

proj2b

May 15, 2020

```
[1]: # Initialize OK
from client.api.notebook import Notebook
ok = Notebook('proj2b.ok')
```

```
=====
Assignment: proj2b
OK, version v1.13.11
=====
```

1 Project 2 Part B: Spam/Ham Classification

1.1 Classifiers

1.1.1 The assignment is due on Monday, April 27th at 11:59pm PST.

Collaboration Policy

Data science is a collaborative activity. While you may talk with others about the project, we ask that you **write your solutions individually**. If you do discuss the assignments with others please **include their names** at the top of your notebook.

Collaborators: *list collaborators here*

1.2 This Assignment

In Project 2 Part A, you made an effort to understand the data through EDA, and did some basic feature engineering. You also built a Logistic Regression model to classify Spam/Ham emails. In Part B, you will learn how to evaluate the classifiers you built. You will also have the chance to improve your model by selecting more features.

1.3 Warning

We've tried our best to filter the data for anything blatantly offensive as best as we can, but unfortunately there may still be some examples you may find in poor taste. If you encounter these

examples and believe it is inappropriate for students, please let a TA know and we will try to remove it for future semesters. Thanks for your understanding!

1.4 Score Breakdown

Question	Points
6a	1
6b	1
6c	2
6d	2
6e	1
6f	3
7	6
8	6
9	15
Total	37

1.5 Setup

```
[2]: import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
sns.set(style = "whitegrid",
        color_codes = True,
        font_scale = 1.5)
```

```
[3]: from utils import fetch_and_cache_gdrive
fetch_and_cache_gdrive('1SCASpLZFKCp2zek-toR3xeKX3DZnBSyp', 'train.csv')
fetch_and_cache_gdrive('1ZDFo90TF96B5GP2Nzn8P8-AL7CTQXmC0', 'test.csv')

original_training_data = pd.read_csv('data/train.csv')
test = pd.read_csv('data/test.csv')

# Convert the emails to lower case as a first step to processing the text
original_training_data['email'] = original_training_data['email'].str.lower()
test['email'] = test['email'].str.lower()

original_training_data.head()

from sklearn.model_selection import train_test_split
```

```
train, val = train_test_split(original_training_data, test_size=0.1,
    ↪random_state=42)
```

Using version already downloaded: Sun Apr 26 17:59:28 2020

MD5 hash of file: 0380c4cf72746622947b9ca5db9b8be8

Using version already downloaded: Sun Apr 26 17:59:29 2020

MD5 hash of file: a2e7abd8c7d9abf6e6fafc1d1f9ee6bf

The following code is adapted from Part A of this project. You will be using it again in Part B.

```
[4]: def words_in_texts(words, texts):
    '''
    Args:
        words (list-like): words to find
        texts (Series): strings to search in

    Returns:
        NumPy array of 0s and 1s with shape (n, p) where n is the
        number of texts and p is the number of words.
    '''
    indicator_array = 1 * np.array([texts.str.contains(word) for word in
    ↪words]).T
    return indicator_array

some_words = ['drug', 'bank', 'prescription', 'memo', 'private']

X_train = words_in_texts(some_words, train['email'])
Y_train = np.array(train['spam'])

X_train[:5], Y_train[:5]
```

```
[4]: (array([[0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0],
            [0, 0, 0, 1, 0]]), array([0, 0, 0, 0, 0]))
```

Recall that you trained the following model in Part A.

```
[5]: from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train, Y_train)

training_accuracy = model.score(X_train, Y_train)
print("Training Accuracy: ", training_accuracy)
```

Training Accuracy: 0.7576201251164648

1.6 Evaluating Classifiers

The model you trained doesn't seem too shabby! But the classifier you made above isn't as good as this might lead us to believe. First, we are evaluating accuracy on the training set, which may provide a misleading accuracy measure, especially if we used the training set to identify discriminative features. In future parts of this analysis, it will be safer to hold out some of our data for model validation and comparison.

Presumably, our classifier will be used for **filtering**, i.e. preventing messages labeled **spam** from reaching someone's inbox. There are two kinds of errors we can make: - False positive (FP): a ham email gets flagged as spam and filtered out of the inbox. - False negative (FN): a spam email gets mislabeled as ham and ends up in the inbox.

These definitions depend both on the true labels and the predicted labels. False positives and false negatives may be of differing importance, leading us to consider more ways of evaluating a classifier, in addition to overall accuracy:

Precision measures the proportion $\frac{TP}{TP+FP}$ of emails flagged as spam that are actually spam.

Recall measures the proportion $\frac{TP}{TP+FN}$ of spam emails that were correctly flagged as spam.

False-alarm rate measures the proportion $\frac{FP}{FP+TN}$ of ham emails that were incorrectly flagged as spam.

The following image might help:

Note that a true positive (TP) is a spam email that is classified as spam, and a true negative (TN) is a ham email that is classified as ham.

1.6.1 Question 6a

Suppose we have a classifier `zero_predictor` that always predicts 0 (never predicts positive). How many false positives and false negatives would this classifier have if it were evaluated on the training set and its results were compared to `Y_train`? Fill in the variables below (answers can be hard-coded):

Tests in Question 6 only check that you have assigned appropriate types of values to each response variable, but do not check that your answers are correct.

```
[6]: Y_train.size
```

```
[6]: 7513
```

```
[7]: Y_train.shape
```

```
[7]: (7513,)
```

```
[8]: zero_predictor_fp = 0
     zero_predictor_fn = len([y for y in Y_train if y == 1])
```

```
[9]: ok.grade("q6a");
```

```

~~~~~
Running tests

-----

Test summary
  Passed: 2
  Failed: 0
[ooooooooook] 100.0% passed

```

1.6.2 Question 6b

What are the accuracy and recall of `zero_predictor` (classifies every email as ham) on the training set? Do **NOT** use any `sklearn` functions.

