Notebook

November 24, 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 is in html while ham is not.

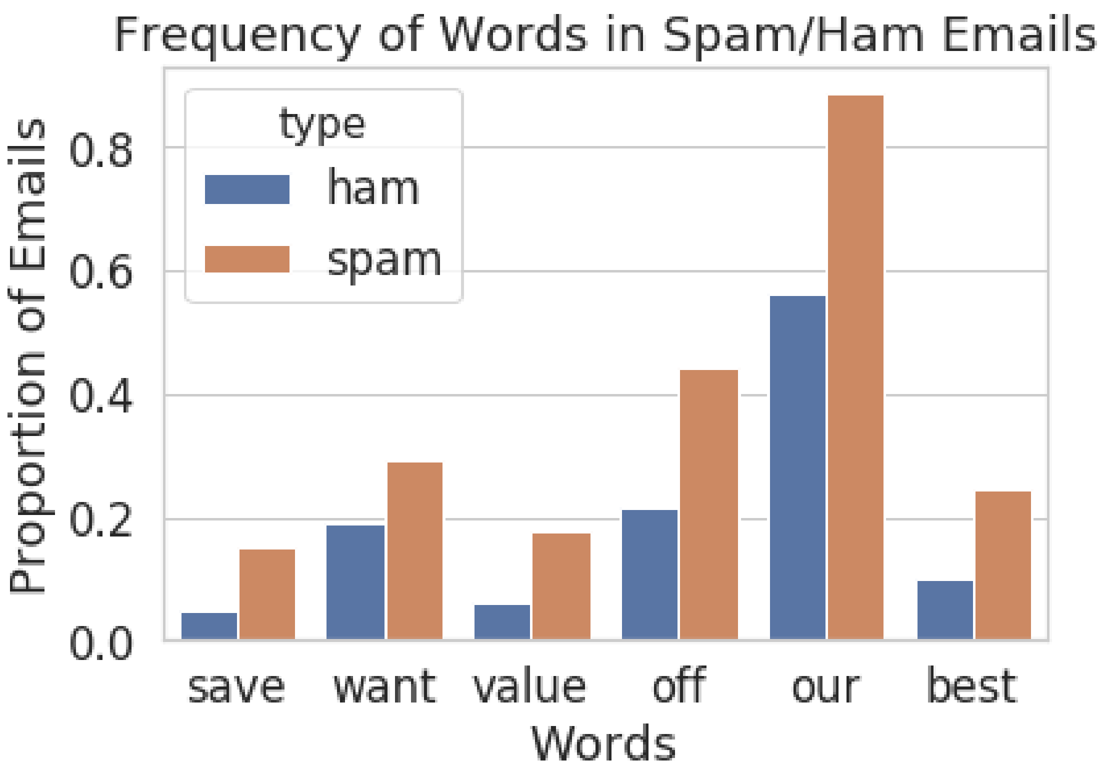
0.0.2 Question 3a

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

         word_lst = ["save", "want", "value", "off", "our", "best"]
         if_contains = words_in_texts(word_lst, train["email"])
         words = pd.DataFrame({"type": train["spam"], "if_contains": if_contains.tolist()})
         sep_words = pd.DataFrame(words.if_contains.values.tolist(), index = words.index)
         sep_words["type"] = words["type"]
         sep_words = sep_words.rename(columns = {0: word_lst[0], 1: word_lst[1], 2: word_lst[2], 3: wor
         sep_words = sep_words.melt("type")
         sep_words["type"] = sep_words["type"].map({0: "ham", 1: "spam"})
         #sep_words
         sns.barplot(x = "variable", y = "value", data = sep_words, hue = "type", ci = None)
         plt.title("Frequency of Words in Spam/Ham Emails")
         plt.xlabel("Words")
         plt.ylabel("Proportion of Emails")

