0.0.1 Question 1c

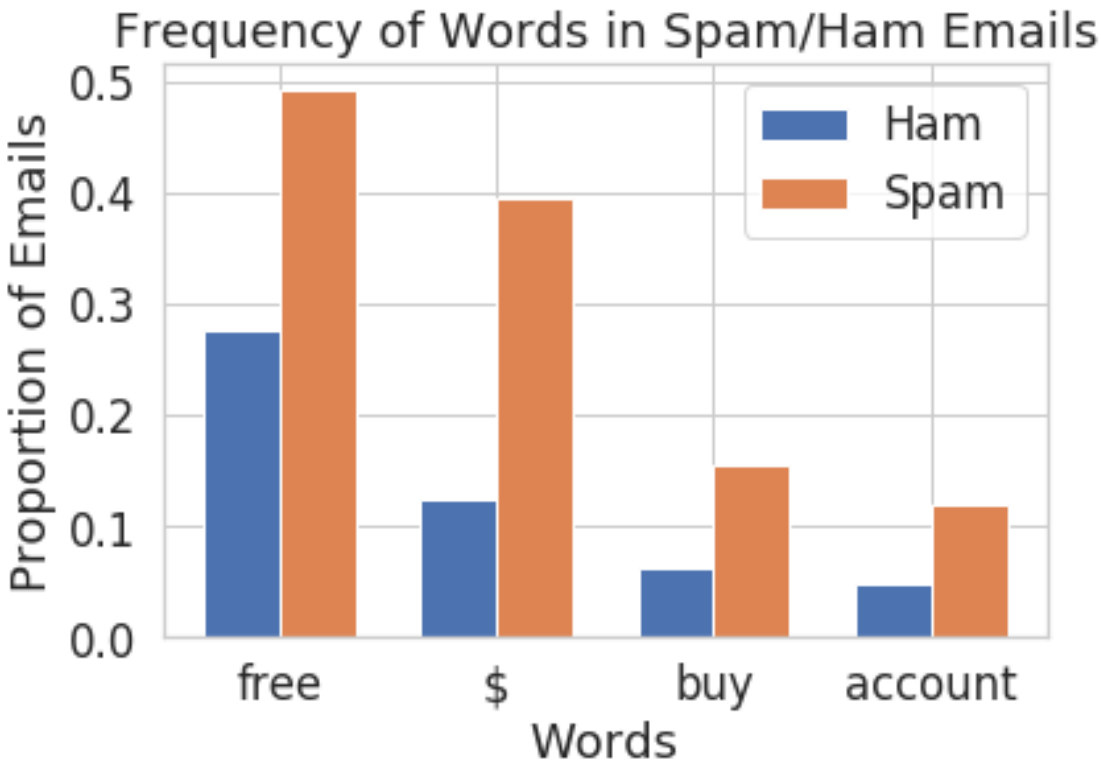
Discuss one thing you notice that is different between the two emails that might relate to the identification of spam.

The spam email is written in what appears to be html while the ham email seems to be a normal string.