```
[10]: spam_in_Ytrain = [y for y in Y_train if y == 1]
```

```
[11]: len(spam_in_Ytrain)
```

```
[11]: 1918
```

```
[12]: len([y for y in Y_train if y == 0])/Y_train.size
```

```
[12]: 0.7447091707706642
```

```
[13]: zero_predictor_acc = len([y for y in Y_train if y == 0])/Y_train.size
      zero_predictor_recall = 0
```

```
[14]: ok.grade("q6b");
```

```

~~~~~
Running tests

-----

Test summary
  Passed: 2
  Failed: 0
[ooooooooook] 100.0% passed

```

1.6.3 Question 6c

Provide brief explanations of the results from 6a and 6b. Explain why the number of false positives, number of false negatives, accuracy, and recall all turned out the way they did.

Explanations of the results:

- For Q6a:

=> Since zero_predictor always predicts 0 (ham), it will not predict any positive values and thus the FPs is 0.

=> Since zero_predictor never predict true positive (it always returns 0, not 1), the false negative (0) is the total number of spams.

- For Q6b:

=> Since all emails are considered hams (not spams at all), the number of accuracy is equal to the number of hams divided by the total number of emails.

=> We have $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$ is zero since all predicted values are 0s (no positive is predicted).

- Reasons number of false positives, number of false negatives, accuracy, and recall all turned out the way they did: => Because the definition of zero_predictor, it always predicts 0 (never predicts positive, or 1).

=> Thus, all values of our prediction in Y_train will become 0 no matter what its actual value is

=> Meaning, the of #FPs = 0 (never positive, only 0), the number of false negatives are the total number of spams.

=> The accuracy is high even though we know for sure it is incorrect.

=> Recall is zero since the zero_predictor is never 1, thus the TP is always 0.

1.6.4 Question 6d

Compute the precision, recall, and false-alarm rate of the `LogisticRegression` classifier created and trained in Part A. Do **NOT** use any `sklearn` functions.

Note: In lecture we used the `sklearn` package to compute the rates. Here you should work through them using just the definitions to help build a deeper understanding.

```
[15]: Y_train_hat = model.predict(X_train)
```

```
[16]: X_train
```

```
[16]: array([[0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0],
          ...,
          [0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0]])
```

```
[17]: Y_train
```

```
[17]: array([0, 0, 0, ..., 0, 0, 0])
```

```
[18]: Y_train_hat
```

```
[18]: array([0, 0, 0, ..., 0, 0, 0])
```

```
[19]: Y_train_hat = model.predict(X_train)
      TP = np.sum((Y_train_hat == Y_train) & (Y_train_hat == 1))
      TN = np.sum((Y_train_hat == Y_train) & (Y_train_hat == 0))
      FP = np.sum((Y_train_hat != Y_train) & (Y_train_hat == 1))
      FN = np.sum((Y_train_hat != Y_train) & (Y_train_hat == 0))

      logistic_predictor_precision = TP / (TP + FP)
      logistic_predictor_recall = TP / (TP + FN)
      logistic_predictor_far = FP / (FP + TN)

      print(logistic_predictor_precision)
      print(logistic_predictor_recall)
      print(logistic_predictor_far)
```

```
0.6422287390029325
0.11418143899895725
0.021805183199285077
```

```
[20]: ok.grade("q6d");
```

```
~~~~~
Running tests

-----
Test summary
  Passed: 3
  Failed: 0
[ooooooooook] 100.0% passed
```

1.6.5 Question 6e

Are there more false positives or false negatives when using the logistic regression classifier from Part A?

```
[21]: FP
```

```
[21]: 122
```

```
[22]: FN
```

```
[22]: 1699
```

Since $FN = 1699$ and $FP = 122$, there are more false negatives than false positives.

1.6.6 Question 6f

1. Our logistic regression classifier got 75.8% 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) Since predicting all 0s for every email (no spam at all) yields 74.47%, which is less than 75.6%, thus our logistic regression classifier is better, but not a lot better, only 1% better.
 - 2) There are words that happen to be in both spam and ham emails. For example, 'memo'.
 - 3) I would prefer the Logistic Regression Classifier for a spam filter, the reason is because it has higher prediction accuracy.

1.7 Moving Forward

With this in mind, it is now your task to make the spam filter more accurate. In order to get full credit on the accuracy part of this assignment, you must get at least **88%** accuracy on the test set. To see your accuracy on the test set, you will use your classifier to predict every email in the `test` DataFrame and upload your predictions to Kaggle.

Kaggle limits you to four submissions per day. This means you should start early so you have time if needed to refine your model. You will be able to see your accuracy on the entire set when submitting to Kaggle (the accuracy that will determine your score for question 9).

Here are some ideas for improving your model:

1. Finding better features based on the email text. Some example features are:
 1. Number of characters in the subject / body
 2. Number of words in the subject / body
 3. Use of punctuation (e.g., how many '!' were there?)
 4. Number / percentage of capital letters
 5. Whether the email is a reply to an earlier email or a forwarded email
2. Finding better (and/or more) words to use as features. Which words are the best at distinguishing emails? This requires digging into the email text itself.
3. Better data processing. For example, many emails contain HTML as well as text. You can consider extracting out the text from the HTML to help you find better words. Or, you can match HTML tags themselves, or even some combination of the two.
4. Model selection. You can adjust parameters of your model (e.g. the regularization parameter) to achieve higher accuracy. Recall that you should use cross-validation to do feature and model selection properly! Otherwise, you will likely overfit to your training data.

You may use whatever method you prefer in order to create features, but **you are not allowed to import any external feature extraction libraries**. In addition, **you are only allowed to train logistic regression models**. No random forests, k-nearest-neighbors, neural nets, etc.

We have not provided any code to do this, so feel free to create as many cells as you need in order to tackle this task. However, answering questions 7, 8, and 9 should help guide you.

Note: You should use the **validation data** to evaluate your model and get a better sense of how it will perform on the Kaggle evaluation.

