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.

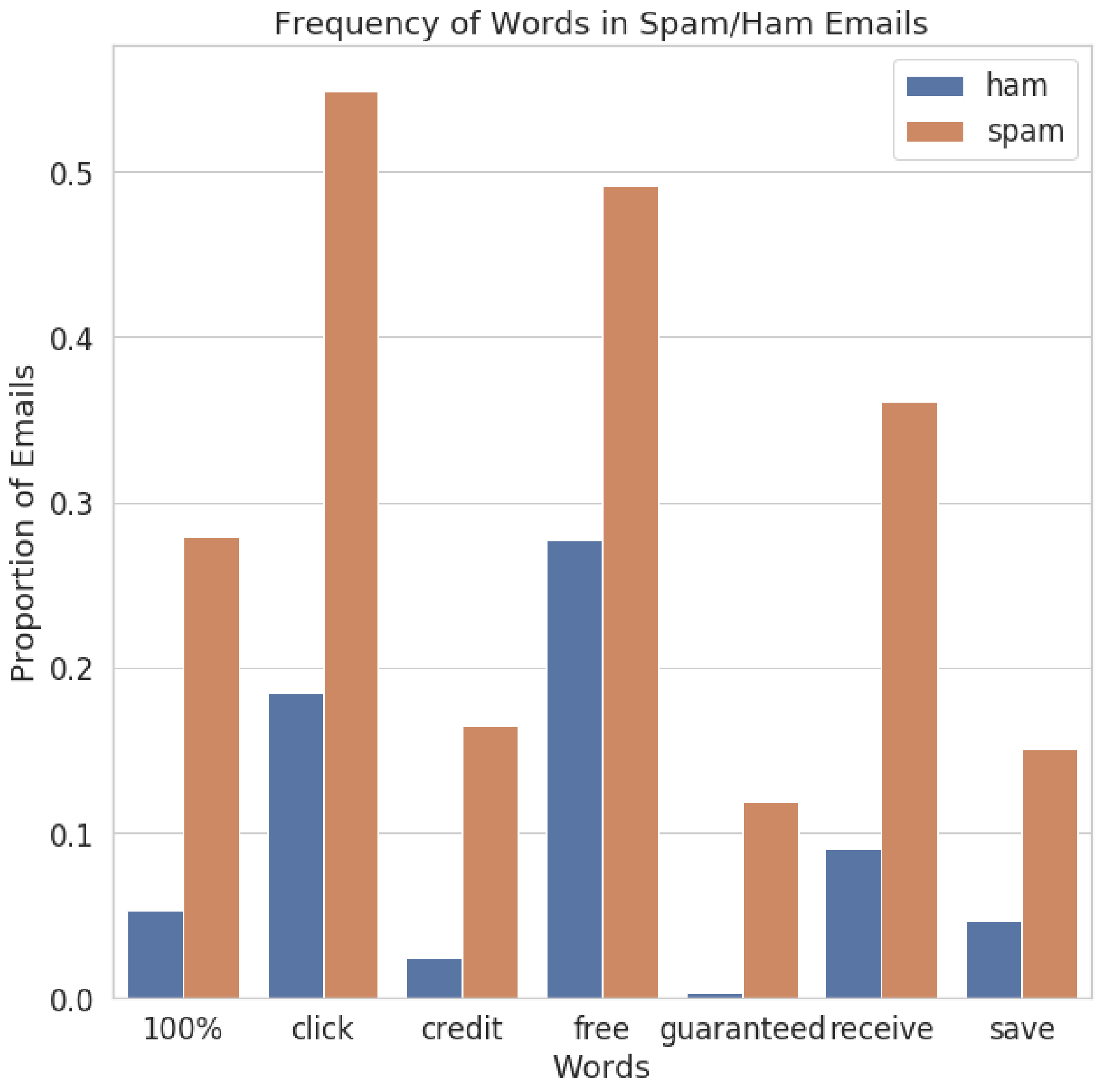
The spam email has lots of HTML tags and the normal email does not. We can use this to to distinguish spam.

