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 seems to be a premade HTML template urging a reader to take action and the other ham email is just written in plain text, not HTML, not urging a reader to take particular action. So a big difference between the two emails is the way they are written and their formatting.

0.0.2 Question 3a

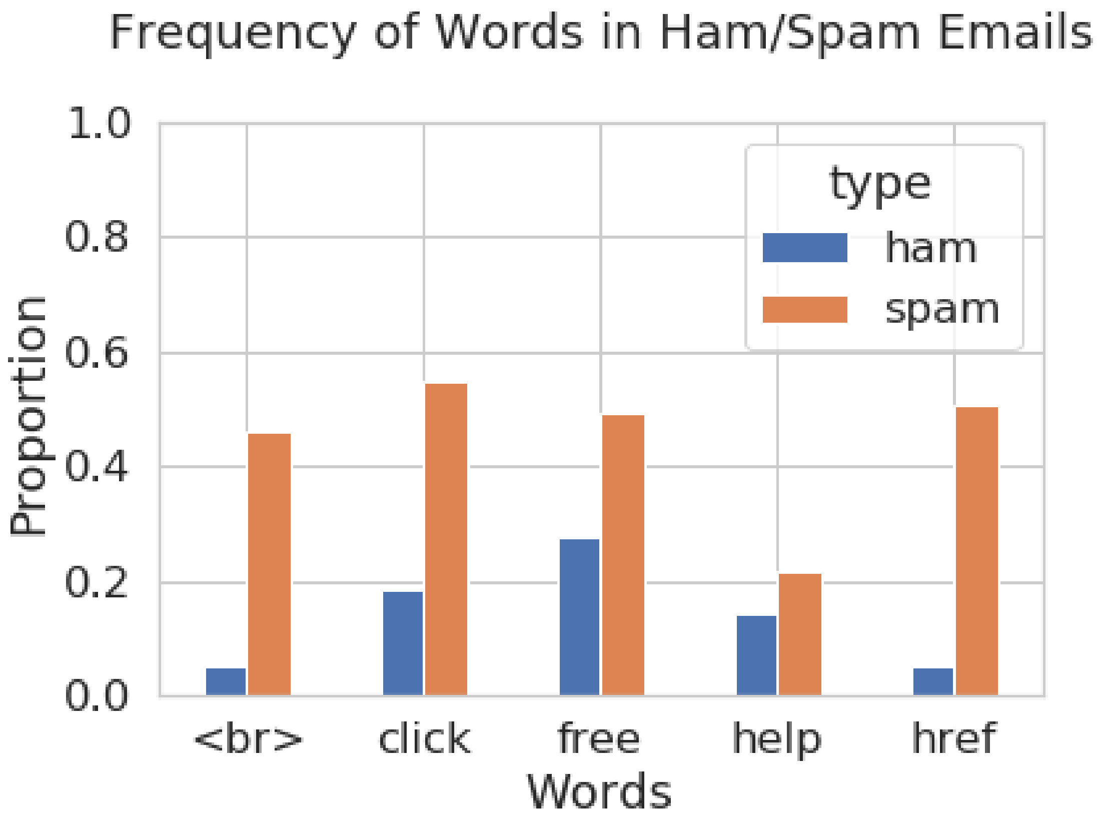
Create a bar chart like the one above comparing the proportion of spam and ham emails containing certain words. Choose a set of words that are different from the ones above, but also have different proportions for the two classes. Make sure to only consider emails from `train`.

```
In [136]: train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of ema

          spam_count = train['spam'].sum()
          ham_count = len(train['spam']) - spam_count

          words_of_interest = ['free', 'href', 'help', 'click', '<br>']
          email_texts = train['email']
          indicator_array = words_in_texts(words_of_interest, email_texts)
          prop_df = pd.DataFrame(indicator_array, columns=words_of_interest)
          prop_df['type'] = train['spam']
          prop_df['type'] = prop_df['type'].apply(lambda x: 'spam' if x == 1 else 'ham')
          melted = prop_df.melt('type')
          melted = melted.groupby(['variable','type']).sum().unstack()
          df = melted['value']
          df['ham'] = df['ham'] / ham_count
          df['spam'] = df['spam'] / spam_count
          df.plot(kind='bar')
          plt.xlabel('Words')
          plt.xticks(rotation=0)
          plt.ylabel('Proportion')
          plt.ylim(top = 1.0)
          plt.title('Frequency of Words in Ham/Spam Emails', y=1.1)
```

```
Out[136]: Text(0.5, 1.1, 'Frequency of Words in Ham/Spam Emails')
```