```
[23]: train.fillna(0, inplace=True)
```

```
/srv/conda/envs/data100/lib/python3.7/site-packages/pandas/core/frame.py:4259:
```

```
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
```

```
**kwargs
```

```
[24]: train.isnull().values.any()
```

```
[24]: False
```

```
[25]: hams = train[train['spam'] == 0]
      spams = train[train['spam'] == 1]
```

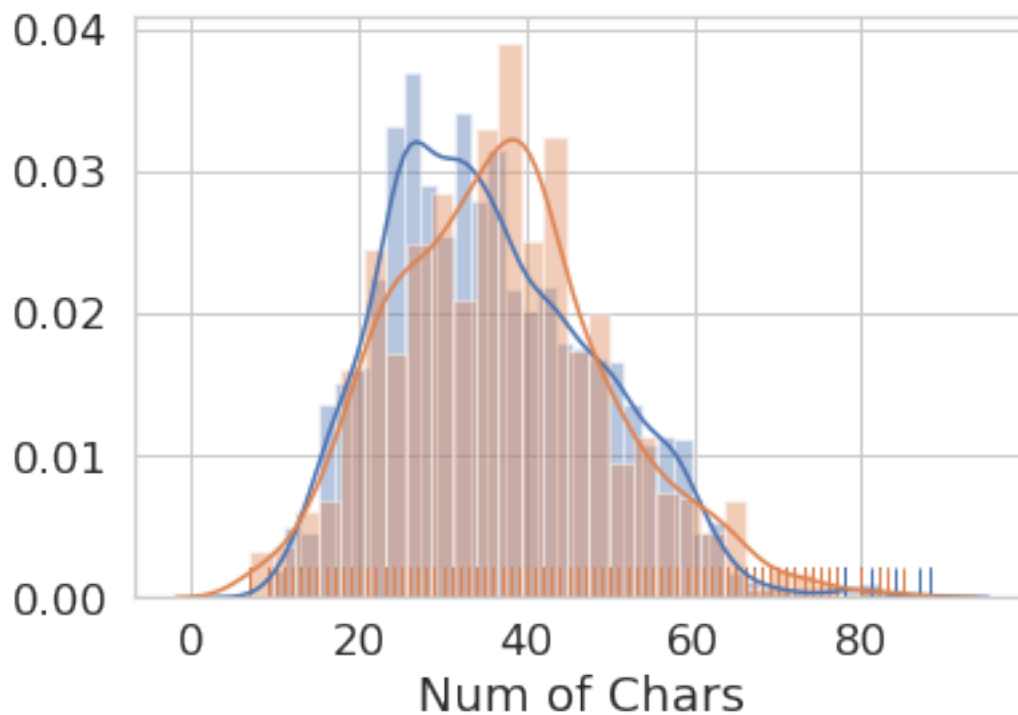
```
[26]: hams.isnull().values.any()
```

```
[26]: False
```

```
[27]: # Number of characters in the subject column Using simple DISTPLOT to
      ↪ distinguish between hams and spams
      def subject_words(df):
          return df['subject'].str.findall('\w').str.len()

      sns.distplot(subject_words(hams), label = 'ham', kde=True, rug=True, hist=True)
      sns.distplot(subject_words(spams), label = 'spam', kde=True, rug=True,
      ↪ hist=True)
      plt.xlabel('Num of Chars')
```

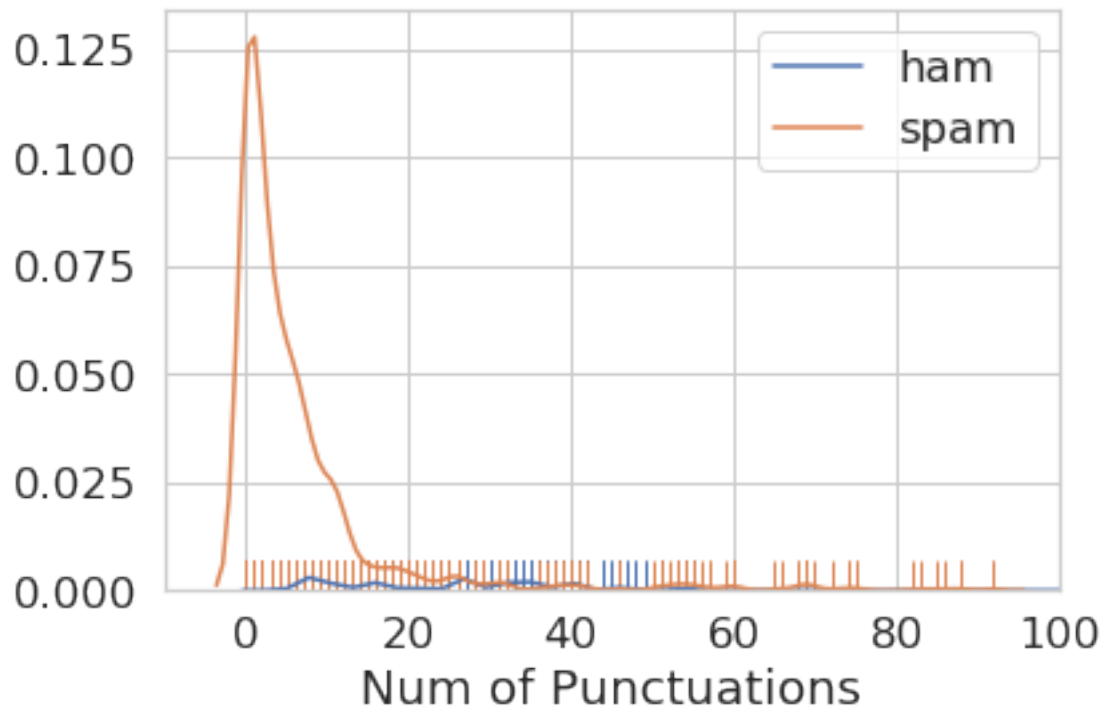
```
[27]: Text(0.5, 0, 'Num of Chars')
```



