0.0.1 Question 1c

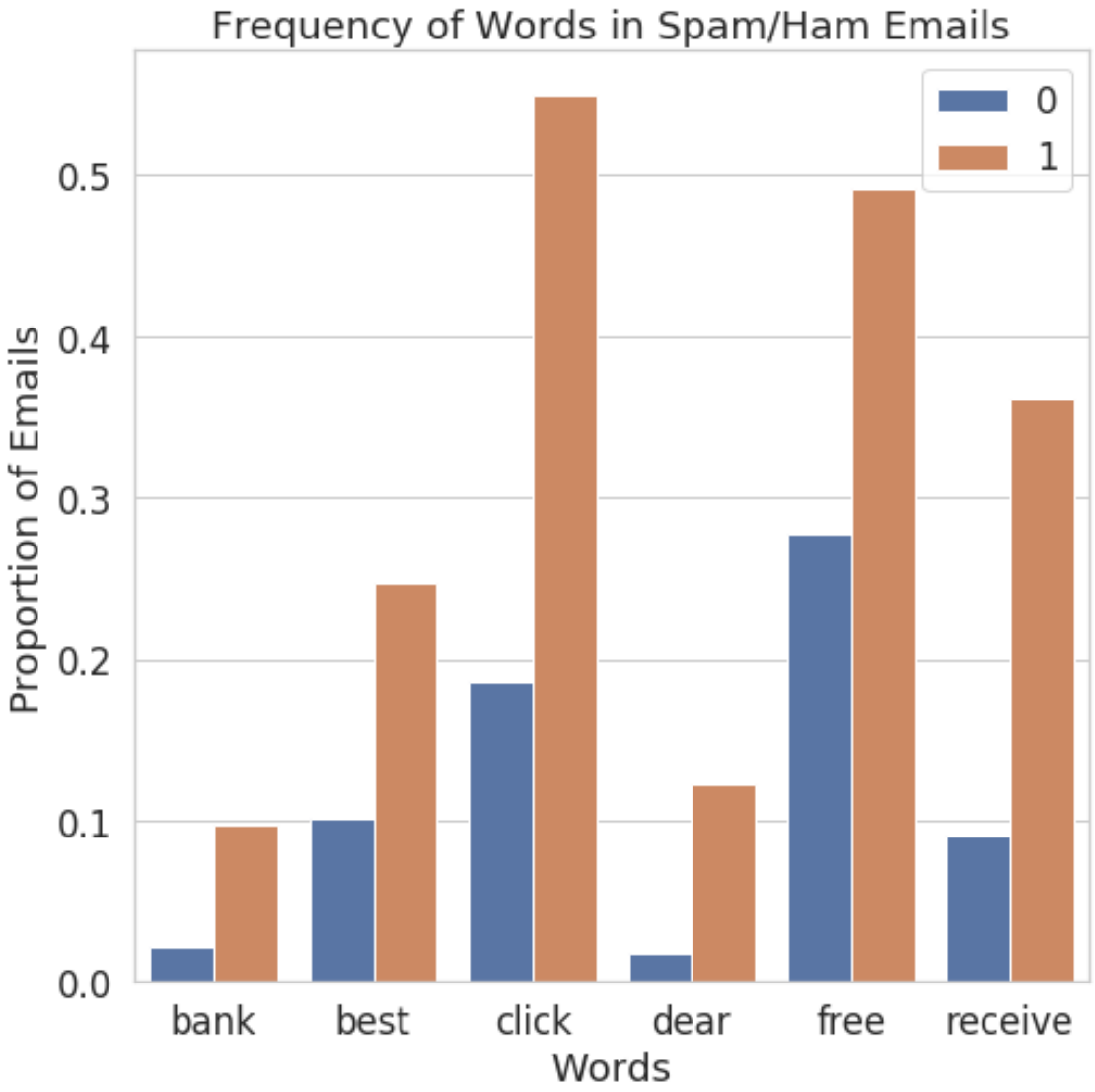
Discuss one thing you notice that is different between the two emails that might relate to the identification of spam.

The spam tag was written in HTML and uses a lot of HML tag. We know that a good indicator this is spam is due to the HTML code written.

