Notebook

December 2, 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.

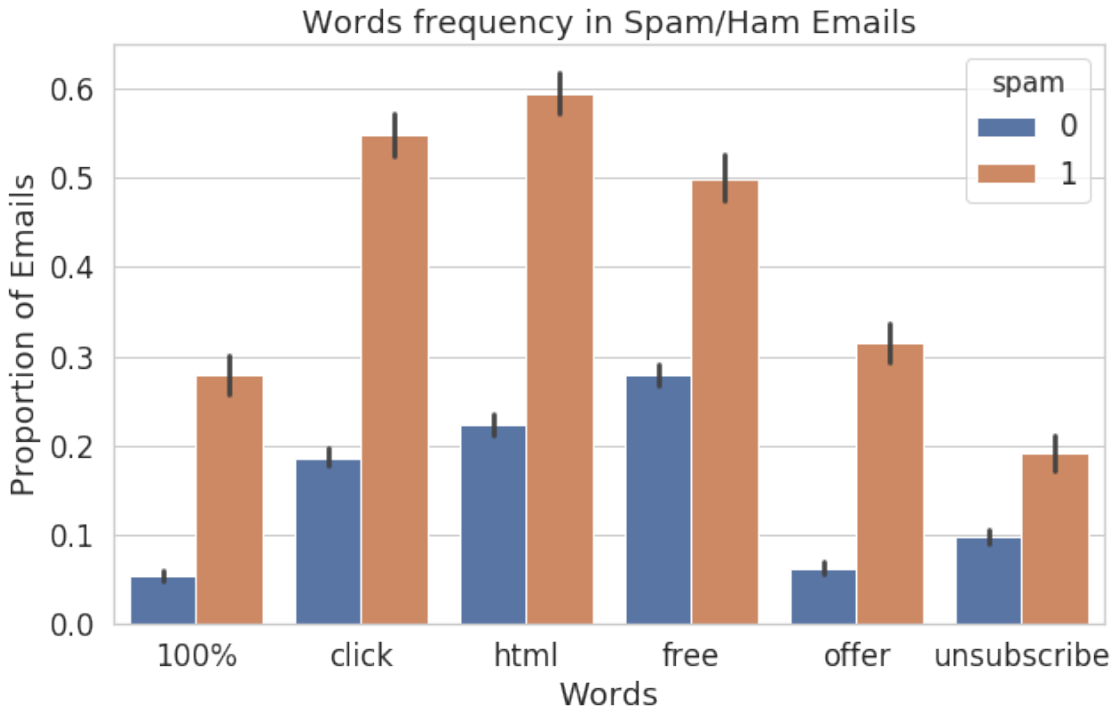
The spam email body has the email embedded in HTML. It also contains language that is more fishy, wants an action related to either clicking a link or paying for something. The ham email body is legibly formatted in plaintext with a date and some embedded urls.

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 [41]:
```
train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai

plt.figure(figsize = (10,6))
spam_words = ['100%', 'click', 'html', 'free', 'offer', 'unsubscribe']
by_word = words_in_texts(spam_words, train['email'])
by_word = pd.DataFrame(by_word)
by_word['spam'] = train['spam']
by_word = by_word.melt('spam')
sns.barplot(x = by_word['variable'], y = by_word['value'], hue = by_word['spam']).set(xticklabe
plt.xlabel('Words')
plt.ylabel('Proportion of Emails')
plt.title('Words frequency in Spam/Ham Emails')
plt.show()
```