```
[28]: # Number of punctuations
def num_punc(df):
    return df['email'].str.findall('!').str.len().fillna(0)

sns.distplot(num_punc(hams), kde=True, label = 'ham', rug=True, hist=False)
sns.distplot(num_punc(spams), kde=True, label = 'spam', rug=True, hist=False)
plt.xlabel('Num of Punctuations')
plt.xlim(-10,100)
```

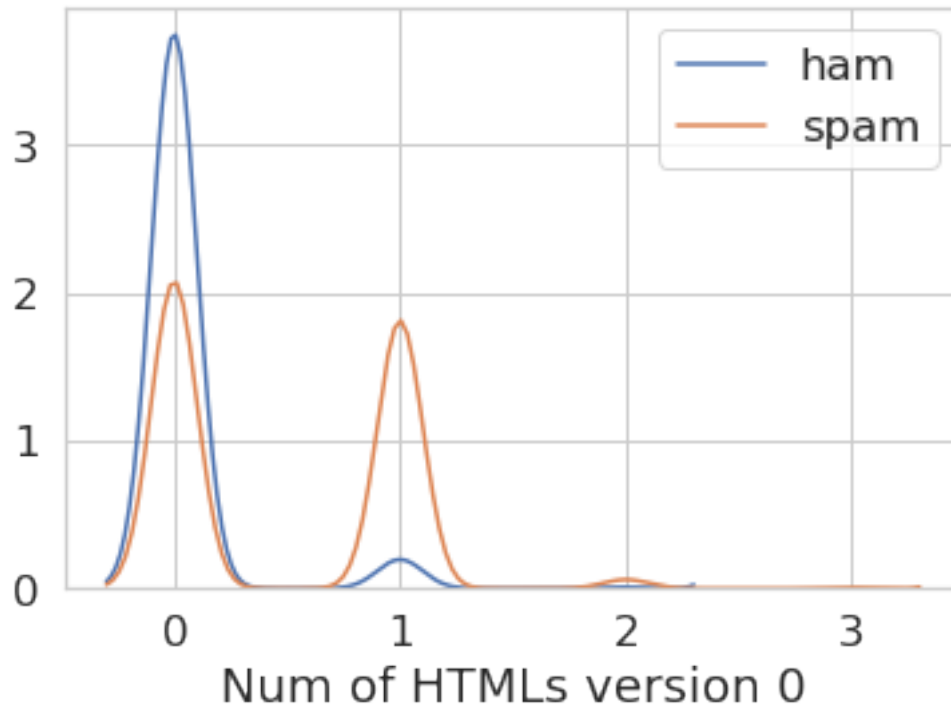
[28]: (-10, 100)



```
[29]: # Number of HTML tags
def num_html(df):
    return df["email"].str.count("<html>")

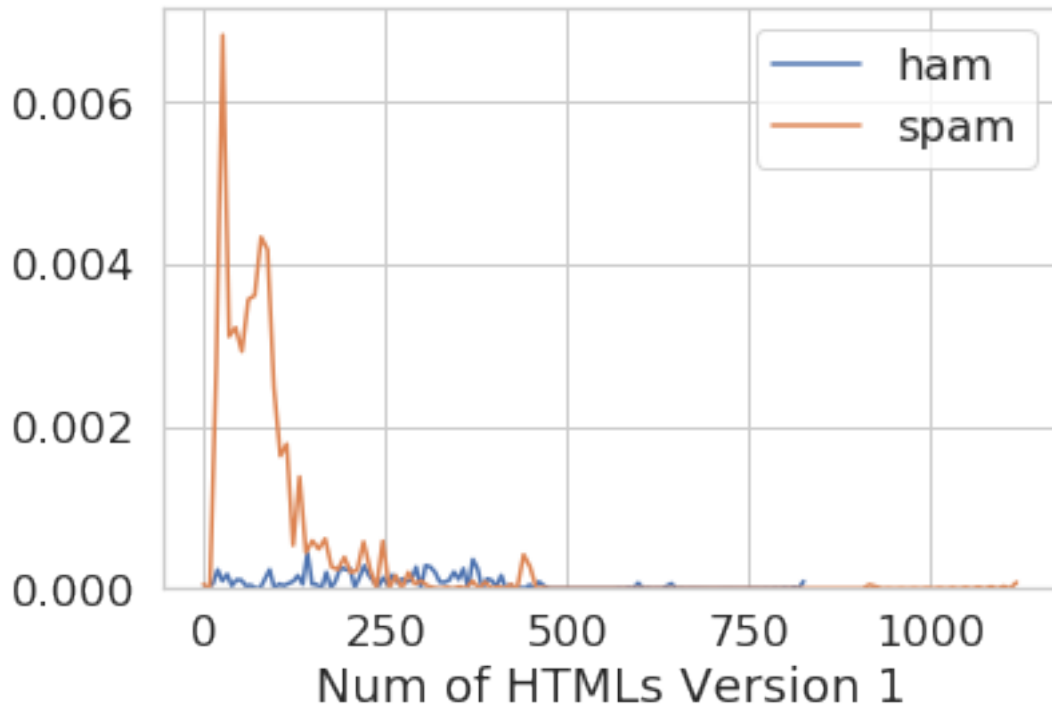
sns.distplot(num_html(hams), label = 'ham', hist=False, kde_kws={'bw':0.1})
sns.distplot(num_html(spams), label = 'spam', hist=False, kde_kws={'bw':0.1})
plt.xlabel('Num of HTMLs version 0')
```

