0.0.1 Question 1c

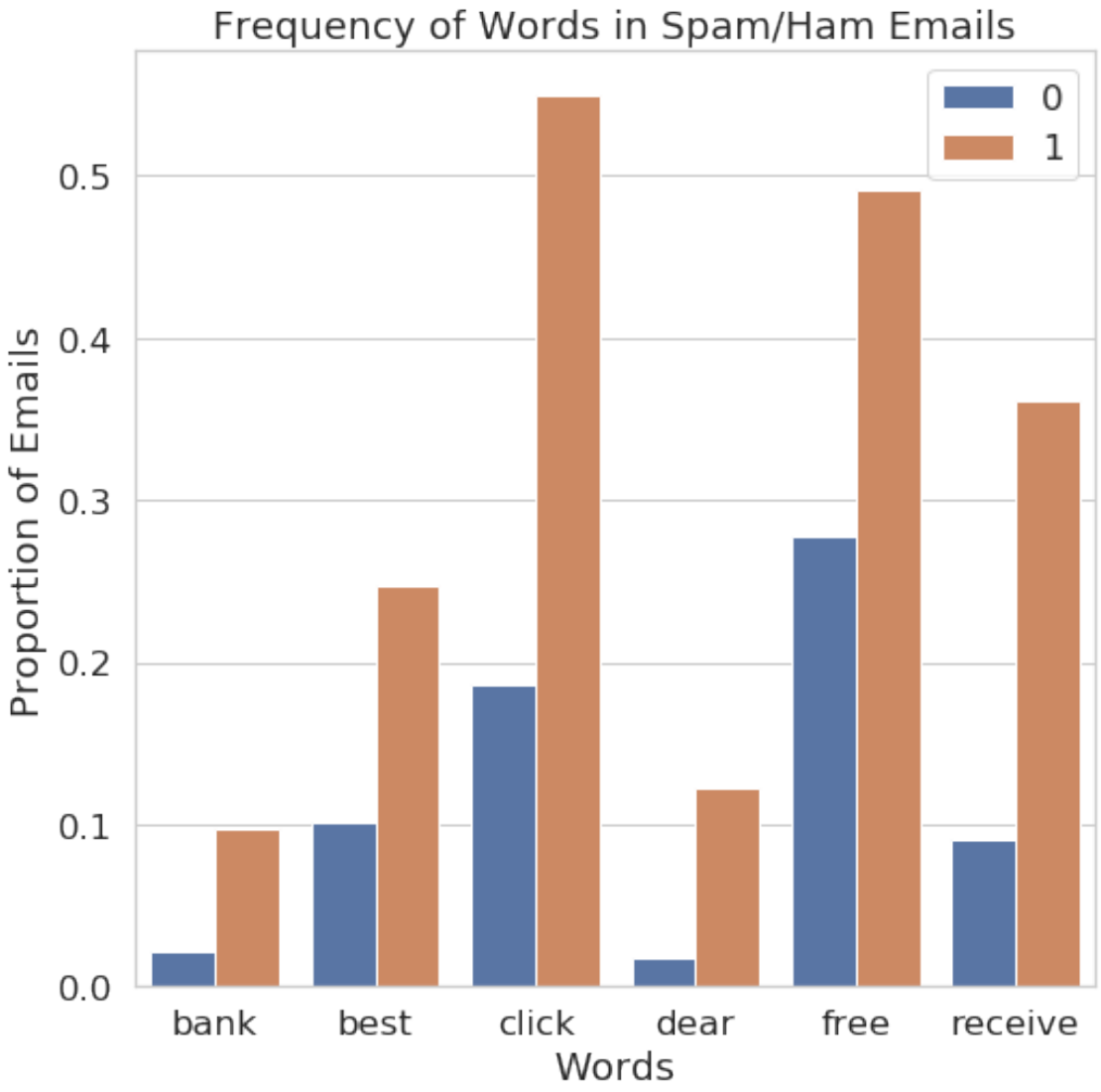
Discuss one thing you notice that is different between the two emails that might relate to the identification of spam.

The spam tag looks like it was written in HTML and has a lot of HTML tags. Emails written in HTML may be an indication that it is 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`.