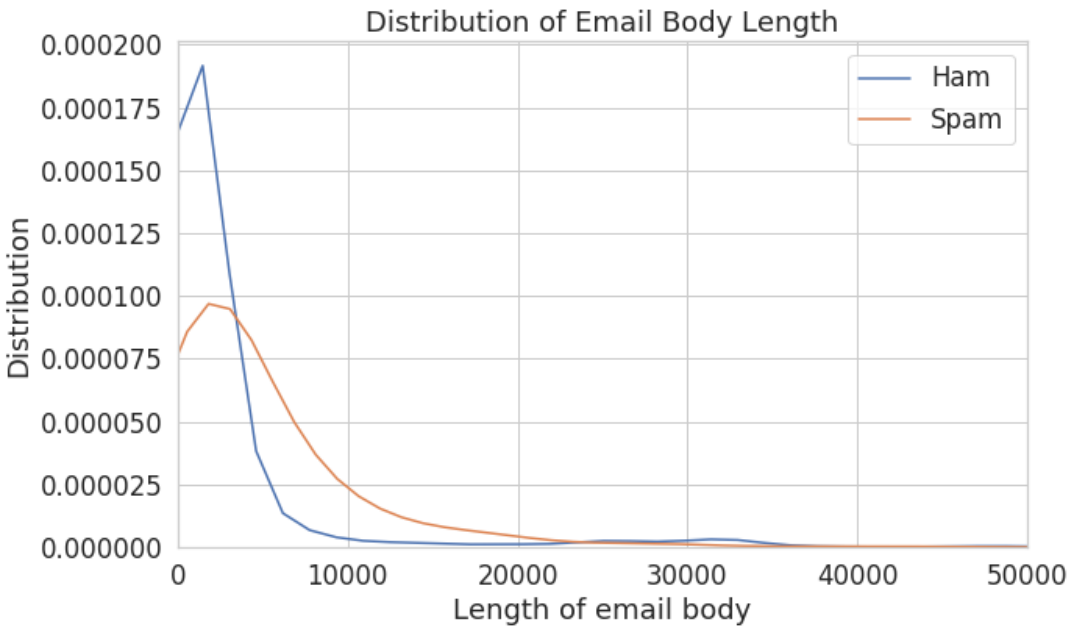
0.0.2 Question 3a

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 [65]:
```
train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai

words = ['bank', 'best', 'dear', 'click', 'free', 'receive']
wit = words_in_texts(words, train['email'])
df = pd.DataFrame(data = wit, columns = words)
df['spam'] = train['spam']
melt_data = df.replace({'label': {0:'Ham', 1:'Spam'}}).melt('spam').groupby(['spam', 'variable
plt.figure(figsize=(8,8))
sns.barplot(x='variable', y='value', hue='spam', data=melt_data)
plt.title('Frequency of Words in Spam/Ham Emails')
plt.xlabel('Words')
plt.ylabel('Proportion of Emails')
plt.legend(title='')
plt.show()
```