Out[12]: Text(0, 0.5, 'Proportion of Emails')
```

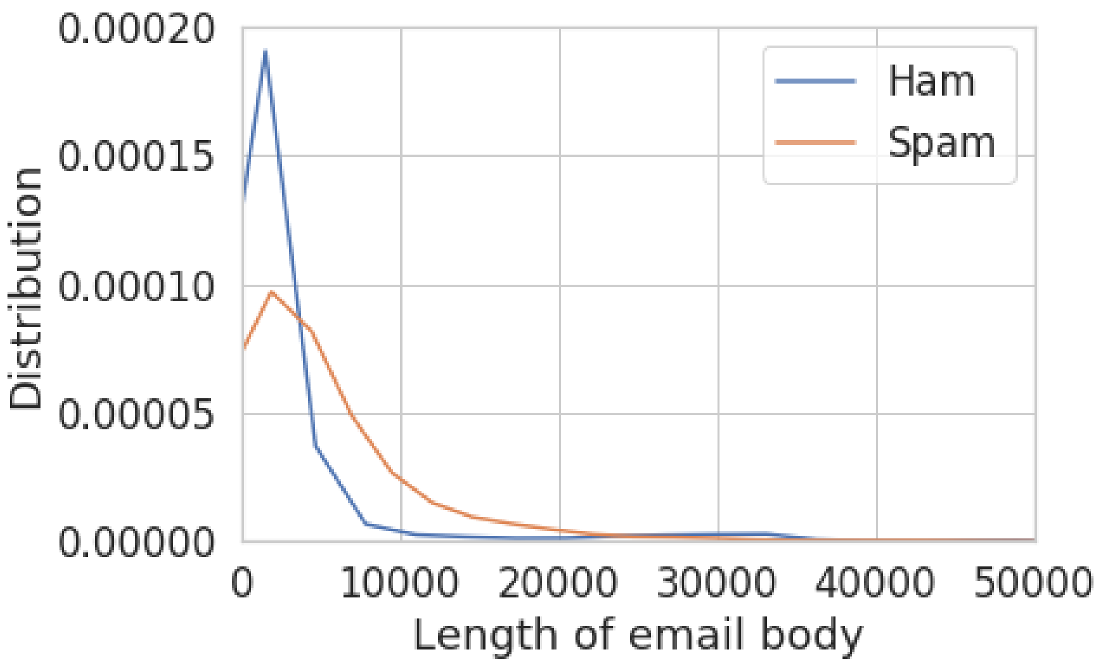


0.0.3 Question 3b

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 [13]: lens = [len(i) for i in train["email"]]
         lenths_hs = pd.DataFrame({"type": train["spam"], "lens": lens})
         lenths_hs["type"] = lenths_hs["type"].map({0: "ham", 1: "spam"})
         lenths_hs
         #type(lenths_hs["lens"])
         sns.distplot(lenths_hs[lenths_hs["type"] == "ham"]["lens"], hist = False, label = "Ham")
         sns.distplot(lenths_hs[lenths_hs["type"] == "spam"]["lens"], hist = False, label = "Spam")
         plt.xlim(0, 50000)
         plt.xlabel("Length of email body")
         plt.ylabel("Distribution")
         plt.legend()
```

```
Out[13]: <matplotlib.legend.Legend at 0x7f5b591f07f0>
```



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)?

For FP, it is the number of email that is classified as spam but is actually ham. Our zero_predictor classifies every email as ham, so that there would be no email that is classified as spam, so FP = 0.

For FN, it is the number of email that is classified as ham but is actually spam. Our zero_predictor classifies every email as ham, so that it would just be those emails that has a wrong classification result as there are no emails classifies as spam.

For accuracy, it is the number of emails whose classification result is the same as its actual type, so that it would just be the proportion of hams in all emails as our zero_predictor classifies every email as ham.

For recall, the formula calculating it is $\frac{TP}{TP+FN}$. Since our zero_predictor classifies every email as ham, there would be no spam emails that is correctly classified as spam, so TP = 0 and recall = 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 negative when using the logistic regression classifier from Question 5, as from 6d we can see that the value of precision is larger than the value of recall, indicating that there are more false negative comparing to false prositives.

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. From 6b we can see that the accuracy of predicting 0 for every email is around 74.5%, which is a bit lower comparing to the logfistic regression classifier's accuracy.

2. One reason that this classifier is performing poorly might be the rare occurance os the words in the email set. The rare occurance means that these words are not the main differences between spam and ham emails, so that our logistic regression classifier performs poorly.

3. I prefer the zero_predictor more, as we can see that the zero_predictor has a lower false alarm rate comparing to the logistic classifier, so that there would be less email falsely labeled as spam.

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 first did all the code stuff, and I tried to calculte rmse for every combination of features and used np.argmin to find the combination that has the lowest rmse.