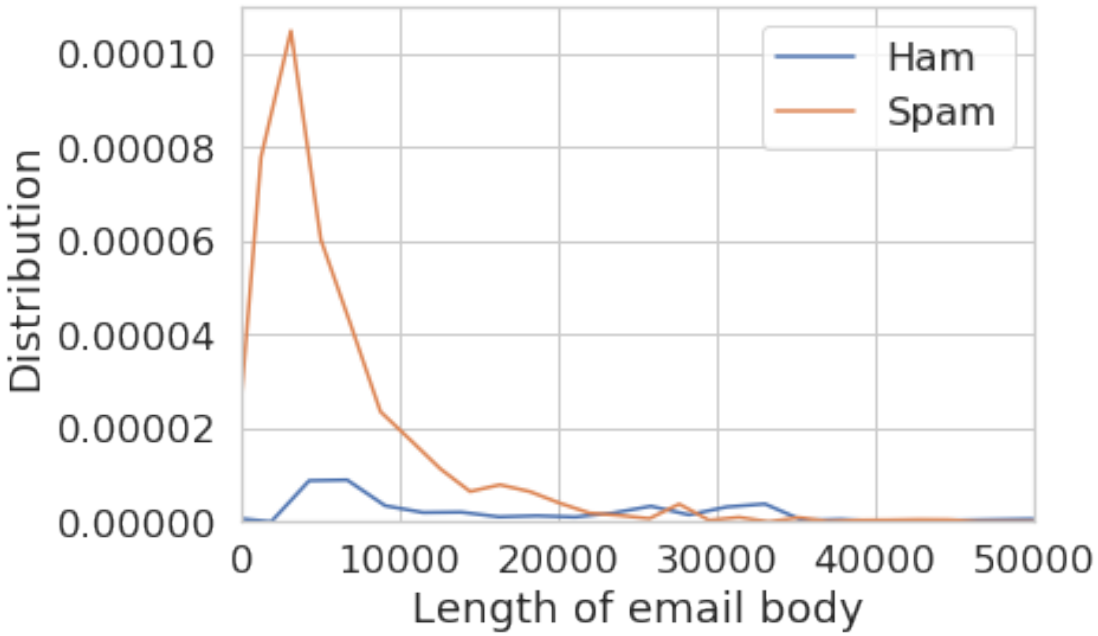
In [12]:
```
train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai

words = ['click', 'dear', 'receive', 'bank', 'best', 'free']
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', 'varia
plt.figure(figsize=(8,8))
sns.barplot(x = "variable", y = "value", hue = "spam", data = melt_data)
plt.xlabel('Words')
plt.ylabel('Proportion of Emails')
plt.legend(title = "")
plt.title("Frequency of Words in Spam/Ham Emails")
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 [13]: 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.xlim((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)?

We get a false positive value of 0 because our guess are all 0 (predicting negative) so there can't be any false positives if there are no predicted positive values. Since everything is labeled a 0 (negative), the false negative value is thus the sum of all the 1's in the training set because these are the values that are truly positive, making the predicted 0 a false negative. We observe a 74% accuracy because accuracy is the proportion of points we correctly classified, so out of the whole set, 74% of the values were labeled correctly (actual value in training set is 0). We get a recall of 0 because recall measures the proportion of spam emails that were correctly flagged as spam; its numerator is the number of true positives but since we classified all of the points as negative, there are 0 true positive values because we didn't classify anything as positive to begin with.

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.

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. This is only a little more than 1% better than the prediction accuracy of predicting 0 for every email, which was 74.5%.
2. The words that were chosen were probably not prevalent in enough emails and thus basing our model off of these words makes the model less accurate. If these words were not in a lot of emails, it becomes difficult to distinguish whether a certain word is characteristic of a spam or ham email, affecting the accuracy of the model.
3. Although the amount of false positives for the logistic regression classifier is pretty low, ideally, we would want this value to be 0 because false positives might incorrectly ham emails as spam, which is especially dangerous for important emails. I would thus prefer the classifier that marks everything ham because I'd rather deal with a flooded imbox than miss any important emails.

0.0.7 Question 7: Feature/Model Selection Process

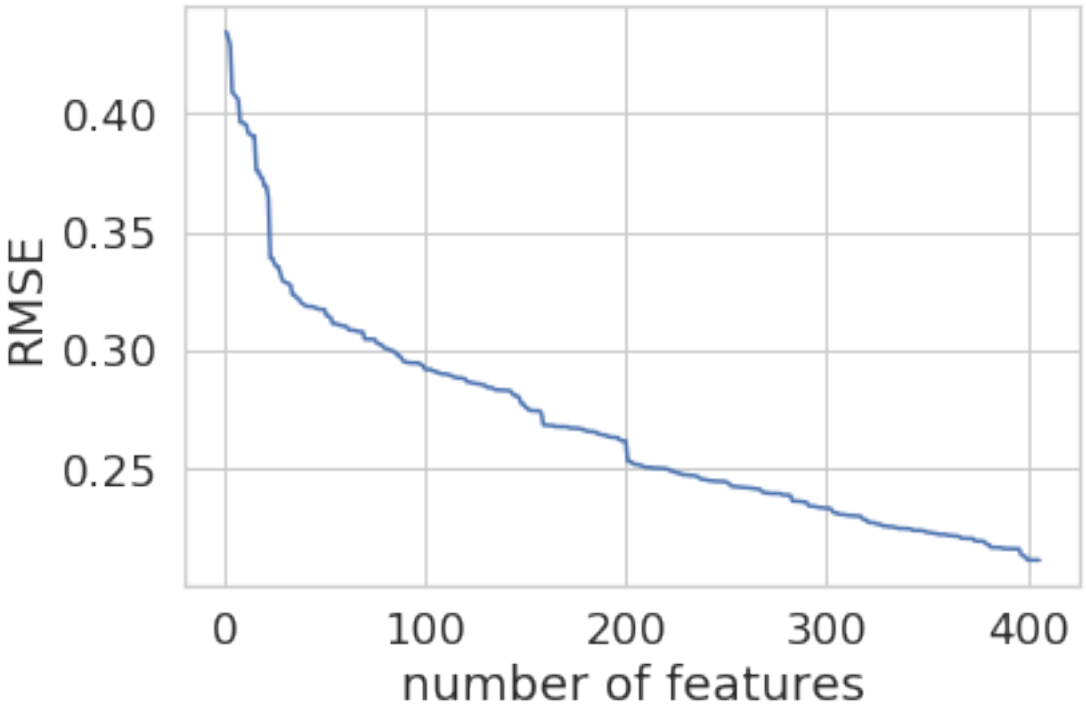
In this following cell, describe the process of improving your model. You should use at least 2-3 sentences each to address the follow questions:

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. The first way was to find the words in spam emails that occur at the highest frequency but do not occur in ham emails. This method is to figure out what are the most common words in spam emails that do not appear in ham emails. I then combined this list with a list of words I found to have high frequency using the method from Q3. I then used the validation method to try with different amount of top spam words and saw that accuracy improved with the more words I used. Additionally, when looking through the data, I noticed that spam emails generally have more capital letters so I made a counter for the amount of capital letters in an email and added that as a feature.
2. The first thing I tried was google common words found in spam emails and use those as features in my matrix but my accuracy was fairly low. This potentially could have resulted from the uniqueness in this dataset.
3. When I looked into my dictionary of most frequently used spam words, I found words that I wouldn't have expected to be words found frequently in spam emails. There were words that were a combination of letters and numbers and not actual words. Additionally, I didn't expect there to be a difference in the amount of capital letters between spam and ham emails.

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

In [27]:
```
# Write your description (2-3 sentences) as a comment here:
# The biggest factor in my feature selection was determining the number of features. Thus, I r
# numbers of features and plotted that against the RSME of the model. As the visualization sho
# features increase, the rsme decreases so I decided to go with a model that incorporated more