Frequency of Words in Ham/Spam Emails
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
type
ham
spam

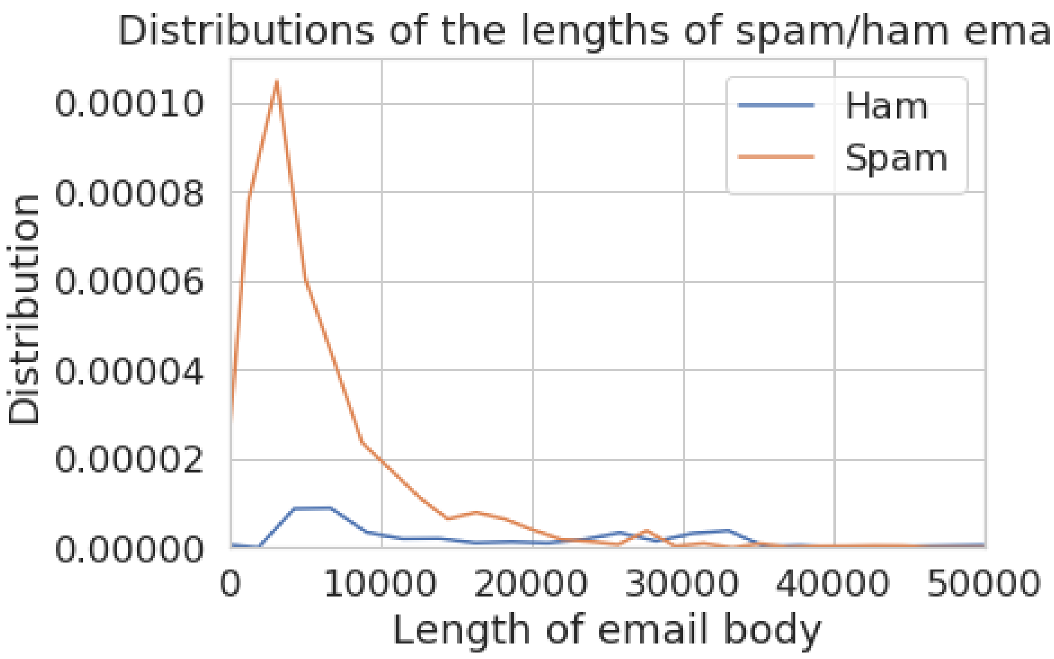
click
free
help
href
Words

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 [137]: spam_lengths = train[train['spam'] == 1]['email'].apply(lambda x: len(x))
          ham_lengths = train[train['spam'] == 0]['email'].apply(lambda x: len(x))
          sns.distplot(ham_lengths, hist = False, label = 'Ham')
          sns.distplot(spam_lengths, hist = False, label = 'Spam')
          plt.xlabel('Length of email body')
          plt.ylabel('Distribution')
          plt.title('Distributions of the lengths of spam/ham emails')
          plt.xlim(0,50000)
```

```
Out[137]: (0, 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)?

Since we are using a zero predictor, we never have any positive values, so we have no true or false positives. Therefore, zero_predictor_fp = 0 because we never have any positive values. The number of false negatives would just be the number of total positive values we actually have in the set. Our false negatives would just come from our total of positive values because they would all be predicted to be negative. Since we can only correctly predict all negative values, our accuracy would just be the number of negative classifications (which we always get right) divided by the total number of emails. Since we have 0 positive values, our recall will be zero (obtained by plugging in TP=0 into the recall formula).

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 positives when we use a Logistic Regression classifier as opposed to the zero predictor because the logistic regression classifier will actually label some of the spam emails while the zero predictor will just label everything (including spam) as zero (not spam). Using this same logic, there will be less false negatives because some emails do get labeled correctly.

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. The logistic regression classifier performs slightly better and achieves a better accuracy (75.6) when compared to predicting zero for every email (74.5).