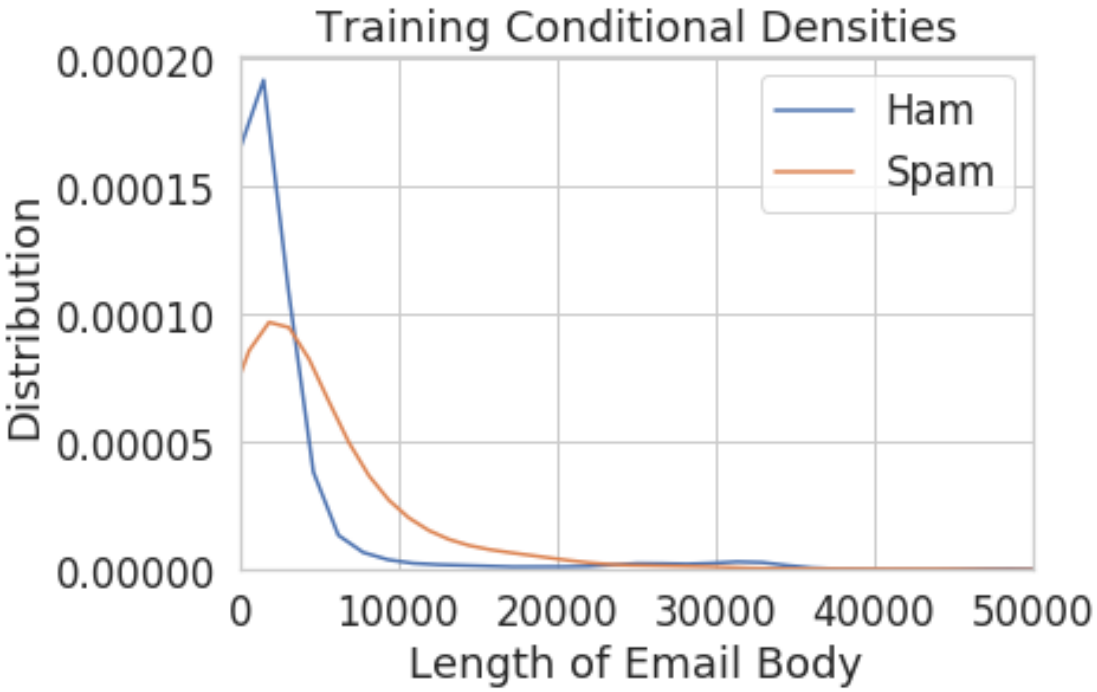

Frequency of Words in Spam/Ham Emails
0
1
0.5
0.4
0.3
0.2
0.1
0.0
Proportion of Emails
bank
best
click
dear
free
receive
Words