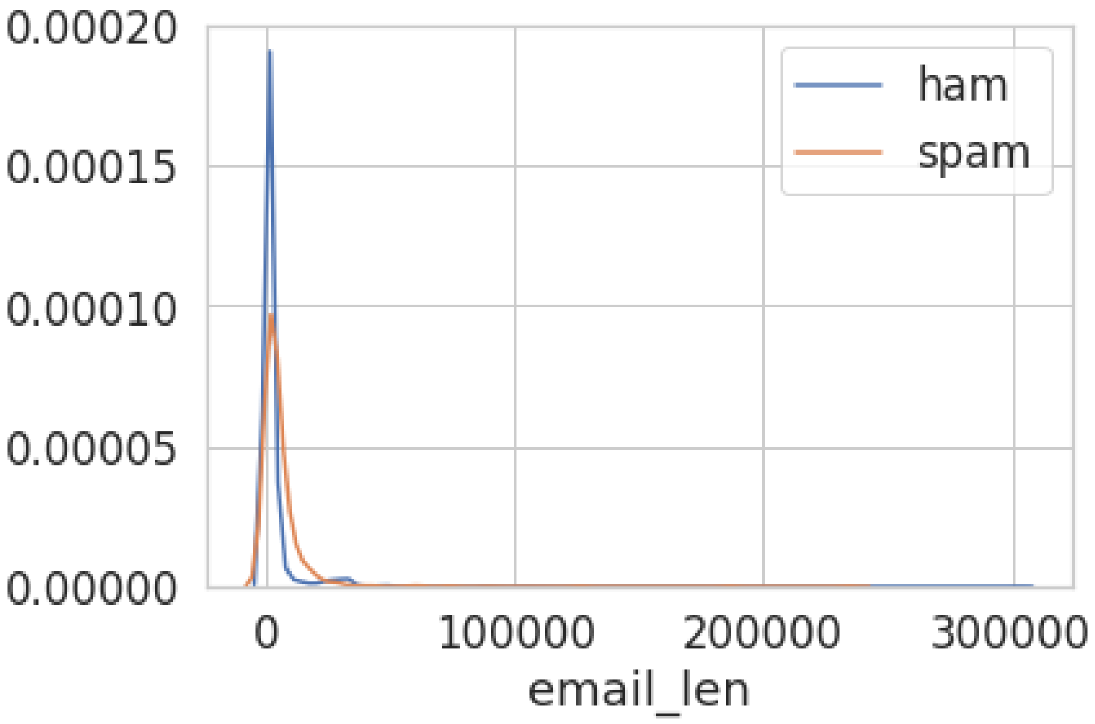
2. I find that only use keywords does not work. I spent a lot of time trying different keywords and I find my accuracy stays nearly the same. So I tried to add more features like the length of the email, which works.

3. I was surprising that although some prevalent words may have a large differnce between spam and ham, adding or dropping it actually changes nothing or really just a bit to my model.

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

```
In [24]: with_features = train.copy()
         with_features["email_len"] = with_features["email"].apply(len)
         sns.distplot(with_features[with_features["spam"] == 0]["email_len"], hist = False, label = "ha
         sns.distplot(with_features[with_features["spam"] == 1]["email_len"], hist = False, label = "sp
         plt.legend()
```

```
Out[24]: <matplotlib.legend.Legend at 0x7f5b5811e048>
```



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.

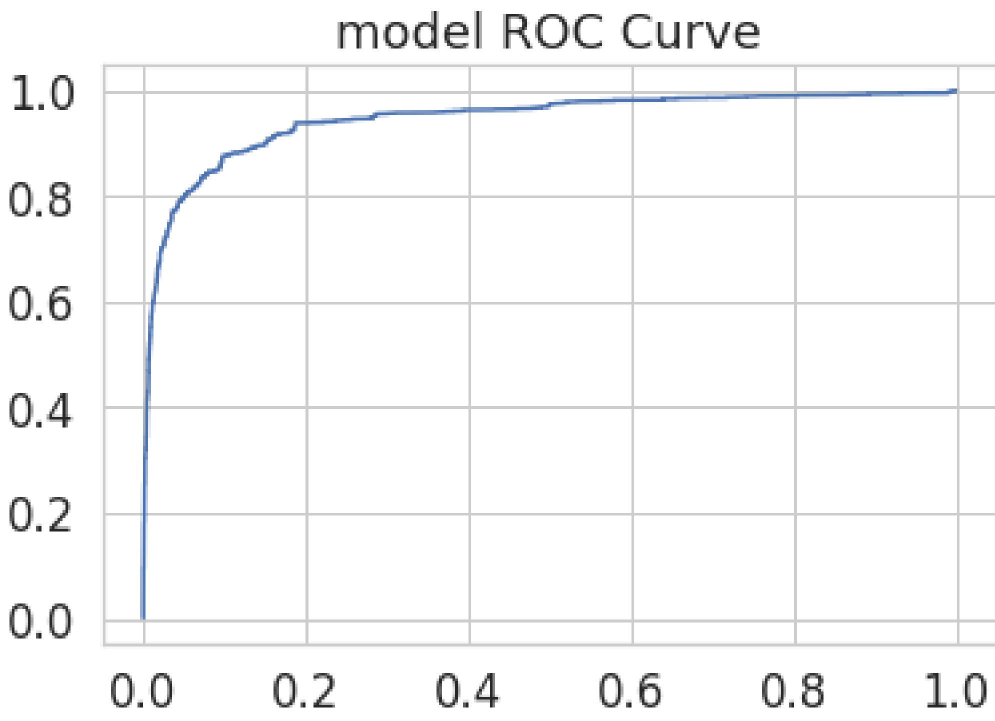
```
In [41]: from sklearn.metrics import roc_curve

         # Note that you'll want to use the .predict_proba(...) method for your classifier
         # instead of .predict(...) so you get probabilities, not classes

         y_scores = final_model.predict_proba(X_train_final)
         scores = [i[1] for i in y_scores]
         false_positive_rate_values, sensitivity_values, thresholds = roc_curve(Y_train_final, scores,

         plt.step(false_positive_rate_values, sensitivity_values, color='b', where='post')
         plt.title('model ROC Curve')
```

```
Out[41]: Text(0.5, 1.0, 'model ROC Curve')
```


model ROC Curve