2. The word features we were given (drug, bank, prescription, memo, private) because they are not prevalent throughout our set of spam and ham emails, so we cannot make a clear distinction between spam and ham using these words which results in our classifier's poor performance.

3. I would prefer using the logistic regression classifier because it would actually manage to detect and label some of the spam emails as opposed to the zero predictor, which would label every single email as not spam. The recall of the zero classifier is 0, which indicates that none of the spam emails were caught. The logistic regression predictor had a recall of approximately 11.4%, which indicates that it performs a much better job of actually catching spam emails when compared to the zero predictor.

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 for my models by searching for patterns in the subject and email fields and finding a way to quantify them for my logistic regression model. Once I found a feature I wanted to analyze, such as amount of capital letters, number of html tags, and usage of specific words, I tested to see the accuracy of the model with the newly included feature using cross validation with 40 folds. If the feature improved the model's accuracy and reduced cross validation error, I kept it, otherwise, I would search for new features.

2. Something that I tried that did work was using percentages instead of counts. For example, simply counting the number of capital characters in an email did not work for the model and when I included it as a feature, the cross validation function would fail to find an optimum value to use. However, when I changed to percentages, (counting the number of capital letters and then dividing it by the total length of the actual email), cross validation worked and the model was able to work successfully.

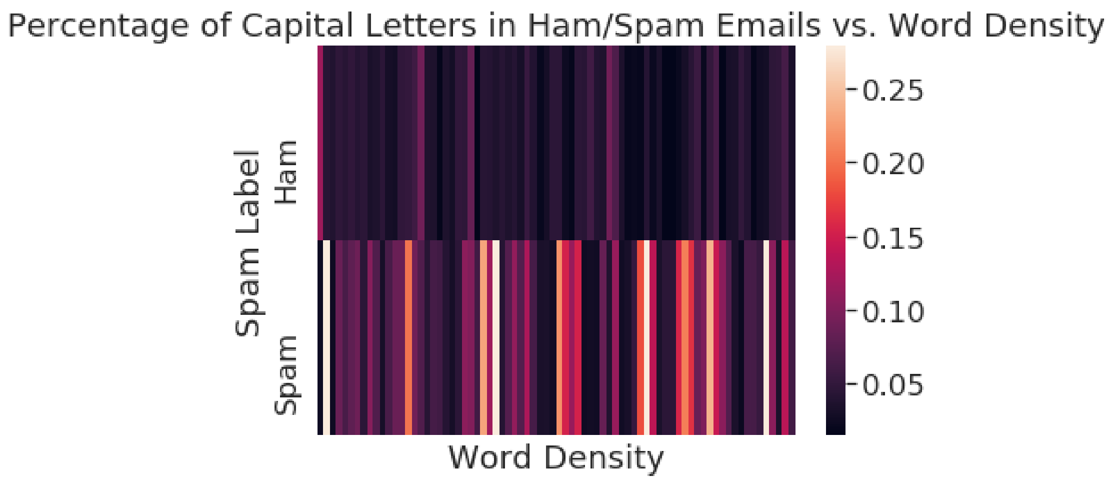
3. Something surprising I found in my search for good features was that not all ham emails were formatted nicely in terms of containing just plain text. Some emails may have triggered the spam classifier since they have many repeating, unnecessary characters such as '<' and '>' (not from HTML tags).

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

```
In [153]: # Write your description (2-3 sentences) as a comment here:
          # A plot that I found useful for feature selection was a heatmap that illustrated
          # the correlation between the word density (wordiness) and amount of
          # capital letters in the email. As seen in the heatmap, as spam emails
          # became more wordy, the colors in the heatmap were relatively
          # brighter (higher percentage of capital letters) for spam
          # emails than ham emails (ham emails appeared to have
          # an almost constant color). This relationship that shows
          # wordier emails with a lot of capital letters helped me decide to include
          # percentage of capital letters as a useful feature
          # for distinguishing between spam/ham emails.

          # Write the code to generate your visualization here:
          df = eda_data.copy()[['email','spam','cap_count','word_density','html','punct']]
          df['len'] = df['email'].apply(len)
          df = df.reset_index()
          match_len = pd.Series(np.intersect1d(df[df['spam'] == 0]['word_density'],df[df['spam'] == 1][
          df = df[df['word_density'].isin(match_len)]
          df = pd.pivot_table(df, index='spam', columns='word_density',values='cap_count')
          df
          ax = sns.heatmap(df, xticklabels=False, robust=True)
          ax.set_yticklabels(['Ham','Spam'])
          plt.xlabel('Word Density')
          plt.ylabel('Spam Label')
          plt.title('Percentage of Capital Letters in Ham/Spam Emails vs. Word Density')
```

```
Out[153]: Text(0.5, 1, 'Percentage of Capital Letters in Ham/Spam Emails vs. Word Density')
```



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.