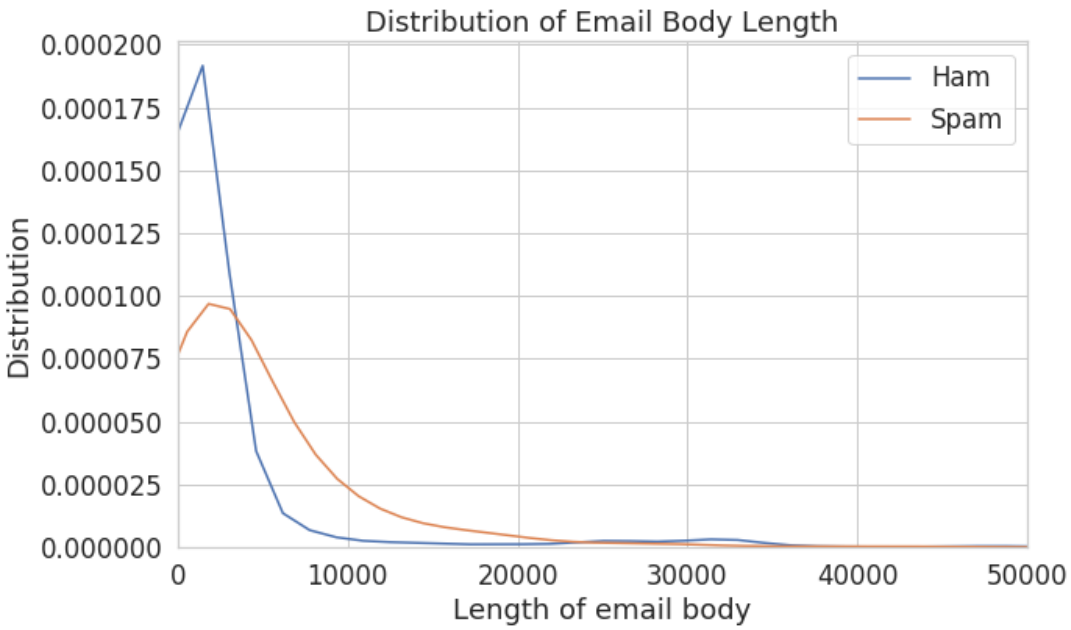
0.0.2 Question 3a

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
         word_set = ['free', '$', 'buy', 'account']
         ham_emails = train[train['spam'] == 0]['email']
         spam_emails = train[train['spam'] == 1]['email']
         words_in_ham = words_in_texts(word_set, ham_emails)
         words_in_spam = words_in_texts(word_set, spam_emails)
         words_in_ham_2 = np.sum(words_in_ham, axis  = 0)
         words_in_spam_2 = np.sum(words_in_spam, axis = 0)
         bar_width = 0.35
         plt.bar(x = word_set, align = 'edge', height = words_in_ham_2/len(ham_emails), label = 'Ham', w
         plt.bar( x = word_set, align = 'edge', height = words_in_spam_2/len(spam_emails), label = 'Spa
         plt.legend()
         plt.xlabel('Words')
         plt.ylabel('Proportion of Emails')
         plt.title('Frequency of Words in Spam/Ham Emails')
         plt.show()
```

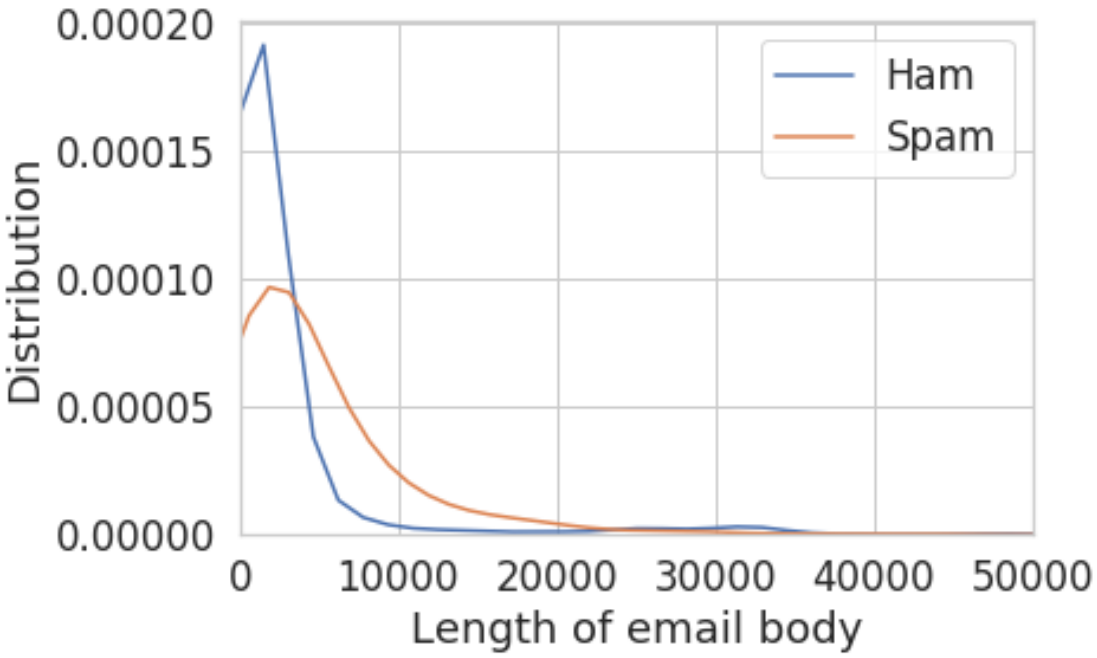


0.0.3 Question 3b



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]: spam_lens = [len(t) for t in train[train['spam'] == 1]['email']]
         ham_lens = [len(t) for t in train[train['spam'] == 0]['email']]
         plt.xlim(0, 50000)
         sns.distplot(ham_lens, label='Ham', hist=False)
         sns.distplot(spam_lens, label='Spam', hist=False)
         plt.legend()
         plt.ylabel('Distribution')
         plt.xlabel('Length of email body')
         plt.savefig('training_conditional_densities.png')
```


0.00020
0.00015
0.00010
0.00005
0.00000
Distribution
Ham
Spam
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)?

Since zero_predictor always predicts 0, it will not predict any emails as spam, so the false positive value is 0. And since zero_predictor never predicts any true positives, the false negative is just the total number of spams, as every spam email is oredicted as a false negative.

Since all emails are considered not spam, the accuracy is the number of original hams divided by the total number of emails. Recall is zero since nothing positive is predicted by the zero_predictor.

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 than false positives (122 vs 1699)

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) The zero predictor classifier has an accuracy of 74.47%, which is less than 75.6%, making the logistic regression classifier the better classifier.

2) The words given could be words common in both spam and ham emails.

3) The logistic regression classifier is preferred for a spam filter, due to its higher prediction accuracy.