0.0.2 Question 3a

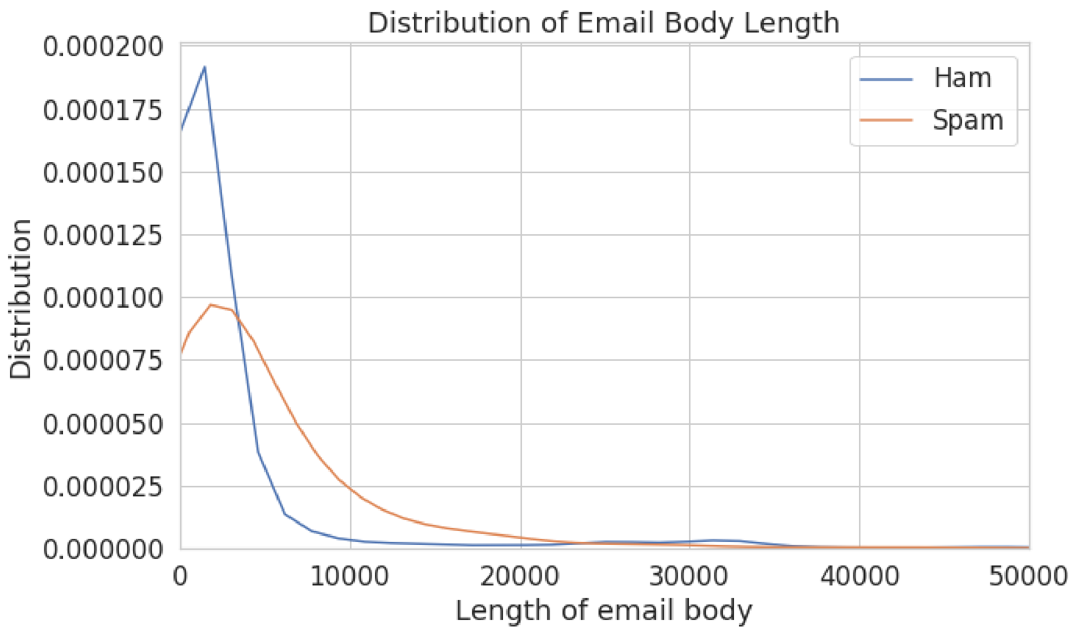
Create a bar chart like the one above comparing the proportion of spam and ham emails containing certain words. Choose a set of words that are different from the ones above, but also have different proportions for the two classes. Make sure to only consider emails from `train`.

```
In [13]: train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai

         words = ['click', 'free', 'receive', 'credit', 'save', 'guaranteed', '100%']
         temp = pd.DataFrame(data = words_in_texts(words, train['email']), columns = words)
         temp['spam'] = train['spam'].replace({0: 'ham', 1: 'spam'})
         melt = temp.melt('spam').groupby(['spam', 'variable']).mean().reset_index()
         plt.figure(figsize=(10,10))
         sns.barplot(data = melt , x = 'variable', y = 'value', hue = 'spam')
         plt.xlabel("Words")
         plt.ylabel("Proportion of Emails")
         plt.title("Frequency of Words in Spam/Ham Emails")
         plt.legend()
```

```
Out[13]: <matplotlib.legend.Legend at 0x7f6dc29ccee0>
```