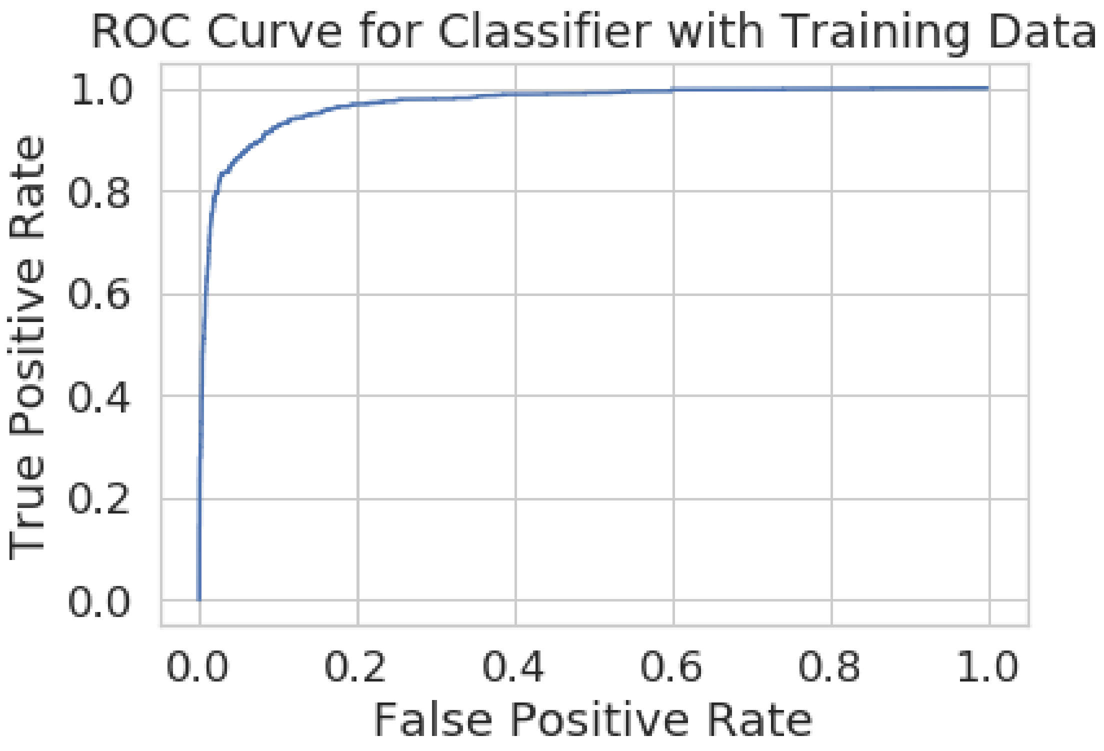
The ROC curve shows this trade off for each possible cutoff probability. In the cell below, plot a ROC curve for your final classifier (the one you use to make predictions for Gradescope) on the training data. Refer to Lecture 19 or Section 17.7 of the course text to see how to plot an ROC curve.

```
In [154]: 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
          predicted_probs = model.predict_proba(final_clean_train)[:,1]
          predicted_probs

          false_positive_rates, sensitivity_vals, thresholds = roc_curve(observed_Y, predicted_probs, p
          plt.plot(false_positive_rates, sensitivity_vals)
          plt.xlabel('False Positive Rate')
          plt.ylabel('True Positive Rate')
          plt.title('ROC Curve for Classifier with Training Data')
```

```
Out[154]: Text(0.5, 1.0, 'ROC Curve for Classifier with Training Data')
```


ROC Curve for Classifier with Training Data
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0