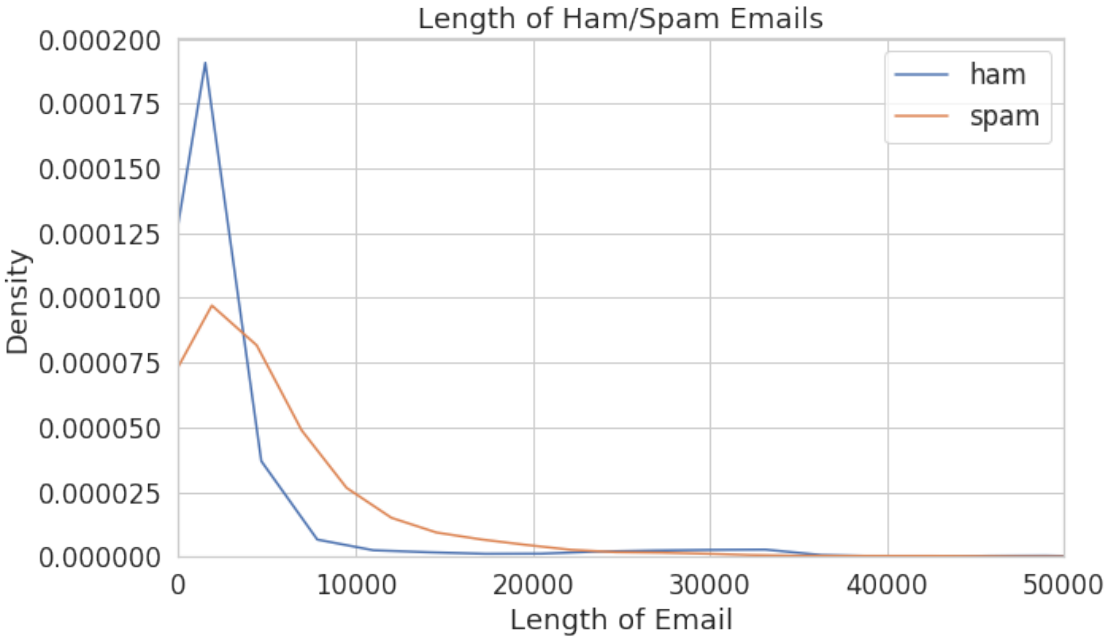


0.0.3 Question 3b

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]: train['length'] = train['email'].apply(len)
         ham = train[train['spam']==0]
         spam = train[train['spam']==1]
         fig = plt.figure(figsize=(10,6))
         sns.distplot(ham['length'], hist = False, label = 'ham').set(xlim = (0,50000))
         sns.distplot(spam['length'], hist = False, label = 'spam').set(xlim = (0,50000))
         plt.xlabel('Length of Email')
         plt.ylabel('Density')
         plt.title('Length of Ham/Spam Emails')
```