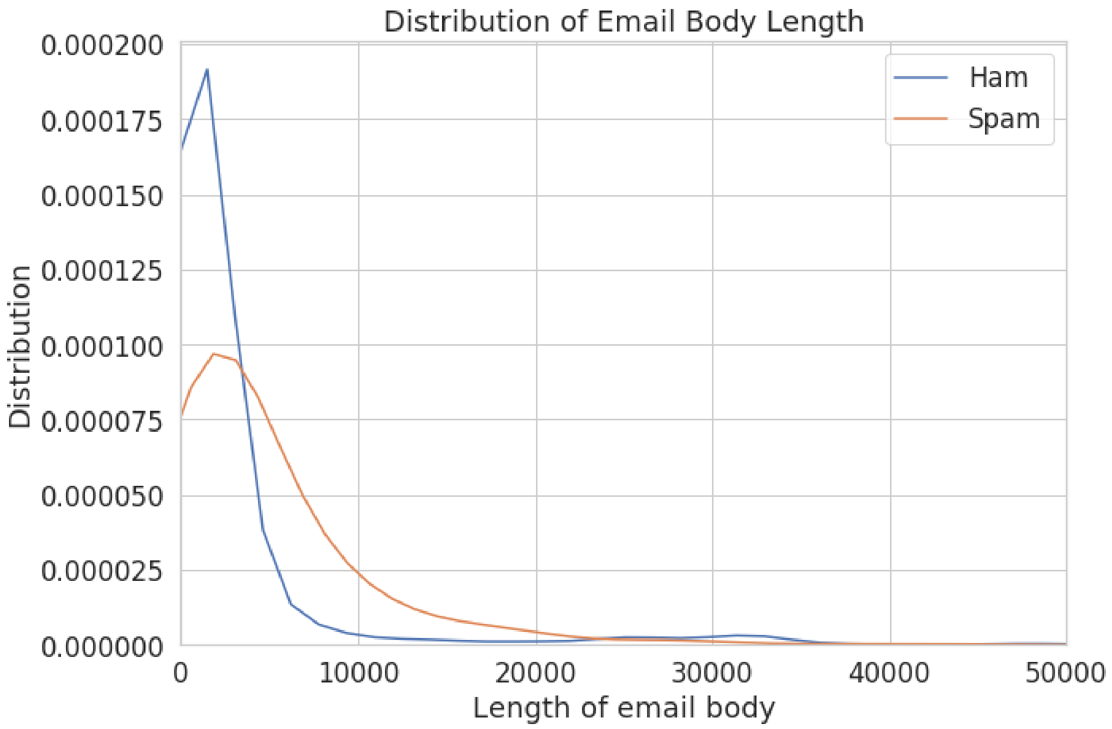

Frequency of Words in Spam/Ham Emails
ham
spam
Proportion of Emails
0.0
0.1
0.2
0.3
0.4
0.5
100%
click
credit
free
guaranteed
receive
save
Words