```
[29]: Text(0.5, 0, 'Num of HTMLs version 0')
```



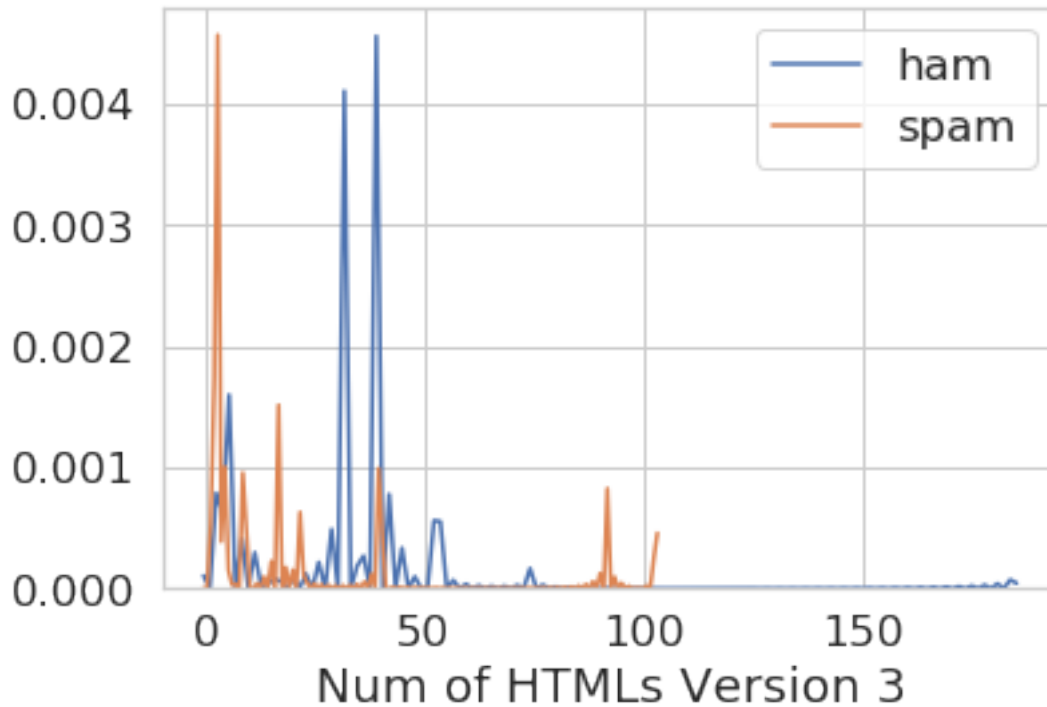
```
[30]: def num_html1(df):  
       return df["email"].str.count("</")  
  
sns.distplot(num_html1(hams), label = 'ham', hist=False, kde_kws={'bw':0.1})  
sns.distplot(num_html1(spams), label = 'spam', hist=False, kde_kws={'bw':0.1})  
plt.xlabel('Num of HTMLs Version 1')
```

```
[30]: Text(0.5, 0, 'Num of HTMLs Version 1')
```



```
[31]: def num_html2(df):  
        return df["email"].str.count("</>")  
  
sns.distplot(num_html2(hams), label = 'ham', hist=False, kde_kws={'bw':0.1})  
sns.distplot(num_html2(spams), label = 'spam', hist=False, kde_kws={'bw':0.1})  
plt.xlabel('Num of HTMLs Version 3')
```

```
[31]: Text(0.5, 0, 'Num of HTMLs Version 3')
```

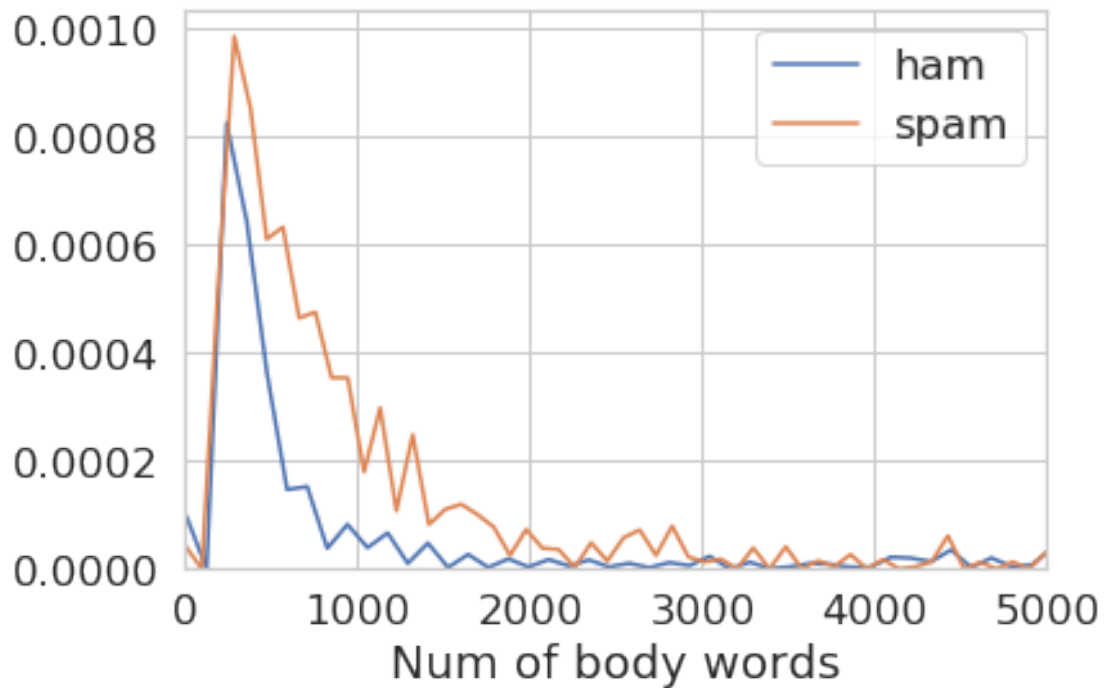


