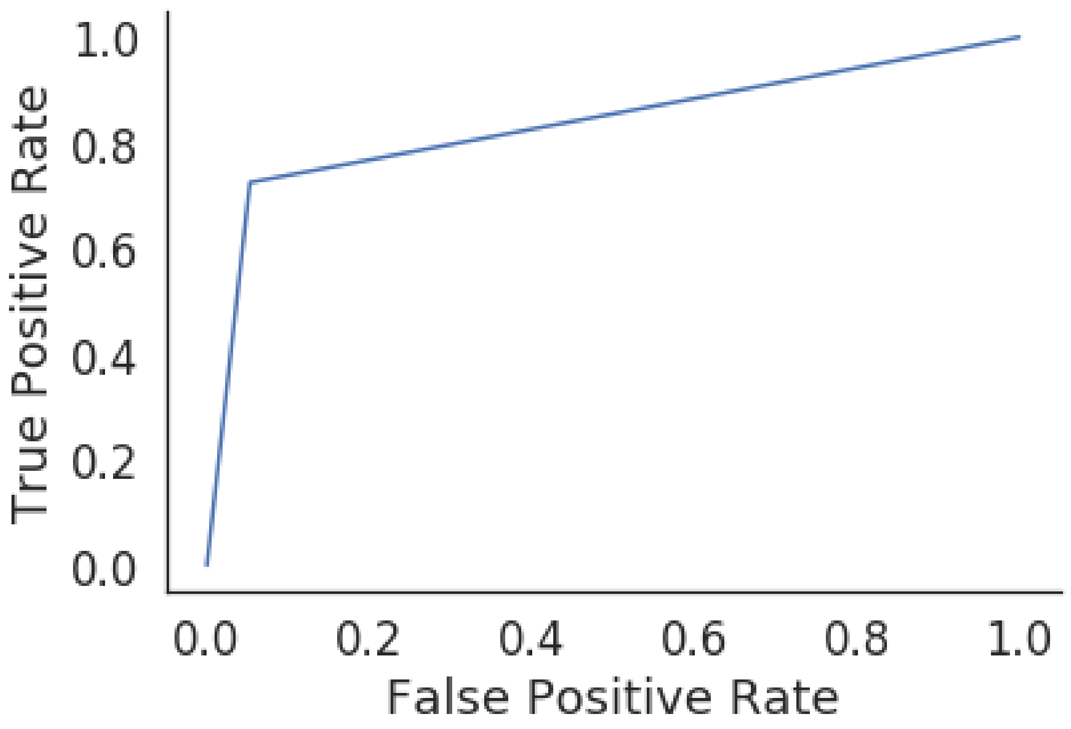
0.0.8 Question 9: ROC Curve

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 20 notebook to see how to plot an ROC curve.

```
In [74]: from sklearn.metrics import roc_curve
         from sklearn.linear_model import LogisticRegression
         # Note that you'll want to use the .predict_proba(...) method for your classifier
         # instead of .predict(...) so you get probabilities, not classes

         email_words = ['nbsp', 'click',' url', '.com', '!', 'width', 'src', 'img', 'size', 'font', 'co
         X_email = words_in_texts(email_words, train['email'])

         subject_words = ['Cost', 'ILUG', 'Re', 'Fw', 'price', 'Save', 'extra', 'Congrats', 'Membership
         X_subject = words_in_texts(subject_words, train['subject'])

         x = np.hstack([X_email, X_subject])
         y = np.array(train['spam'])

         model_roc = LogisticRegression()
         model_roc.fit(x, y)

         y_predict = (model_roc.predict_proba(x)[:,0] <=0.55).astype(int)

         fpr, tpr, thresholds = roc_curve(y, y_predict)
         with sns.axes_style('white'):
                 plt.plot(fpr,tpr)

         sns.despine()
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate');
```


True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0