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 [14]: temp_spam = train[train['spam'] == 0]['email'].agg(len)
         temp_ham = train[train['spam'] == 1]['email'].agg(len)
         plt.figure(figsize=(10,7))
         sns.distplot(temp_spam, label='Ham', hist=False)
         sns.distplot(temp_ham, label='Spam', hist=False)
         plt.xlim((0,50000));
         plt.title('Distribution of Email Body Length')
         plt.xlabel('Length of email body')
         plt.ylabel('Distribution')
         plt.legend()
         plt.savefig('training_conditional_densities.png')
```


Distribution of Email Body Length
Distribution
Length of email body
Ham
Spam
0.000200
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
0
10000
20000
30000
40000
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)?

The false positive is 0 because we always predict email is ham (y_hat = 0), which means there will be no positive predictions. Thurs, fp=0. Fn is what we predict wrong, since we know fp,tp = 0 and y_hat always = 0, we can get that false negative is the number of total spam emails(spam = 1). We know recall = $\frac{\text{TP}}{\text{TP+FN}}$ and fp,tp = 0. Therefore, recall = 0. accuracy = $\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} = \frac{\text{TN}}{\text{TN+FN}} = 1 - \frac{\text{FN}}{\text{TN+FN}} = 1 - \frac{\text{FN}}{\text{total emails}}$ since total emails = TN + TP + FN + FP = TN + FN.

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 or false negatives because (1-accuracy) = (fn+fp)/(tp+tn+fp+fn) is bigger than the value in Question 5.

0.0.6 Question 6f

1. Our logistic regression classifier got 75.76% 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. Although the logistic regression classifier is more accurate, it is only 1% higher than the the zero predictor.
2. The selected words may be very common in a wide range of emails, which means that we cannot use these words to determine whether it is spam. This will reduce the accuracy of the model.
3. I would choose logistic regression classifier because it can filter spam(Recall > 0) . In contrast, zero cannot filter spam (Recall $= 0$) since Recall measures the proportion of spam emails that were correctly flagged as spam.

0.0.7 Question 7: Feature/Model Selection Process

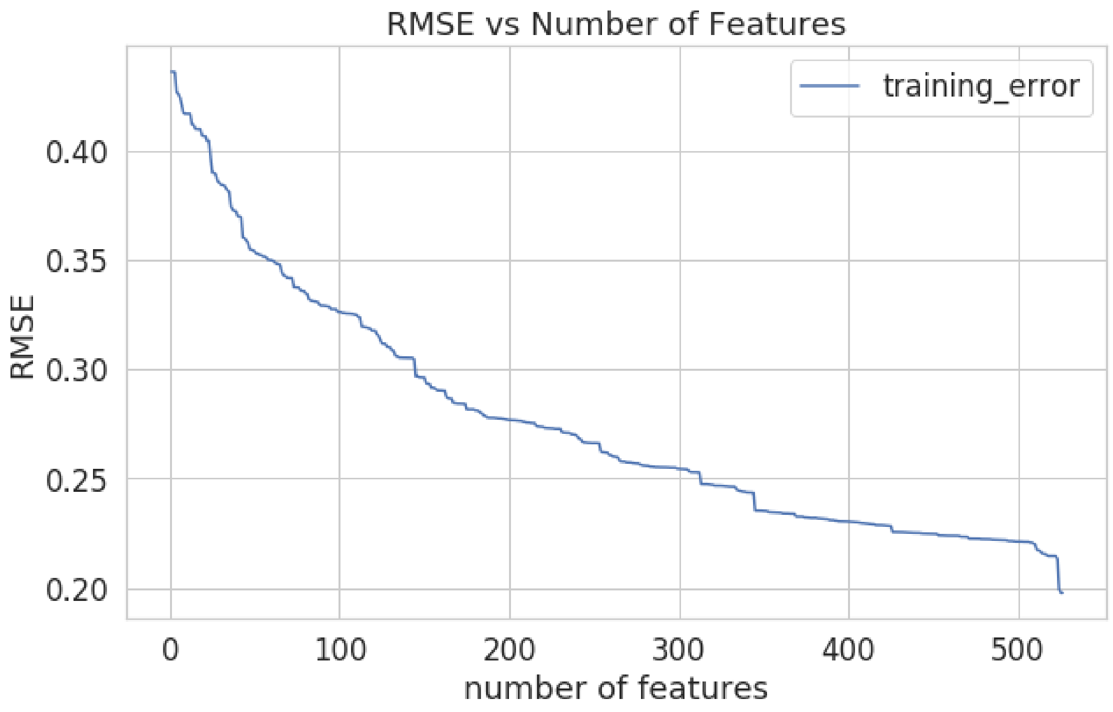
In this following cell, describe the process of improving your model. You should use at least 2-3 sentences each to address the follow questions:

1. How did you find better features for your model?
2. What did you try that worked or didn't work?
3. What was surprising in your search for good features?

1. I followed the above ideas to build my model. For the feature words, what I have adopted is to first obtain the high-frequency words of spam and ham, and then removing some of the high-frequency words of ham and adding the words obtained on Google from the high-frequency words of spam to form the feature words. In addition, through comparison, I found that the proportion of capital letters in the text and whether it is a forwarding and replying email is very useful for distinguishing spam.
2. I tried to calculate the number of words and letters in email and subject, it did not work but reduces the accuracy. I think it's because the length of the text has nothing to do with spam. Calculating how many exclamation marks and capital letters are in context is very useful for distinguishing whether it is spam. Normal emails will not have too many capital letters and exclamation points, but spam emails will use these to attract people.
3. What surprises me is that spam emails often use capital letters and exclamation points(!) to attract attention. Another is that spam emails use a lot of HTML tags, which is not found in normal emails. Spam emails will also try to use 'Re' or 'Fw' as the subject to pretend that they are not spam.

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

```
In [26]: # Write your description (2-3 sentences) as a comment here:
         # I want to see the relationship between the number of features and errors in my model.
         # This allows me to better judge how many features I need to in order to improve my model.
         #