```
[32]: def num_body_words(df):
        return df['email'].str.findall('\w+').str.len()

sns.distplot(num_body_words(hams), label = 'ham', hist=False, kde_kws={'bw':0.
    ↪1})
sns.distplot(num_body_words(spams), label = 'spam', hist=False, kde_kws={'bw':0.
    ↪1})
plt.xlabel('Num of body words')
plt.xlim(0, 5000)

type(num_body_words(hams))
```

[32]: pandas.core.series.Series



```
[33]: # Return true if the email is a reply to another email
def is_rep(df):
    return df["subject"].str.contains("Re:").fillna(0).astype(int)
# Return true if the email is a forwarded email
def is_forward(df):
    return df["subject"].str.contains("Fw:").fillna(0).astype(int)

print("For Reply:")
print("Ham: ", is_rep(hams).sum(), "      ||      Spam: ", is_rep(spams).sum())

print("\nFor Forward:")
print("Ham: ", is_forward(hams).sum(), "      ||      Spam: ", is_forward(spams).
      ↪sum())
```

```
For Reply:
Ham:  2509      ||      Spam:  66
```

```
For Forward:
Ham:  8      ||      Spam:  25
```

```
[34]: # contains ip address
def has_link_http(df):
    return df['email'].str.contains('http').astype(int)
```

```

def has_link_org(df):
    return df["email"].str.contains("org").astype(int)

def has_link_net(df):
    return df["email"].str.contains("net").astype(int)

def has_link_com(df):
    return df["email"].str.contains("com").astype(int)

def has_free(df):
    return df["email"].str.contains("free").astype(int)

def has_offer(df):
    return df["email"].str.contains("offer").astype(int)

print("HTTP:")
print("Ham: ", has_link_http(hams).sum(), "      ||      Spam: ",
      ↪has_link_http(spams).sum())

print("\nORG:")
print("Ham: ", has_link_org(hams).sum(), "      ||      Spam: ",
      ↪has_link_org(spams).sum())

print("\nNET:")
print("Ham: ", has_link_net(hams).sum(), "      ||      Spam: ",
      ↪has_link_net(spams).sum())

print("\nCOM:")
print("Ham: ", has_link_com(hams).sum(), "      ||      Spam: ",
      ↪has_link_com(spams).sum())

print("\nFREE:")
print("Ham: ", has_free(hams).sum(), "      ||      Spam: ", has_free(spams).
      ↪sum())

print("\nOFFER:")
print("Ham: ", has_offer(hams).sum(), "      ||      Spam: ", has_offer(spams).
      ↪sum())

```

HTTP:

Ham: 4849 || Spam: 1610

ORG:

Ham: 1965 || Spam: 462

NET:
Ham: 2761 || Spam: 1016

COM:
Ham: 4718 || Spam: 1645

FREE:
Ham: 1551 || Spam: 943

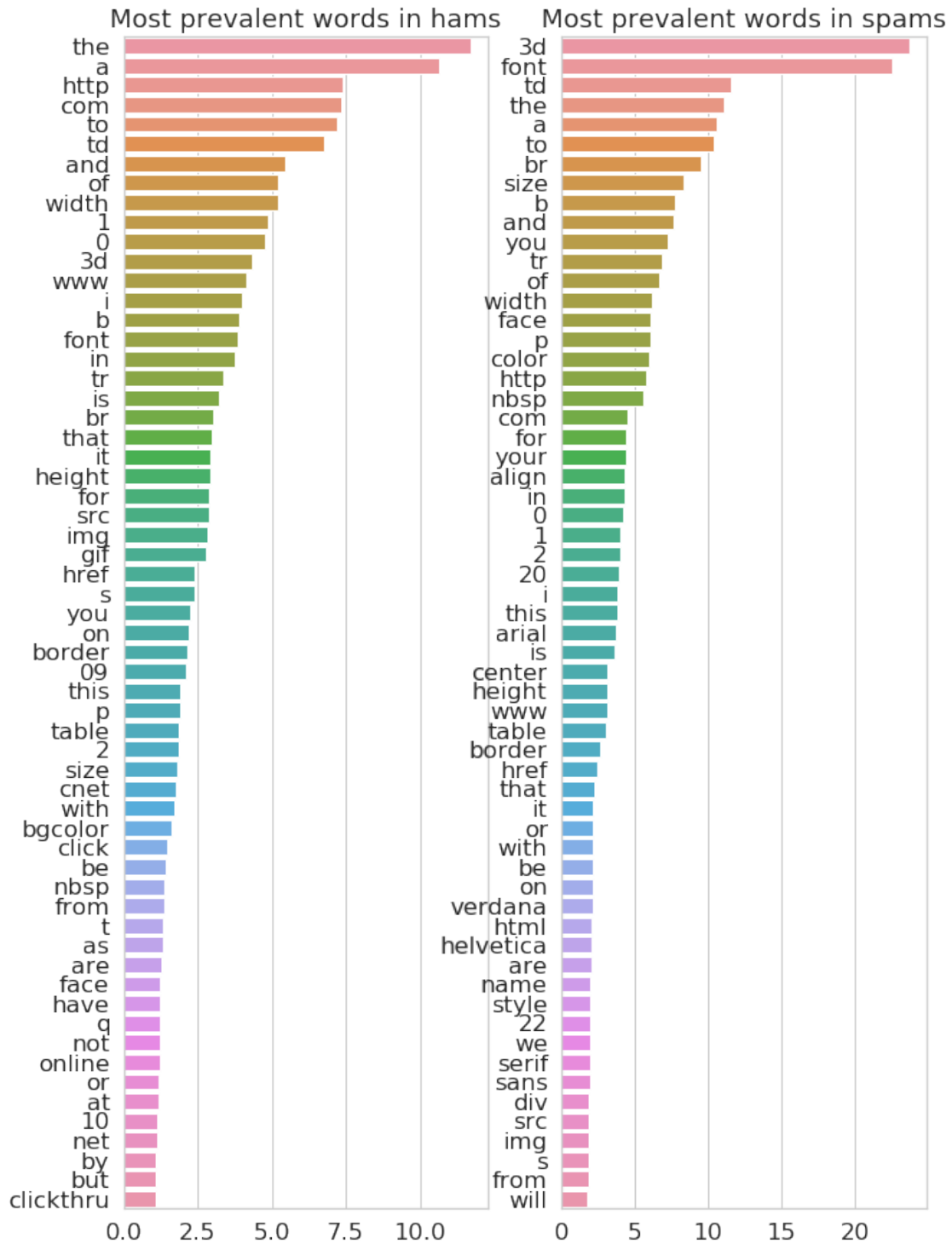
OFFER:
Ham: 348 || Spam: 603