```
Out[14]: Text(0.5, 1.0, 'Length of Ham/Spam Emails')
```



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)?

The zero_predictor function sends all emails, regardless of any feature, to the inbox. In other words all emails, whether spam or ham, will get classified as ham. The zero_predictor catches absolutely no spam emails.

Thus, the number of true positives (spam classified as spam) and false positives (ham classified as spam) are both zero. The number of true negatives (ham classified as ham) and false negatives (spam classified as ham) will simply be the number of ham emails and the number of spam emails, respectively, since all ham and all spam get classified as ham.

Finally, these four values help explain accuracy and recall of the zero_predictor model. Accuracy refers to the proportion of emails classified correctly: TP + TN / # of emails. No spam is caught (TP), so we get back the proportion of ham emails in the training data. Recall refers to of all predicted positive, the proportion classified correctly: TP / TP + FN. Since we catch no spam (TP), the recall of this model 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 more false negatives than false positives using this classifier.

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. Our zero_predictor model got a 74.4% accuracy. The logistic regression model has a very slightly better prediction accuracy of 75.8%.

2. The word features we used include the words 'drug', 'bank', 'prescription', 'memo', 'private'. Some of these words can definitely appear in spam. However, many of these words can also be used in ham emails. For example, a spam email containing the word "bank" could be an email fishing for a user's bank account information. A ham email containing the word "bank" could be an email from a user's actual bank signed off with "Chase Bank". Similarly, many of the other word features are not mutually exclusive to both classes.

3. I prefer the logistic regression classifier. The zero predictor classifier sends all emails to inbox. The false positives and true positives are effectively zero, making the accuracy low. It is not filtering. The logistic regression classifier has already shown some improvement with the accuracy score over the zero predictor accuracy. With better features, I believe the logistic regression classifier will be able to better catch more false positives, making accuracy better since it is filtering.

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. I found better features by visualizing and optimizing each feature. For example, when selecting word features, I bar plotted the proportions every single time I added a word. I then only kept the words which I deemed had a significant enough difference in proportion. I also realized that all the features I had based on email bodies could also be used on the subject line. Adding features for subjects in the mix also helped!

2. I found that word features were really difficult to identify! I started off by manually looking at some email bodies and trying to qualitatively identify words. I then cleaned the data and utilized Counter from the Collections library to find common spam/ham words. This definitely helped identify better words, but after experimenting with a couple different feature sets, I didn't find that adding words was as nearly as helpful as I thought it might be.

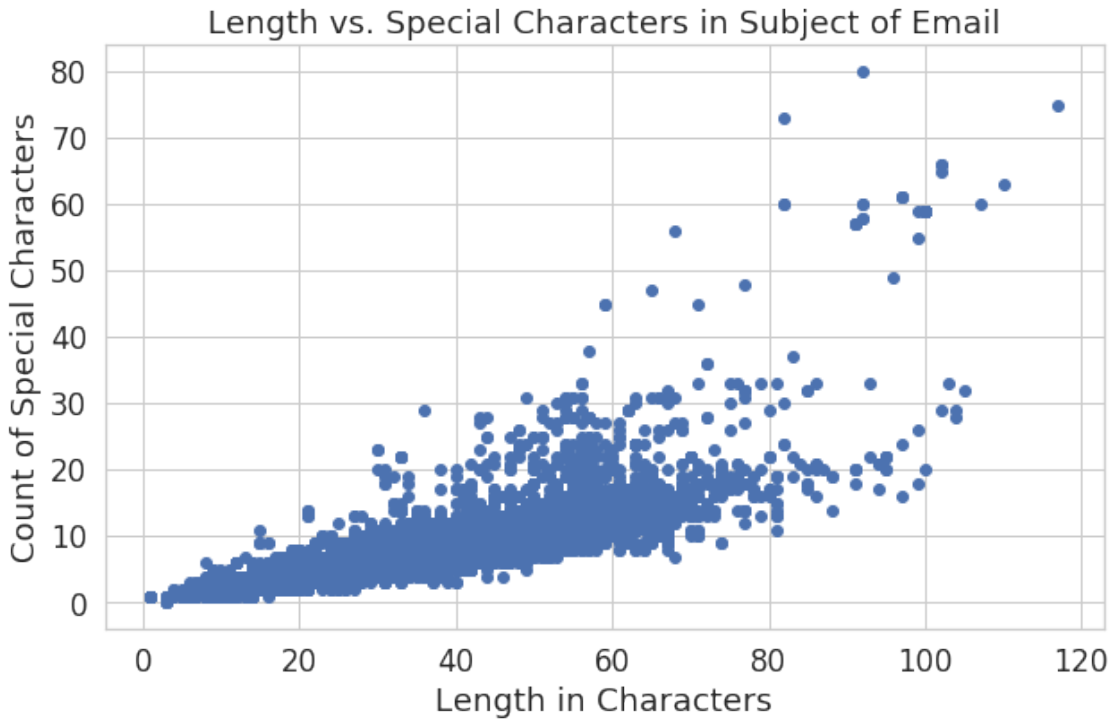
3. Something that surprised me was how much removing stopwords affected my model. I was stuck on how using both # of characters and # of words # of punctuations could all be helpful. The numbers got very skewed by the stopwords. By writing out my own list of frequent stopwords, it immensely helped improve the features I had already written out as well as identify better word features. If I went further to extract html plain text from the html, I'm sure this would have also helped better identify spam emails.

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

In [34]:
```
# Write your description (2-3 sentences) as a comment here:
'''
This is a scatter plot of the length versus number of special characters
in the subject line of an email. These features are linearly related,
indicating a bit of redundancy. I found it easier to choose features which
catch True Postives (spam) instead of True Negatives (ham) I chose to keep
both features, as high values in both indicate higher probability of an email
being spam. They become less correlated at high values, so keeping both features
helped my model retain better accruacy.
'''

# Write the code to generate your visualization here:
plt.figure(figsize = (10,6))
plt.scatter(train.iloc[:,5], train.iloc[:,8])
plt.xlabel("Length in Characters")
plt.ylabel("Count of Special Characters")
plt.title("Length vs. Special Characters in Subject of Email")
```

Out[34]:
```
Text(0.5, 1.0, 'Length vs. Special Characters in Subject of Email')
```



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 [35]:
```
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

probability = model.predict_proba(X_train)[:, 1]
FPR, TPR, thresholds = roc_curve(Y_train, probability, pos_label=1)

plt.plot(FPR, TPR)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

Out[35]:
```
Text(0, 0.5, 'True Positive Rate')
```