# 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.xlabel("number of features")
plt.ylabel("RMSE");
```



0.0.8 Question 9: ROC Curve

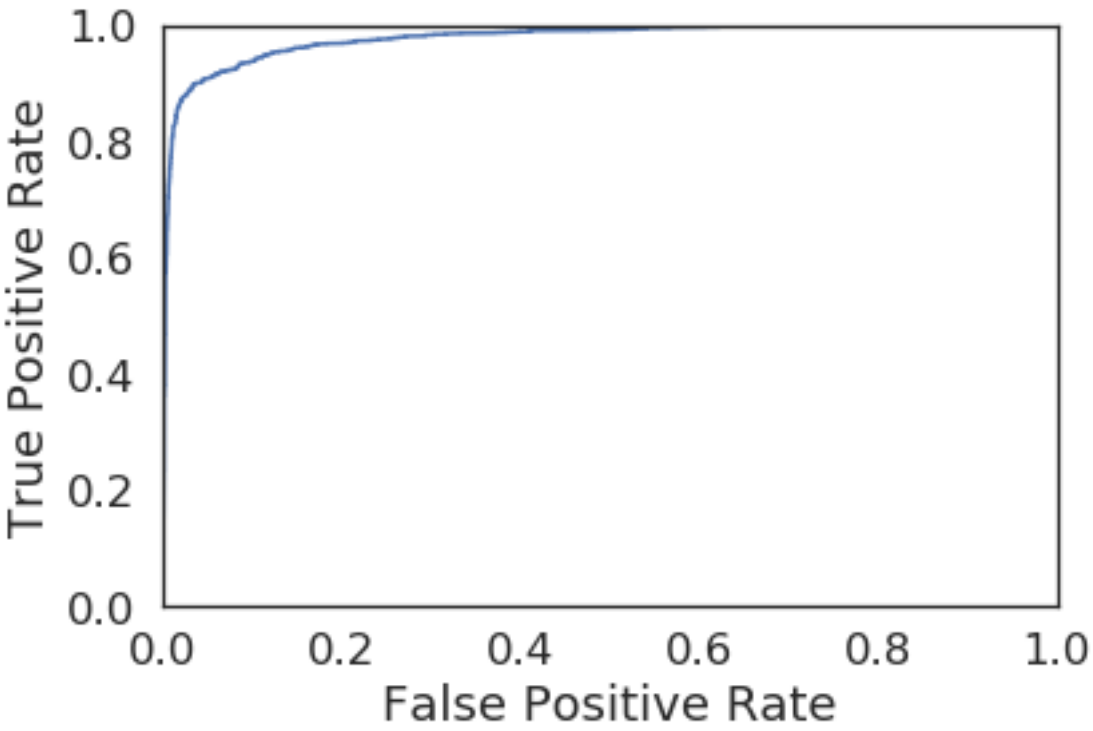
In most cases we won't be able to get no false positives and no false negatives, so we have to compromise. For example, in the case of cancer screenings, false negatives are comparatively worse than false positives — a false negative means that a patient might not discover a disease until it's too late 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 [28]:
```
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.xlim([0, 1])
plt.ylim([0, 1])
plt.show()
```

```
/srv/conda/envs/data100/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWa
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)
```


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