```
[35]: import re
def word_counts(df):
    words_dict = {}
    for email in df['email']:
        word_list = re.findall('\w+', email)
        for w in word_list:
            words_dict[w] = (words_dict[w] + 1) if w in words_dict else 1
    return words_dict

#Series of word according to its descending order in spams and hams to sort it
spam_count_des = (pd.Series(word_counts(spams)) / spams.shape[0]).
    ↪sort_values(ascending=False)
ham_count_des = (pd.Series(word_counts(hams)) / hams.shape[0]).
    ↪sort_values(ascending=False)

#Make 2 columns
fig, axs = plt.subplots(ncols=2)
fig.set_size_inches(10,15)
#Only takes the head 60 words because it took too long to load more and get
    ↪better values.
spam_bar = sns.barplot(x=spam_count_des.head(60).values, y=spam_count_des.
    ↪head(60).index, ax=axs[1])
spam_bar.set_title("Most prevalent words in spams")
hams_bar = sns.barplot(x=ham_count_des.head(60).values, y=ham_count_des.
    ↪head(60).index, ax=axs[0])
hams_bar.set_title("Most prevalent words in hams")

train_word_count = (pd.Series(word_counts(train)) / train.shape[0]).
    ↪sort_values(ascending=False)[:300]
```



1.7.1 Question 7: Feature/Model Selection Process

In the 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?

Answer:

- 1) I tried to design and plot all the value counts (densities) of all features suggested above (beginning of part II). This helps me to get an idea of what might be good to do for improving my model.
- 2) I plotted many features and compared the densities, proportion, and distribution of the features in hams and spam emails to get the ideas of what are good to do.
- 3) Surprising search:
 - Firstly, the word '3d' and 'font' appear the most in spams, which is very surprising to me since they are not common words I see everyday, and they are not a html tag nor a common word. I actually checked my Spam emails and did not see these words much, but it's fun to explore though.
 - Secondly, punctuations such as '!' works a lot better than most other words!
 - Lastly, I am surprised that hams have more World Wide Internet Domains (i.e http,org,net,com, etc.) than spams. I usually think spammers usually want to maliciously get people into doing something bad such as clicking into a link.

```
[36]: from sklearn.linear_model import LogisticRegressionCV
def process_data(df):
    desired_words = ['congratulations', 'message', 'fyi', 'ham',
                     'spam', 'account', 'become', 'hi', 'dear',
                     'good', 'professional', 'best', 'cash', 'sorry',
                     'miss'] + train_word_count.index.tolist()
    X_train = np.array(words_in_texts(desired_words, df['email'])).astype(int)
    features = pd.concat([
        #Here we chose the func that has most differences between hams and spams
        #subject_words(df),
        num_punc(df),
        #num_html(df),
        num_html1(df),
        #num_html2(df),
        #num_body_words(df),
        is_rep(df),
        is_forward(df),
        has_link_http(df),
        has_link_org(df),
        has_link_net(df),
```

```

        has_link_com(df),
        has_free(df),
        has_offer(df)], axis=1).values
X_train = np.concatenate((X_train, features), axis=1)
return X_train

X_train = process_data(train)
Y_train = train['spam']

model = LogisticRegressionCV(fit_intercept=True)
model.fit(X_train, Y_train)
accuracy = model.score(X_train, Y_train)
print("Accuracy: ", accuracy)

```

```

/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```

/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```

/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```
/srv/conda/envs/data100/lib/python3.7/site-  
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed  
to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```
/srv/conda/envs/data100/lib/python3.7/site-  
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed  
to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```
/srv/conda/envs/data100/lib/python3.7/site-  
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed  
to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```
/srv/conda/envs/data100/lib/python3.7/site-  
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed  
to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```
/srv/conda/envs/data100/lib/python3.7/site-  
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed  
to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:

```
https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

```
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

```
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

```
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

```
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
```

```
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```


Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
regression
    extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
    extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
    extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
    extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
    extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
```

```
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/srv/conda/envs/data100/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Accuracy: 0.9932117662717955

/srv/conda/envs/data100/lib/python3.7/site-

```
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

```
[37]: def add_cols_rm_cols(df):
      df['word_c'] = subject_words(df)
      df['punc'] = num_punc(df)
      df['html'] = num_html1(df)
      df['body_c'] = num_body_words(df)
      df['rep'] = is_rep(df)
      df['forward'] = is_forward(df)
      df['http'] = has_link_http(df)
      df['org'] = has_link_org(df)
      df['net'] = has_link_net(df)
      df['com'] = has_link_com(df)
      df['offer'] = has_offer(df)
      df['free'] = has_free(df)
      df.drop(columns=['id', 'spam'], inplace=True)
      return df

hams = add_cols_rm_cols(hams)
spams = add_cols_rm_cols(spams)
h_corr = hams.corr()
s_corr = spams.corr()
```

```
/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:2:
```

SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using `.loc[row_indexer,col_indexer] = value` instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:3:
```

SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using `.loc[row_indexer,col_indexer] = value` instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

This is separate from the ipykernel package so we can avoid doing imports

```

until
/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
    after removing the cwd from sys.path.
/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
    """
/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
    import sys
/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

```