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 [66]: tmp = train.copy()
         tmp['length'] = tmp['email'].str.len()
         sns.distplot(tmp.loc[tmp['spam'] == 0, 'length'], hist=False, label='Ham')
         sns.distplot(tmp.loc[tmp['spam'] == 1, 'length'], hist=False, label='Spam')
         plt.xlabel("Length of Email Body")
         plt.ylabel("Distribution")
         plt.title("Training Conditional Densities")
         plt.xlim((0, 50000))
         plt.legend(title='')
         plt.savefig('training_conditional_densities.png')
```


Training Conditional Densities
Ham
Spam
Distribution
0.00020
0.00015
0.00010
0.00005
0.00000
0
10000
20000
30000
40000
50000
Length of Email Body

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)?

We observe a false positive value of 0 because our guesses are all 0 (predicting negative) therefore there cannot be any false positives if we do not have any predicting positives. Since all negatives are labeled 0 then the false neative value is the sum of the 1s in the training set. We observe a 74.47% accuracy due to the proportion of points we correctly classified and this aslo represents the vlaues labeled correctly. We get a recall of 0 because this is the measure of proportion of spam emails that were correctly labeled as spam. The numerator is the number of tru positives and since we classified all the points as negatives there will be 0 true positive values.

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 when we use the logistic regression classifier

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. When we predicted 0 for every email the prediction was 74.47% which slightly 1% worse than the logistic regression classifier of 75.76%

2. We can assume that the words that were chosen were probably not prevalent enough which means our model that is based off these words are less accurate. When the words are not in a lot of emails it is hard to differentiate if the characteristic of the word is Spam or Ham mail thus affecting the accuracy of the model.

3. Even though the amount of false positives for the logistic regression classifier are low, we would want the value to be 0 because the false positives might inorrectly classify ham emails as spam. We definately do not want this as this would be trashing important emails. Therefore we would want the classifier that marks the unknown as ham which would then mean it is up to the user to then redirect those emails but not miss any important mail!

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:

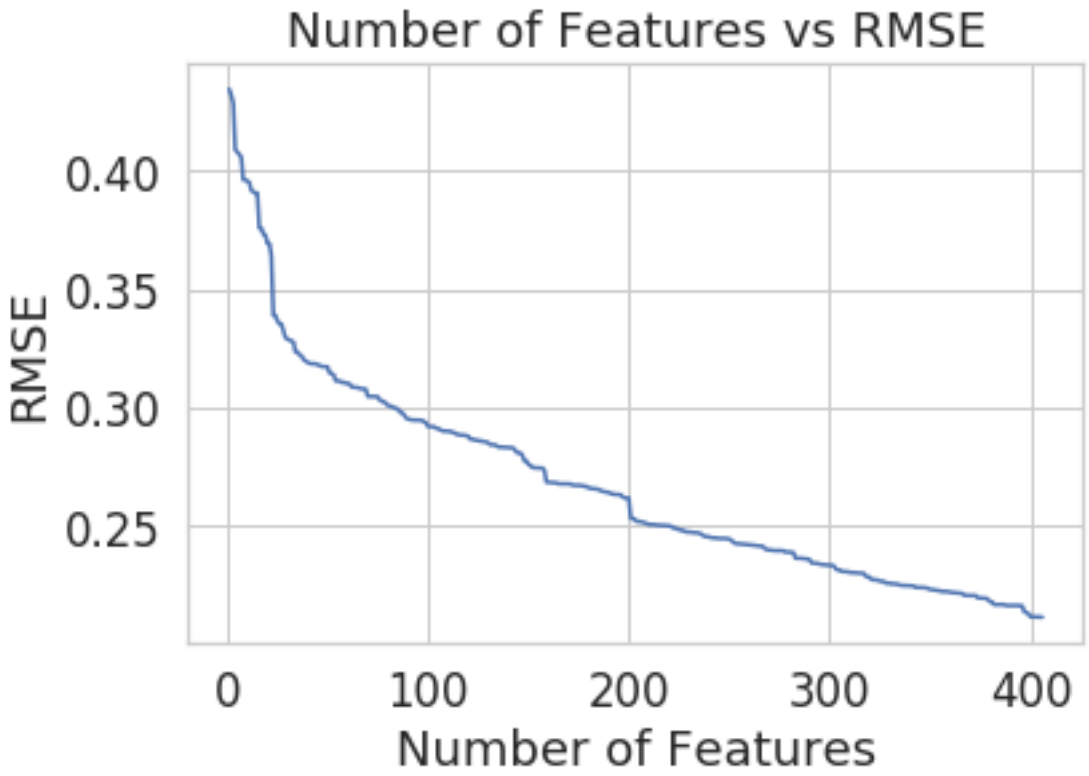
1. How did you find better features for your model?
2. What did you try that worked or didn't work?
3. What was surprising in your search for good features?

1. I first found the words that were the most common in spam emails and not ham emails so there wont be any overlap. Then I combined the list of words with the high frequency method from Q3. Then I used a validation method ot try the different combination of words at the top of spam words and then saw that the accuracy increased with more words. Also I noticed that spam emails generally used more capital letters than the ham emails so a counter was made to capture this as a feature
2. I first tried to search on google what the most common language is in spam emails. I cross-compared this list to the features in the matrix but it wasnt very successful as the accuracy was very low.
3. I wasnt expecting that there would be a difference in capitalization in spam emails and ham emails but also I was surprised to see that there was a combination of letters and numbers and words that dont exist frequently found in spam emails

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

In [80]:
```
# Write your description (2-3 sentences) as a comment here:
# I found that the number of features was important in determining the biggest facor. The mode
# different numbers of features and then plotted agains the RMSE error. The visuaization shows
# of features increase the RMSE decreases.

# Write the code to generate your visualization here:
import sklearn.linear_model as lm
linear_model = lm.LinearRegression()
train_error_vs_N = []

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

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

    linear_model.fit(X_train_first_N_features, train_data['spam'])
    train_error_overfit = rmse(train_data['spam'], linear_model.predict(X_train_first_N_featur
    train_error_vs_N.append(train_error_overfit)

plt.plot(range_of_num_features, train_error_vs_N)
plt.title('Number of Features vs RMSE')
plt.xlabel('Number of Features')
plt.ylabel("RMSE")
```

Out[80]:
```
Text(0, 0.5, 'RMSE')
```


Number of Features vs RMSE
RMSE
0.40
0.35
0.30
0.25
0
100
200
300
400
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 [ ]:
```
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

model = LogisticRegression().fit(xtrain_clean, train_data['spam'])
y_hat = model.predict_proba(xtrain_clean)[:, 1]

fpr, tpr, thresholds = roc_curve(train_data['spam'], y_hat)
with sns.axes_style('white'):
    plt.plot(fpr, tpr)

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
```