         # Write the code to generate your visualization here:
         import sklearn.linear_model as lm
         def rmse(actual_y, predicted_y):
             return np.mean((actual_y - predicted_y)**2)**0.5
         linear_model = lm.LinearRegression()
         train_error_vs_N = []

         range_of_num_features = range(1, X_mytrain.shape[1] + 1)

         for N in range_of_num_features:
             X_train_first_N_features = X_mytrain.iloc[:, :N]

             linear_model.fit(X_train_first_N_features, Y_mytrain)
             train_error_overfit = rmse(Y_mytrain, linear_model.predict(X_train_first_N_features))
             train_error_vs_N.append(train_error_overfit)

         plt.figure(figsize=(10, 6))
         plt.plot(range_of_num_features, train_error_vs_N)
         plt.legend(["training_error"])
         plt.title('RMSE vs Number of Features')
         plt.xlabel("number of features")
         plt.ylabel("RMSE");
```


RMSE vs Number of Features
training_error
RMSE
0.40
0.35
0.30
0.25
0.20
0
100
200
300
400
500
number of features

0.0.8 Question 9: ROC Curve

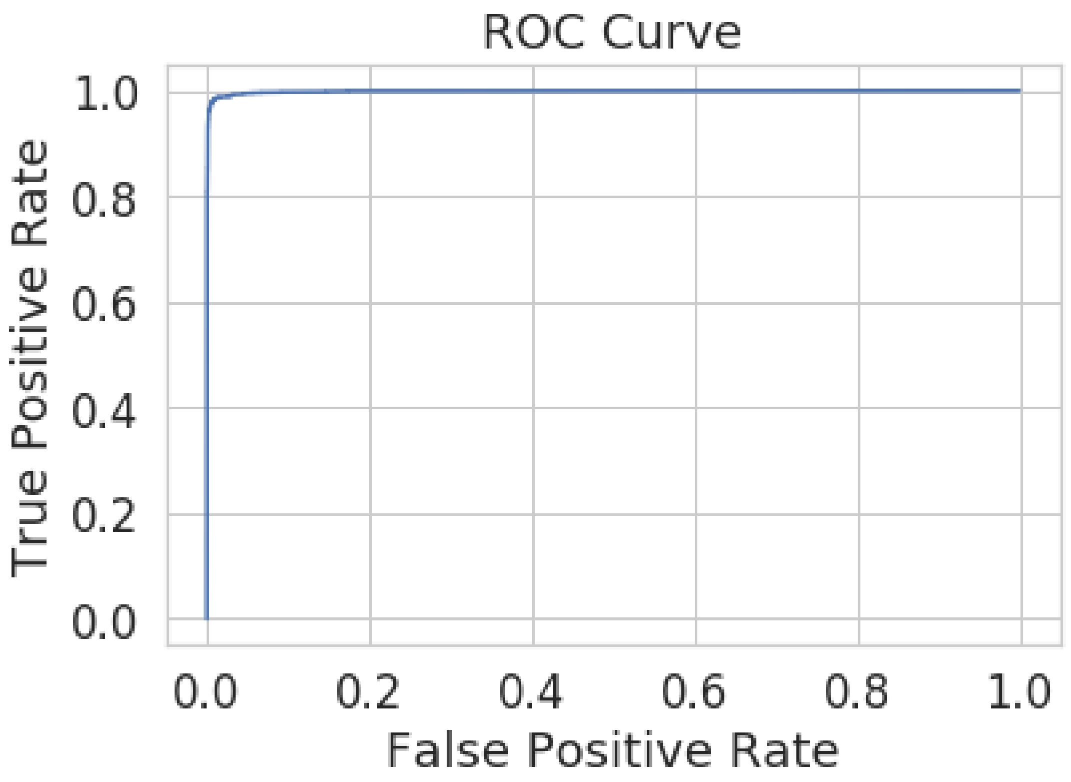
In most cases we won't be able to get 0 false positives and 0 false negatives, so we have to compromise. For example, in the case of cancer screenings, false negatives are comparatively worse than false positives — a false negative means that a patient might not discover that they have cancer until it's too late, whereas a patient can just receive another screening for a false positive.

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 [27]: 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

         fpr, tpr, threshold = roc_curve(Y_mytrain, model.predict_proba(X_mytrain)[:,1])
         plt.plot(fpr,tpr)
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.title('ROC Curve')
```

```
Out[27]: Text(0.5, 1.0, 'ROC Curve')
```


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