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?

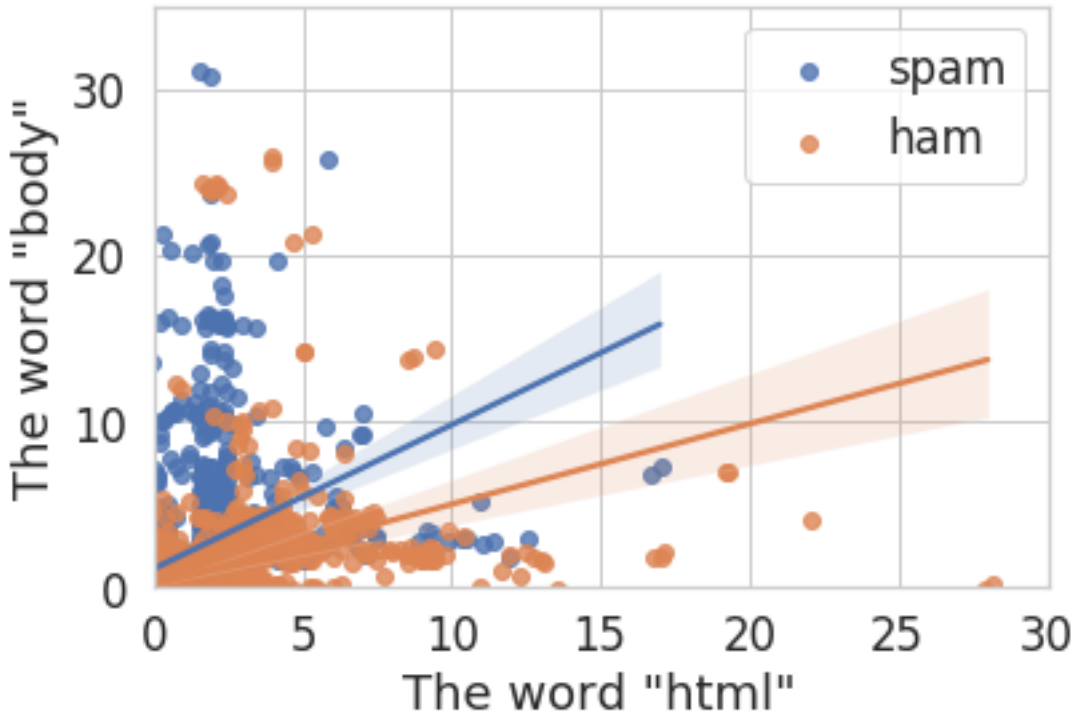
1. I started by following the suggested features the hint gave. I made functions that looked at the proportions of a certain word or character in the spam and ham emails, and I plotted them for visuals. I tried to pick ones that possessed a distinctive difference in proportions between the two types of emails. Using this idea, I tested different traits to see if there is a difference. If there was, I used it as a feature for my model. I also considered just the presence of certain words, instead of their proportions.
2. Some of the features that were useful were the existence of symbols in the body and subject, how many of the word 'you' was in the emails, and how many words were in the subject. These features gave me distribution plots that had a difference in proportions. Another feature that didn't help was the count of number of words in the subject. In terms of the words are included in my model, I used words such as 'gift' and 'please', which i thought would be more present in spam or ham emails. I attained these through trial and error. Some other words I thought would be used in spam ads actually didn't help as much as I had expected.
3. One surprising aspect was what I had mentioned before regarding certain words like 'sale' that seemed intuitively would make a difference, but instead went against my accuracy. Other than there were not much surprises.

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

In [37]:
```
# Write your description (2-3 sentences) as a comment here:
# Using the second hint, I plotted two regplots, one for spam and one for ham. The data points
# of body tags compared to head tags in the emails. Spam emails seems to have more body tags
# and less head tags, and the opposite for ham emails. Body tags would most lilely correspond
# portion of the email, so this plot implies that spam emails are wordier. This helped me deci
#  whether to use the number of words/letters in the emails, as well as in the subjects, as fe


# Write the code to generate your visualization here:
def counter(df, name):
    return df['email'].str.findall(name).str.len()
sns.regplot(x = counter(spams,'head'), y = counter(spams,'body'), x_jitter = 0.5, y_jitter = 0
sns.regplot(x = counter(hams,'head'), y = counter(hams,'body'), x_jitter = 0.5, y_jitter = 0.5
plt.legend()
plt.xlabel('The word "html"')
plt.ylabel('The word "body"')
plt.xlim(0, 30)
plt.ylim(0, 35)
```

Out[37]: (0, 35)



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_probabilities = model.predict_proba(X_train)[:, 1]
false_positive_rate_values, sensitivity_values, thresholds = roc_curve(Y_train, model_probabili
plt.plot(false_positive_rate_values, sensitivity_values)
plt.xlabel('False Positive Rate')
plt.ylabel('Sensitivity')
plt.title('Model ROC Curve');
```