```

    if __name__ == '__main__':
/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
    # Remove the CWD from sys.path while we load stuff.
/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:11:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
    # This is added back by InteractiveShellApp.init_path()
/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:12:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
    if sys.path[0] == '':
/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:13:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
    del sys.path[0]
/srv/conda/envs/data100/lib/python3.7/site-packages/pandas/core/frame.py:4117:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
    errors=errors,

```

1.7.2 Question 8: EDA

In the cell below, show a visualization that you used to select features for your model. Include

1. A plot showing something meaningful about the data that helped you during feature selection, model selection, or both.

2. Two or three sentences describing what you plotted and its implications with respect to your features.

Feel to create as many plots as you want in your process of feature selection, but select one for the response cell below.

You should not just produce an identical visualization to question 3. Specifically, don't show us a bar chart of proportions, or a one-dimensional class-conditional density plot. Any other plot is acceptable, as long as it comes with thoughtful commentary. Here are some ideas:

1. Consider the correlation between multiple features (look up correlation plots and `sns.heatmap`).
2. Try to show redundancy in a group of features (e.g. `body` and `html` might co-occur relatively frequently, or you might be able to design a feature that captures all html tags and compare it to these).
3. Visualize which words have high or low values for some useful statistic.
4. Visually depict whether spam emails tend to be wordier (in some sense) than ham emails.

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

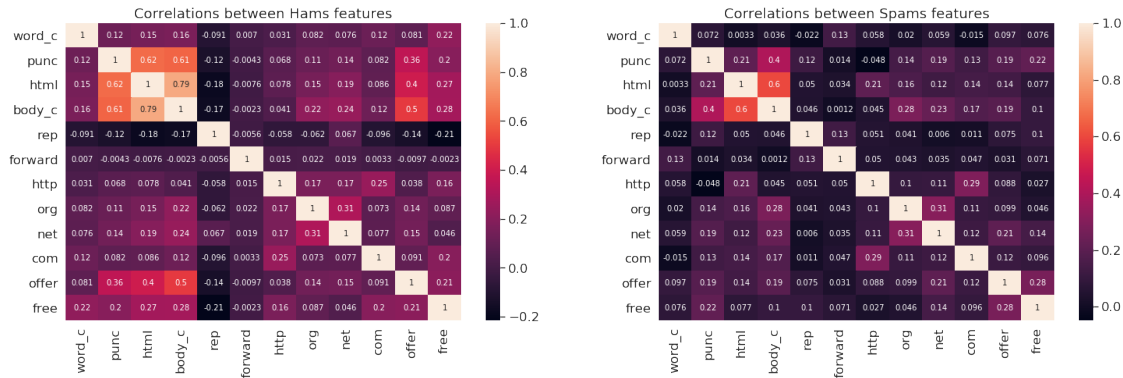
```
[38]: # Write your description (2-3 sentences) as a comment here:
# Below are my plots. It focuses on the correlation of all feature selections.
# Using this colorful
# and also helpful plots, I can see clearly which features are useful for my
# modelling selection.
# For example, all the World Wide Internet Domains (com, net, org, etc.) have
# very high correlations
# in hams, but not in spams, which surprised me alot. !!

# Write code to generate your visualization here:
fig, axs = plt.subplots(ncols=2)
fig.set_size_inches(25,7)
#Only takes the head 60 words because it took too long to load more and get
# better values.
hams_heat = sns.heatmap(h_corr, annot=True, ax =axs[0])
hams_heat.set_title("Correlations between Hams features")

spams_heat = sns.heatmap(s_corr, annot=True, ax =axs[1])
spams_heat.set_title("Correlations between Spams features")

# Note: if your plot doesn't appear in the PDF, you should try uncommenting the
# following line:
# plt.show()
```

```
[38]: Text(0.5, 1, 'Correlations between Spams features')
```



2 Question 9: Submitting to Kaggle

The following code will write your predictions on the test dataset to a CSV, which you can submit to Kaggle. You may need to modify it to suit your needs.

Save your predictions in a 1-dimensional array called `test_predictions`. *Even if you are not submitting to Kaggle, please make sure you've saved your predictions to `test_predictions` as this is how your score for this question will be determined.*

Remember that if you've performed transformations or featurization on the training data, you must also perform the same transformations on the test data in order to make predictions. For example, if you've created features for the words “drug” and “money” on the training data, you must also extract the same features in order to use scikit-learn's `.predict(...)` method.

You should submit your CSV files to <https://www.kaggle.com/t/c76d80f7d3204159865a324ec2936f18>

Note: You may submit up to 4 times a day. If you have submitted 4 times on a day, you will need to wait until the next day for more submissions.

Note that this question is graded on an absolute scale based on the accuracy your model achieves on the test set and the score does not depend on your ranking on Kaggle.

The provided tests check that your predictions are in the correct format, but you must submit to Kaggle to evaluate your classifier accuracy.

```
[39]: X_test = process_data(test)
      test_predictions = model.predict(X_test)
      test_predictions
```

```
[39]: array([0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
            0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
            0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
            0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
```

```

0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0,
1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0,
1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1])

```

```
[40]: ok.grade("q9");
```

```

~~~~~
Running tests
-----

```

Test summary
Passed: 3
Failed: 0
[ooooooooook] 100.0% passed

The following saves a file to submit to Kaggle.

```
[41]: from datetime import datetime

# Assuming that your predictions on the test set are stored in a 1-dimensional
# array called
# test_predictions. Feel free to modify this cell as long you create a CSV in
# the right format.

# Construct and save the submission:
submission_df = pd.DataFrame({
    "Id": test['id'],
    "Class": test_predictions,
}, columns=['Id', 'Class'])
timestamp = datetime.isoformat(datetime.now()).split(".")[0]
submission_df.to_csv("submission_{}.csv".format(timestamp), index=False)

print('Created a CSV file: {}'.format("submission_{}.csv".format(timestamp)))
print('You may now upload this CSV file to Kaggle for scoring.')
```

Created a CSV file: submission_2020-04-27T15:07:19.csv.
You may now upload this CSV file to Kaggle for scoring.

3 Submit

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. **Please save before submitting!**

```
[ ]: # Save your notebook first, then run this cell to submit.
import jassign.to_pdf
jassign.to_pdf.generate_pdf('proj2b.ipynb', 'proj2b.pdf')
ok.submit()
```

Generating PDF...
Saved proj2b.pdf
<IPython.core.display.Javascript object>

[]: