0.0.1 Question 1c

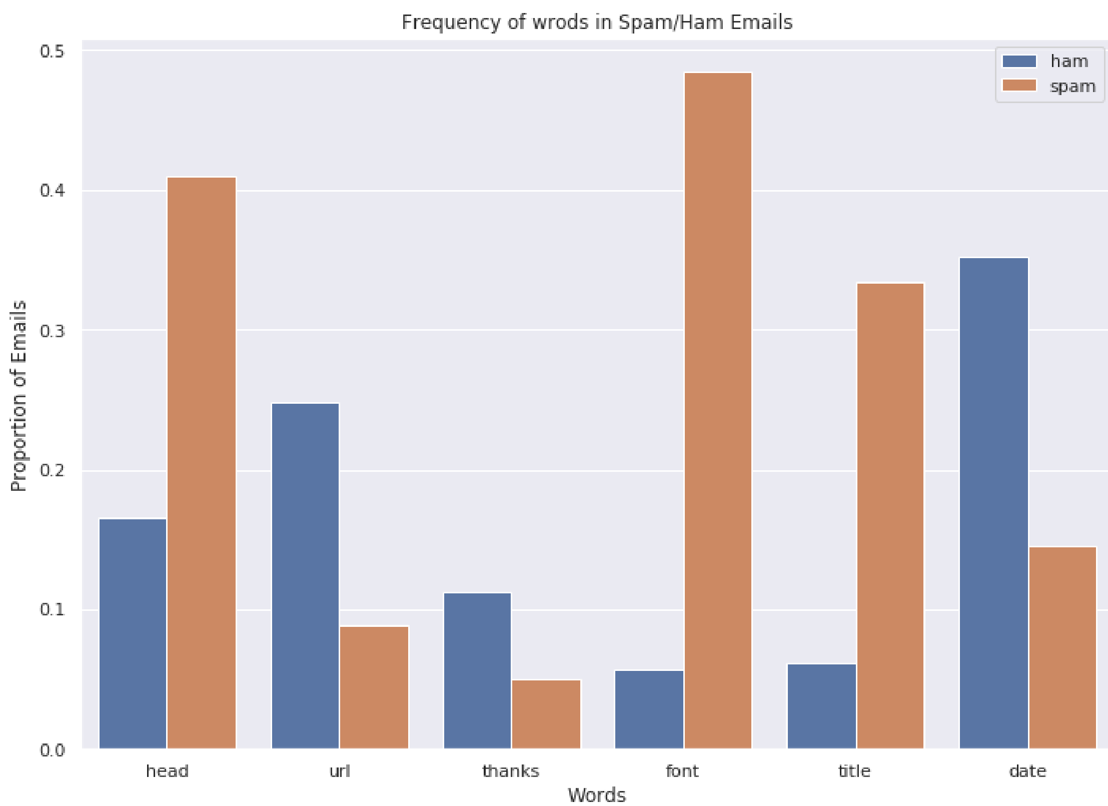
Discuss one thing you notice that is different between the two emails that might relate to the identification of spam.

The first email contains both url and text. The spam email contains the html format text.

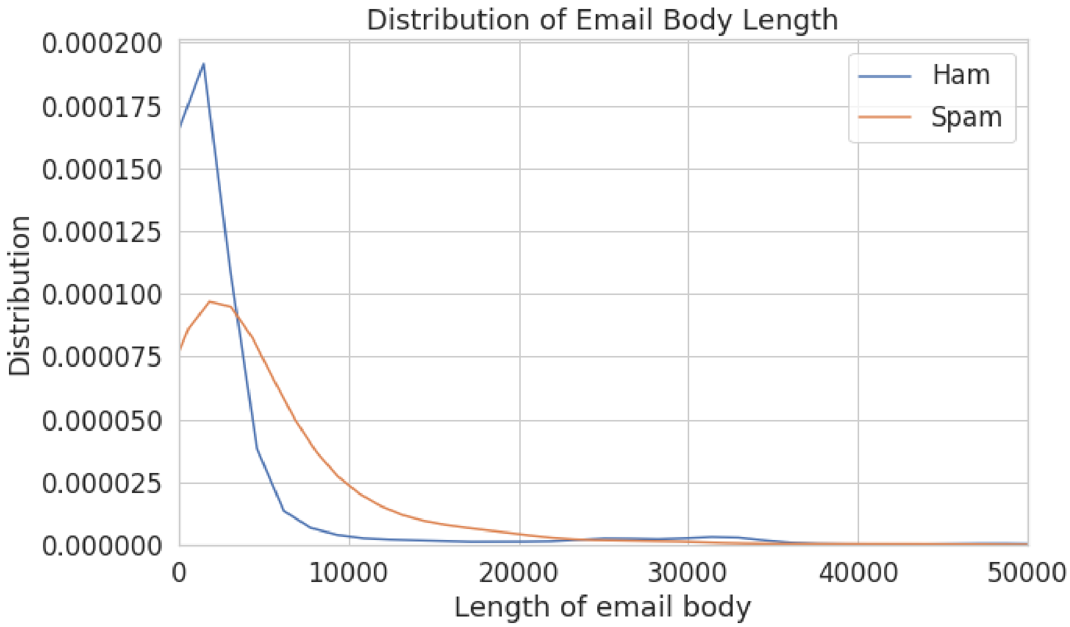
0.0.2 Question 3a

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

         word_list = np.array(['head', 'url', 'thanks', 'font', 'title', 'date'])
         indicator = pd.DataFrame(words_in_texts(word_list, train['email']))
         indicator = indicator.rename({0: "head", 1: 'url', 2:'thanks', 3:'font', 4:'title', 5: 'date'}

         prop_table = indicator.merge(pd.DataFrame(train['spam']), left_index = True, right_index = Tru
         prop_table['spam'].replace({0: 'ham', 1:'spam'}, inplace =True)
         prop_table = prop_table.melt('spam')


         sns.set(rc={'figure.figsize':(11.7,8.27)})
         sns.barplot(x="variable", y="value", hue="spam", data=prop_table, ci=None)
         plt.title("Frequency of wrods in Spam/Ham Emails")
         plt.xlabel("Words")
         plt.ylabel("Proportion of Emails")
         plt.legend(loc = 'upper right');
```

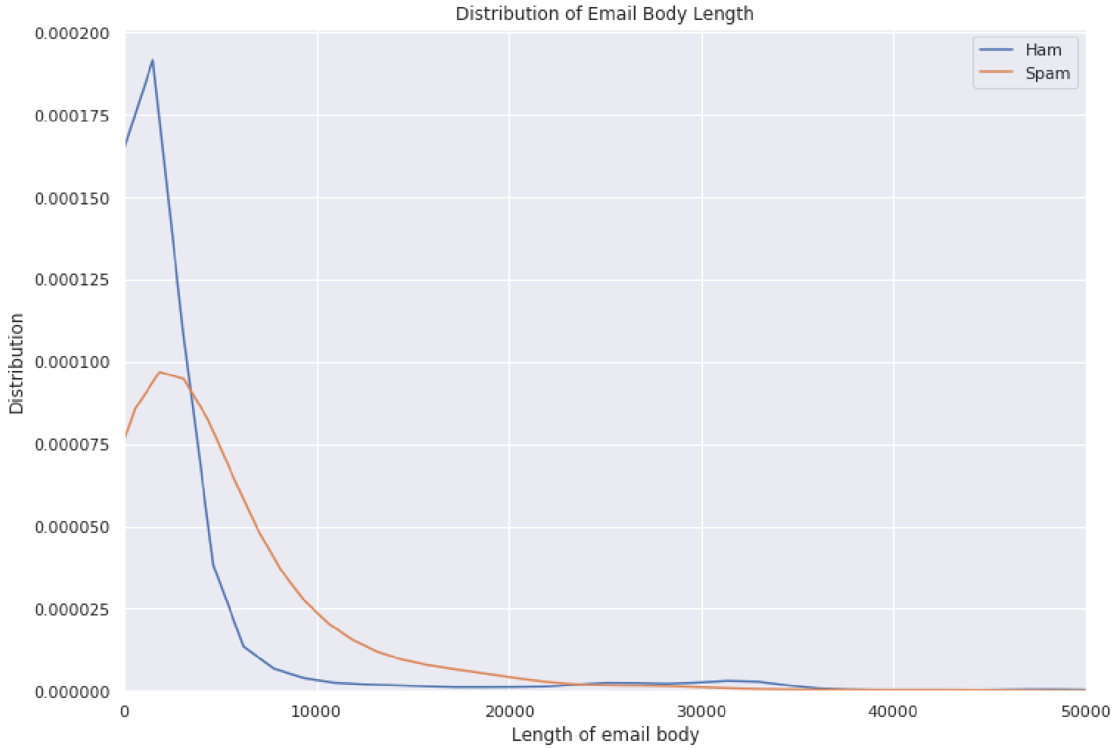


0.0.3 Question 3b



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 [15]: train['len'] = train['email'].apply(len)
         plt.xlim(0, 50000)
         sns.distplot(train.loc[train['spam'] == 0]['len'], hist=False)
         sns.distplot(train.loc[train['spam'] == 1]['len'], hist=False)
         plt.title('Distribution of Email Body Length')
         plt.xlabel('Length of email body')
         plt.ylabel('Distribution')
         plt.legend(labels =["Ham", "Spam"], loc= "upper right")
         plt.savefig('training_conditional_densities.png')
```



Distribution of Email Body Length
Ham
Spam
Distribution
Length of email body
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)?

False Positive is 0, because there is no value labelled as positive(all of them are 0) by the predictor.

The false negative stands for "a spam email gets mislabeled as ham and ends up in the inbox", so all of the spam email is labelled as ham by the predictor, which means the false negative equals to the number of spam emails.

The accuracy is the proportion of all the spam emails being classified as spam and all the ham emails being classified as ham. All the ham emails are predicted as ham by the predictor, but no the spam email is correctly classfied. The accuracy then equals to the proportion of ham email in Y_train set. The true positive(correclty classified the spam emails) equals to 0.

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 using the logistic regression classidier from 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. The accuracy of logistic regression model is higher than just predicting 0 for every email.
2. The word list "['drug', 'bank', 'prescription', 'memo', 'private']" contains no obvious word that distinguish between spam and ham emails. There words are both common in the spam/ham emails.
3. I recommand the logistic regression model. The logistic regression model improves the accuracy rate by 1%. Though there will be no ham email being classifeid as spam if just predicting all emails as 0, the flase-alarm-rate is 0.021805183199285077 for the logistic regression is still pretty low.

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. For email text format: Use of punctuation(How many "!" are there?), and Number of words in the email(either greater than 8,000 or not). For words as features: try to use more HTML tags to distinguish between the spam/ham email.

2. I tried to display the number of character in the subject to distinguish between ham/spam emails, but the distribution is pretty similar. Also, when I add the number of character in the email as a feature in the model, the accuracy goes down. So I decided to remove it from my model. Also, I tried to optimize the logistic model, but got a runtime error, so I commented all my codes related to that.

3. I surprised by non-characters in both the subject and email are good features.

```
In [31]: # Extract Spam Email and Ham Email from the training data
         hams = train.loc[train['spam'] == 0]
         spams = train.loc[train['spam'] == 1]
```

```
In [32]: hams
```

```
Out[32]:        id                                           subject  \
         0     7657            Subject: Patch to enable/disable log\n
         1     6911        Subject: When an engineer flaps his wings\n
         2     6074  Subject: Re: [Razor-users] razor plugins for m…
         3     4376  Subject: NYTimes.com Article: Stop Those Press…
         4     5766  Subject: What's facing FBI's new CIO? (Tech Up…
         …     …                                                …
         7506   466  Subject: Re: [SAtalk] Badly Formatted Spam Rep…
         7508  5734  Subject: [Spambayes] understanding high false …
         7510  5390                        Subject: Facts about sex.\n
         7511   860  Subject: Re: Zoot apt/openssh & new DVD playin…
         7512  7270  Subject: Re: Internet radio - example from a c…

                                                          email  spam    len
         0     while i was playing with the past issues, it a…     0   1641
         1     url: http://diveintomark.org/archives/2002/10/…     0   4713
         2     no, please post a link!\n \n fox\n ----- origi…     0   1399
         3     this article from nytimes.com \n has been sent…     0   4435
         4     <html>\n <head>\n <title>tech update today</ti…     0  32857
         …                                                   …    …      …
```

```
7506  on wed, 11 sep 2002, vince puzzella wrote:\n \…     0   1342
7508  >>>>> "tp" == tim peters <tim.one@comcast.net>…     0    465
7510  \n forwarded-by: flower\n \n did you know that…     0   1732
7511  on tue, oct 08, 2002 at 04:36:13pm +0200, matt…     0   1098
7512  chris haun wrote:\n > \n > we would need someo…     0    812

[5595 rows x 5 columns]
```

In [33]:
```python
spams
```

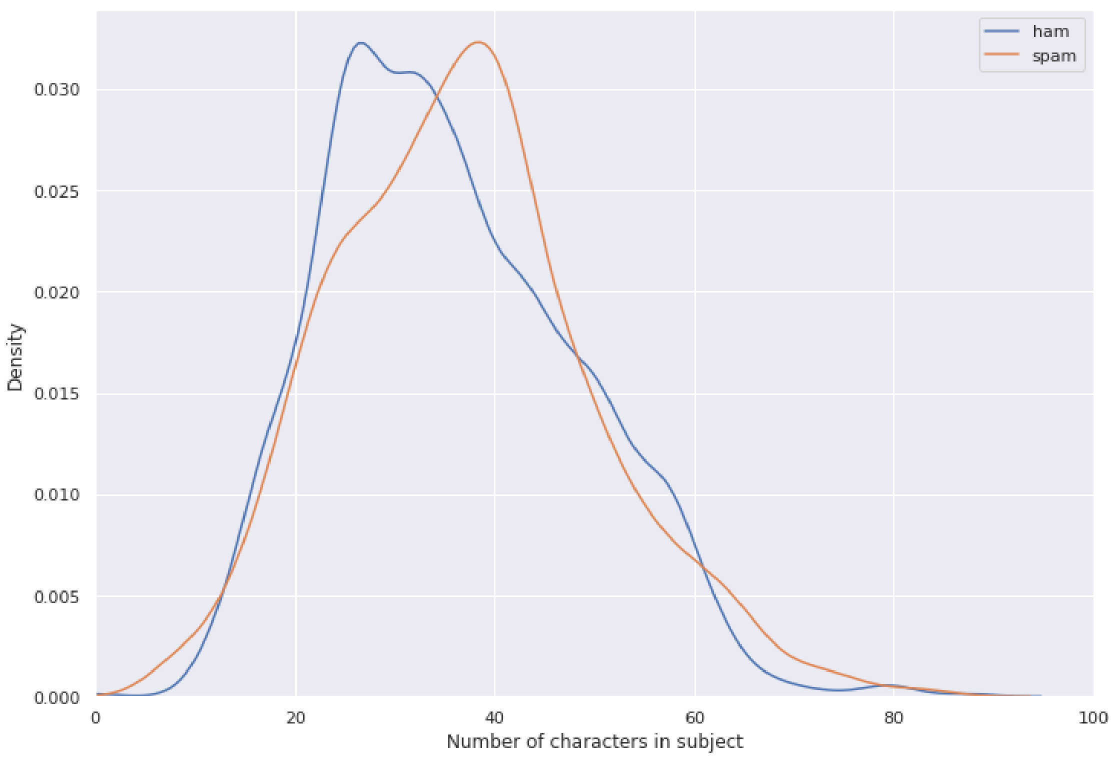
Out[33]:
```
        id                                           subject  \
5     5247                                   Subject: asap\n
13     354                          Subject: Re: Your VIP Pass\n
15    6449  Subject: wives and girlfriends cheating and wh…
18    5338  Subject: Fw: Offring Membership To 16 Sites Fo…
19    5366      Subject: The Government grants you $25,000!\n
…     …                                                 …
7502  1685  Subject: Hi Janet, are you going to call me?  …
7503  8322                        Subject: WWW Form Submission\n
7505  4426    Subject: cell phone ring tones 84221111000000\n
7507  6265             Subject: MY PLEA FOR ASSISTANCE PLEASE\n
7509  5191         Subject: Reach millions on the internet!!\n

                                                  email  spam    len
5     --===_secatt_000_1fuklemuttfusq\n content-type…     1   1156
13    ####################################\n \n   fre…     1   2865
15    <table width="600" border="20" align="center" …     1    820
18    <html>\n <head>\n </head>\n <body>\n <center>\…     1   7361
19    <html>\n <head>\n </head>\n <center>\n <h1>\n …     1  27775
…                                                   …   …     …
7502  <html>\n <body bgcolor=3d"#003300">\n <p align…     1   4809
7503  below is the result of your feedback form.  it…     1    612
7505  <html>\n <table width="350" border="0" cellspa…     1   8504
7507  \n mr. ayanda maredi\n department of minerals …     1   2663
7509  \n dear consumers, increase your business sale…     1   7054

[1918 rows x 5 columns]
```

In [34]:
```python
def countstr(df, col,name):
    '''
    Args:
        df (dataframe): dataframe to operate
        name (string): string to count

    Returns:
        the number of occurance of certain string in each row of mail in the dataframe
    '''
    return df[col].str.findall(name).str.len()
```

In [35]:
```python
def plotting(col, name):
```

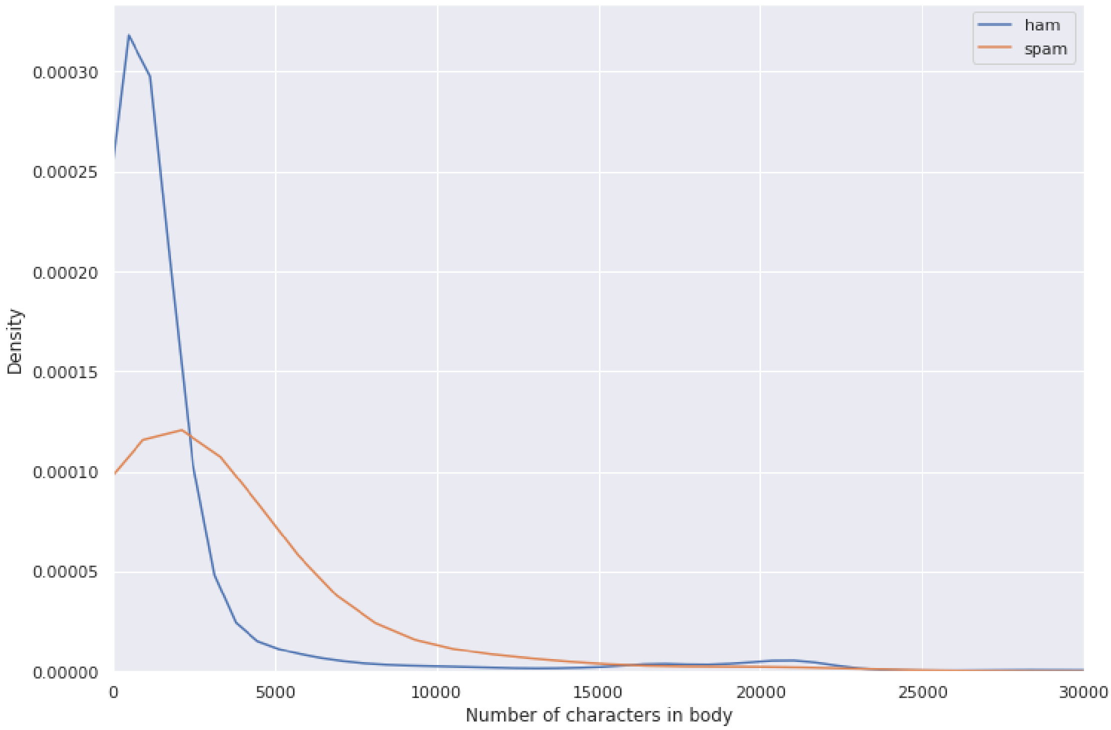
```
    '''
    Args:
        col (dataframe column): dataframe column to operate
        name (string): string to count

    Returns:
        the distplot of distribution of string in spam/ham emails
    '''
    count_ham = countstr(hams, col, name)
    count_spam = countstr(spams, col, name)
    sns.distplot(count_ham, hist=False)
    sns.distplot(count_spam, hist=False)
    plt.legend(labels = ['ham', 'spam'], loc = 'upper right')
```

```
In [36]: #A. Number of characters in the subject
         plotting('subject', r'\w')
         plt.xlim(0, 100)
         plt.xlabel('Number of characters in subject');
```

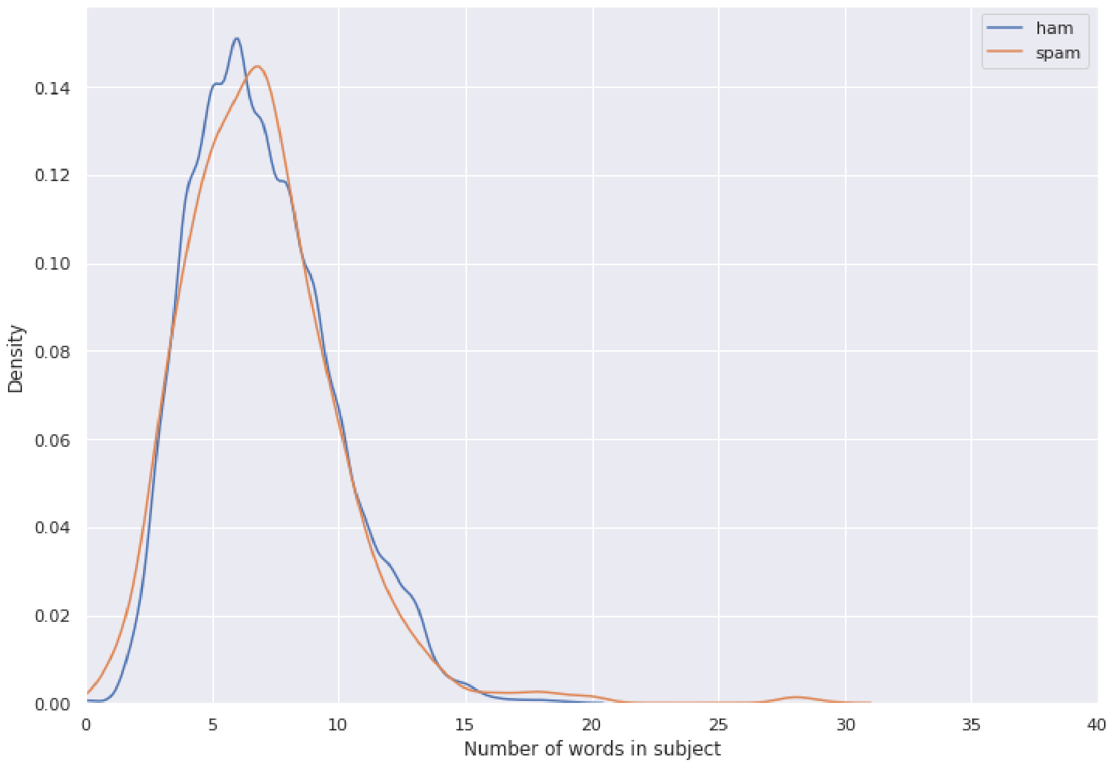



```
In [37]: plotting('email',r'\w')
```

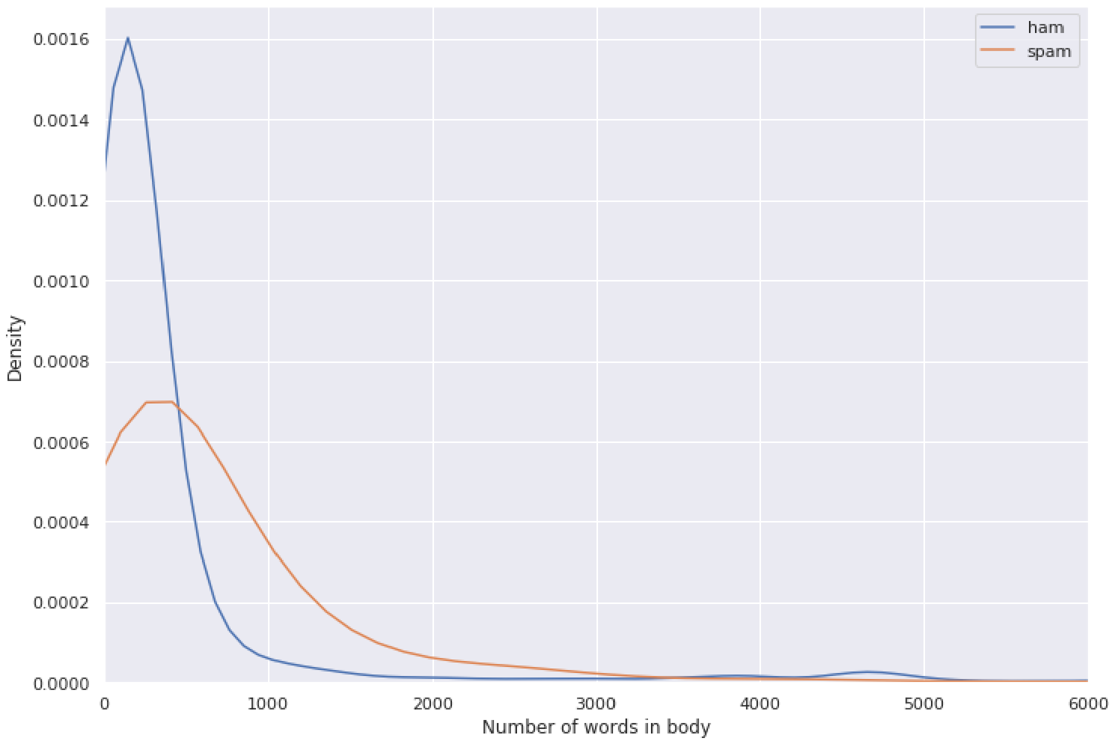
```
plt.xlim(0,30000)
plt.xlabel('Number of characters in body');
```



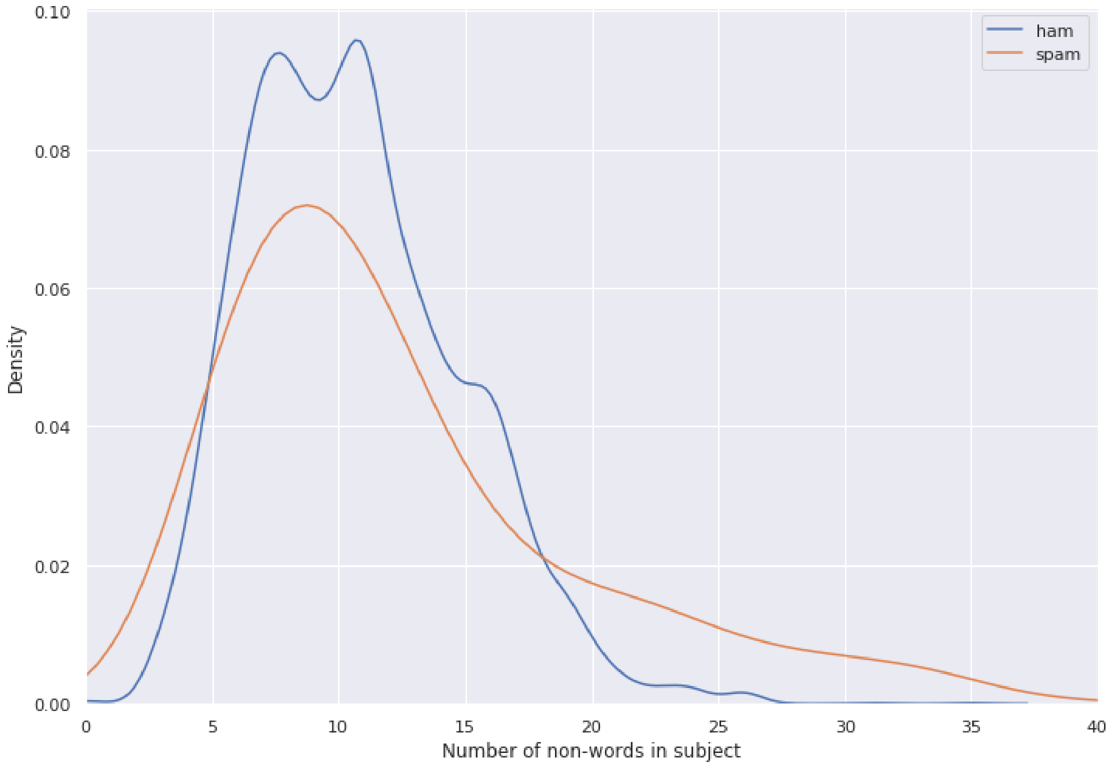

```
In [38]: #B. Number of words in the subject
         plotting('subject',r'[a-zA-Z]+')
         plt.xlim(0, 40)
         plt.xlabel('Number of words in subject');
```



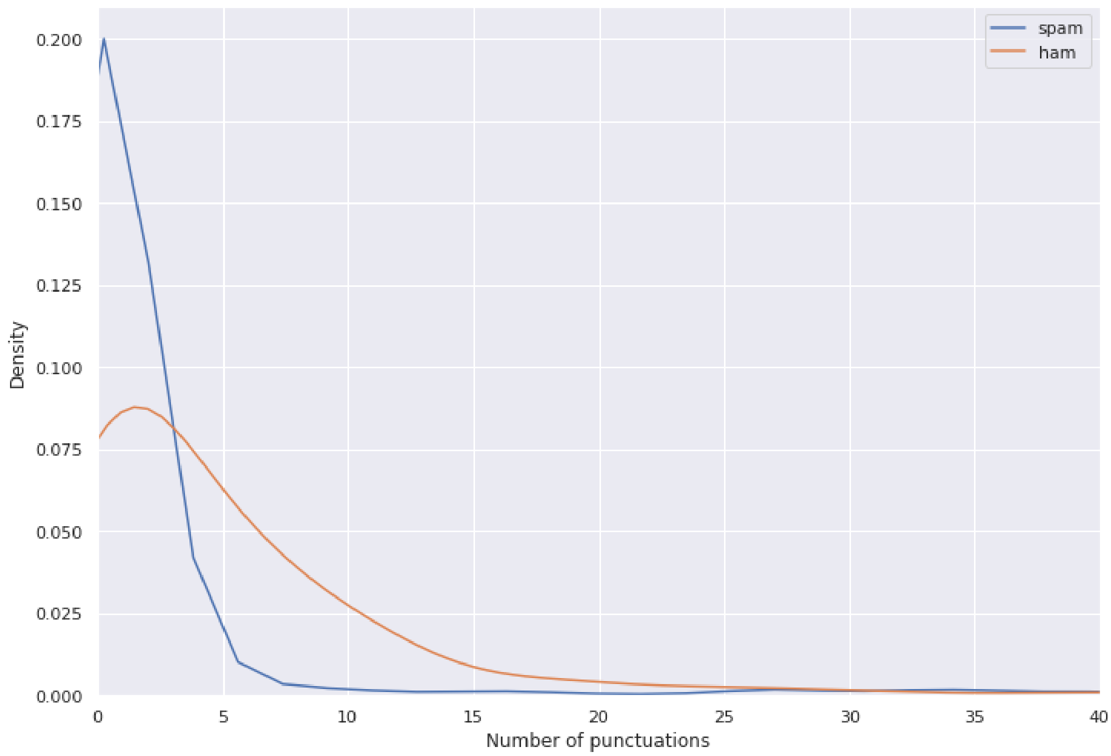
In [39]:
```
#Number of words in the email
plotting('email',r'[a-zA-Z]+')
plt.xlim(0, 6000)
plt.xlabel('Number of words in body');
#there's difference
```



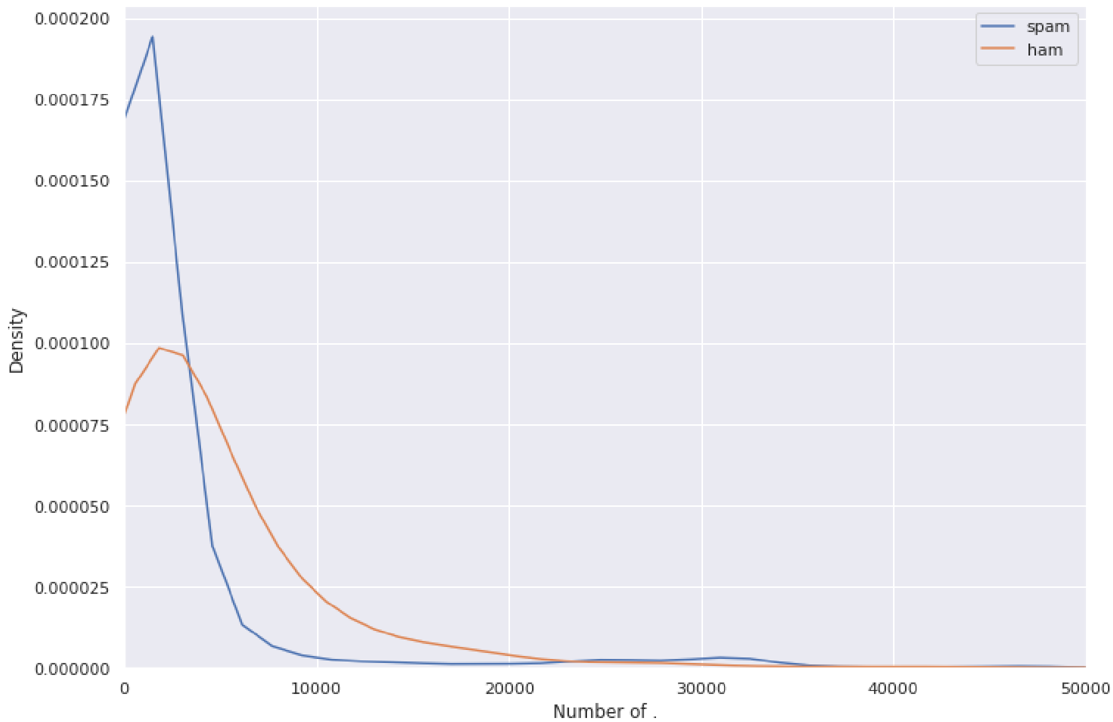
In [40]:
```
plotting('subject',r'[^a-zA-Z0-9]')
plt.xlim(0, 40)
plt.xlabel('Number of non-words in subject');
```




```
In [41]: #C. Use of punctuation (e.g., how many '!'s were there?)
         ham_punct = countstr(hams, 'email','!')
         spam_punct = countstr(spams, 'email','!')
         sns.distplot(ham_punct, hist=False)
         sns.distplot(spam_punct, hist=False)
         plt.legend(labels = ['spam', 'ham'], loc = 'upper right')
         plt.xlabel('Number of punctuations')
         plt.xlim(0,40);
         #there's difference
```




```
In [42]: ham_dot = countstr(hams, 'email','.')
         spam_dot  = countstr(spams, 'email','.')
         sns.distplot(ham_dot, label = 'ham', hist=False)
         sns.distplot(spam_dot, label = 'spam', hist=False)
         plt.xlabel('Number of .')
         plt.legend(labels = ['spam', 'ham'], loc = 'upper right')
         plt.xlim(0,50000);
         #there's difference
```




```
In [43]: def count_top(df):
             word_count = {}
             word_list= pd.DataFrame(df['email']).stack().str.split("[^\w+]").explode().tolist()
             for w in word_list:
                 word_count[w] = (word_count[w] + 1) if w in word_count else 1
             return word_count
```

```
In [44]: words_features = (pd.Series(count_top(train)) / train.shape[0]).sort_values(ascending=False).i
         words_features.index
```

```
Out[44]: Index(['', 'the', 'a', '3d', 'font', 'to', 'td', 'http', 'com', 'and',
                …
                'document', 'bank', 'federal', 'women', 'collapse', 'increase',
                'freedom', 'political', 'd3', 'notice'],
               dtype='object', length=1000)
```

```
In [59]: # Logistic Regression()
         def feature(df):
             words = ['click', 'url', 'thanks', 'date', 'title','wish', 'bank', '$',
                      'credit', 'subscribe', 'gift', 'font','body', 'business',
```

```
                'html', 'money', 'offer', 'please', '!', '.', 'free',
                'td', 'br','reply', 'http','href'] + words_features.index.tolist()
    X_train = words_in_texts(words, df['email']).astype(int)
    feature = pd.concat([
        countstr(df, 'email',r'[a-zA-Z]+').fillna(0),
        df["subject"].fillna("").apply(lambda x: 1 if "Re:" in x else 0),
        df["subject"].fillna("").apply(lambda x: 1 if "Fw:" in x else 0),
        countstr(df, 'email',r'[A-Z]').fillna(0),
        countstr(df, 'subject',r'[A-Z]').fillna(0),
        countstr(df, 'email','[^a-zA-Z0-9]').fillna(0)], axis=1).values
    X_train = np.concatenate((X_train, feature), axis=1)
    return X_train

X_train_more = feature(train)

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

```
Training Accuracy:  1.0
```

Add square root and tanh values for each columns as features

In [60]:
```
feature_df = pd.DataFrame(X_train_more)
feature_df_with_extra_features = feature_df.copy()
for feature_name in feature_df.columns:
    #feature_df_with_extra_features[str(feature_name) + "^2"] = feature_df_with_extra_features
    feature_df_with_extra_features["sqrt" + str(feature_name)] = np.sqrt(feature_df_with_extra_
    feature_df_with_extra_features["tanh" + str(feature_name)] = np.tanh(feature_df_with_extra_
feature_df_with_extra_features
```

Out[60]:
```
      0  1  2  3  4  5  6  7  8  9  …  sqrt1027  tanh1027  sqrt1028  \
0     0  0  0  0  0  0  0  1  0  0  …       0.0  0.000000       0.0
1     0  1  0  1  0  0  0  0  0  0  …       0.0  0.000000       0.0
2     0  0  0  0  0  0  0  0  0  0  …       1.0  0.761594       0.0
3     0  0  0  0  0  0  0  0  0  0  …       0.0  0.000000       0.0
4     1  1  0  1  1  1  0  1  0  1  …       0.0  0.000000       0.0
…    .. .. .. .. .. .. .. .. .. ..  …       …         …         …
7508  0  0  0  0  0  0  0  0  0  0  …       0.0  0.000000       0.0
7509  0  0  0  1  0  0  1  1  1  0  …       0.0  0.000000       0.0
7510  0  0  0  0  0  0  0  0  0  0  …       0.0  0.000000       0.0
7511  0  0  0  1  0  0  0  0  0  0  …       1.0  0.761594       0.0
7512  0  0  0  1  0  0  0  1  0  0  …       1.0  0.761594       0.0

      tanh1028  sqrt1029  tanh1029  sqrt1030  tanh1030   sqrt1031  tanh1031
0          0.0       0.0       0.0  1.414214  0.964028  24.698178       1.0
1          0.0       0.0       0.0  1.414214  0.964028  34.394767       1.0
2          0.0       0.0       0.0  1.732051  0.995055  23.409400       1.0
```

```
3           0.0       0.0       0.0  3.464102  1.000000   32.511536       1.0
4           0.0       0.0       0.0  3.162278  1.000000  106.235587       1.0
…        …       …       …       …       …       …       …
7508        0.0       0.0       0.0  1.414214  0.964028   12.247449       1.0
7509        0.0       0.0       0.0  1.414214  0.964028   52.287666       1.0
7510        0.0       0.0       0.0  1.414214  0.964028   20.856654       1.0
7511        0.0       0.0       0.0  2.449490  0.999988   18.165902       1.0
7512        0.0       0.0       0.0  1.732051  0.995055   14.966630       1.0

[7513 rows x 3096 columns]
```

In [61]:
```
model.fit(feature_df_with_extra_features, Y_train)
training_accuracy_1 = model.score(feature_df_with_extra_features, Y_train)
print("Training Accuracy(more features): ", training_accuracy_1)
```

```
Training Accuracy(more features):  1.0
```

In [62]:
```
#def mse(y_obs, y_hat):
#    return np.mean((y_obs - y_hat)**2)

#def sigma(t):
#    return 1 / (1 + np.exp(-t))

#def predicted_data_given_features(X, theta):
#    return sigma(theta @ X.T)
```

In [63]:
```
#from scipy.optimize import minimize
#def mse_for_model_on_full_data(theta):
#    y_hat = predicted_data_given_features(feature_df_with_extra_features, theta)
#   return mse(Y_train, y_hat)
#theta_68_hat = minimize(mse_for_model_on_full_data, x0=np.zeros(68)).x
#mse_for_model_on_full_data(theta_68_hat)
```

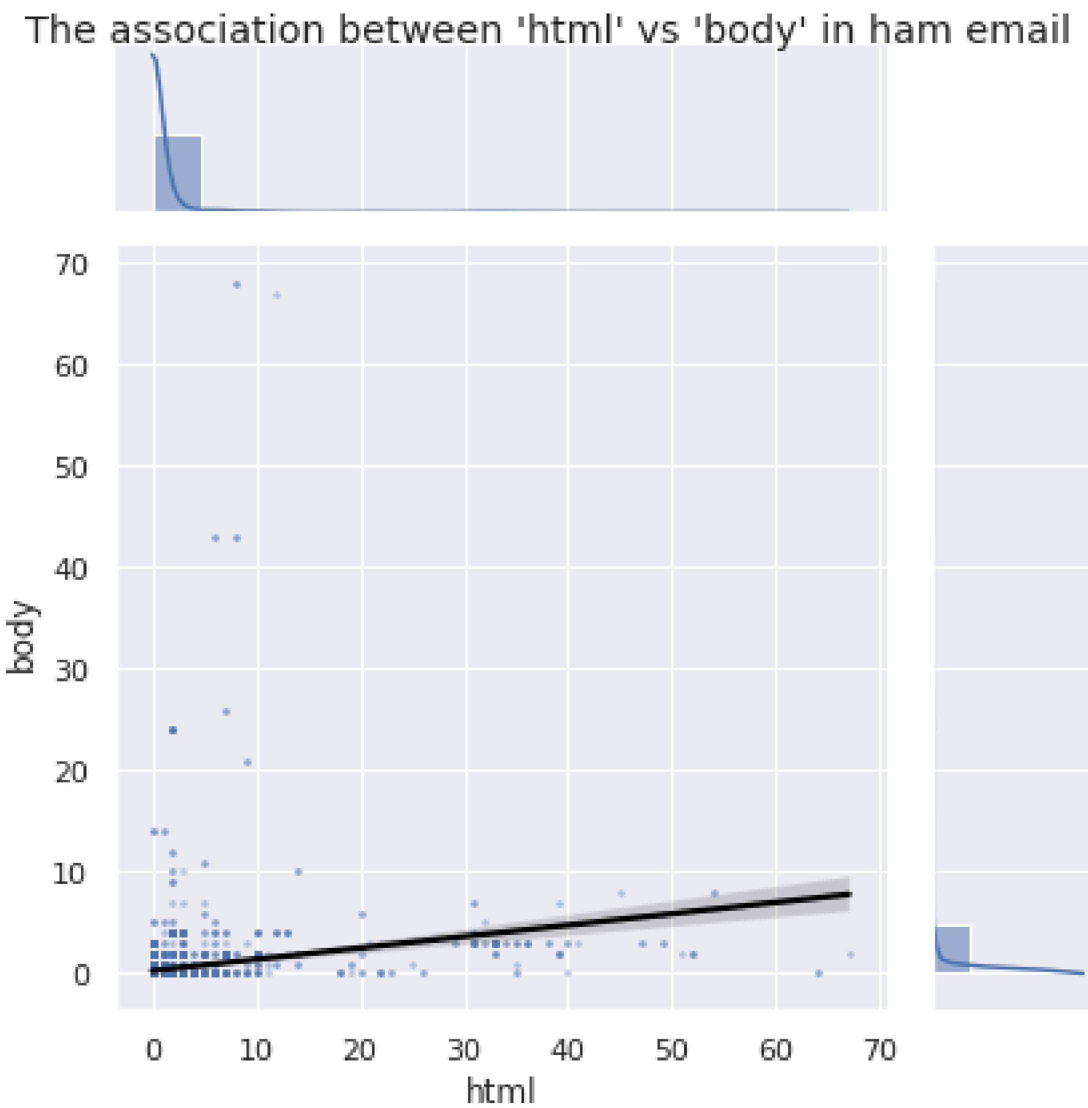
In [ ]:

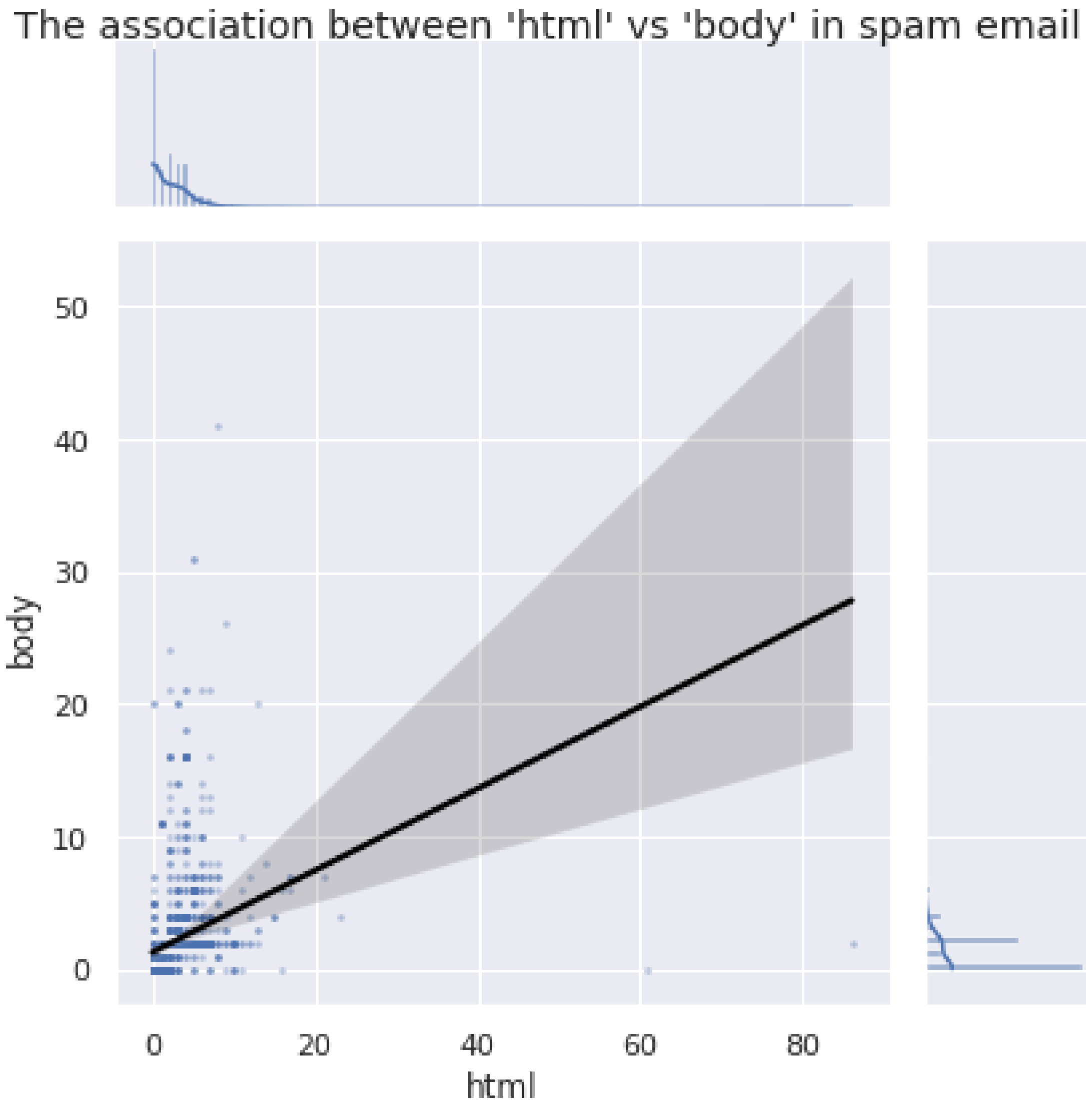
In [ ]:

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

In [64]:
```
# Write your description (2-3 sentences) as a comment here:
# The body and html did co-occur relatively frequently(from the distribution diagram), because
# frequently-used html tags; In hams, 'html' spreads over a larger range, but the occurance of
# [0,20] in spams. Spam emails have more frequent occurance of "body" than the one of "html",
# at the beginning and the end of one html block. But one html block can have multiple <body>
# content of email.
# Ham emails have more frequent occurance of html, maybe in the form of url links. Less usage
# is more likely be in the content of email.
#

# Write the code to generate your visualization here:

html_body_ham = pd.DataFrame({"html":countstr(hams, 'email','html') , "body": countstr(hams, '

html_body_spam = pd.DataFrame({"html":countstr(spams, 'email','html') , "body": countstr(spams



p1 = sns.jointplot(
    x='html',
    y='body',
    data=html_body_ham,
    kind="reg",
    ratio=4,
    space=0,
    scatter_kws={
        's': 3,
        'alpha': 0.25
    },
    line_kws={
        'color': 'black'
    }
)

p1.fig.suptitle("The association between 'html' vs 'body' in ham email")
p1.fig.tight_layout()
p1.fig.subplots_adjust(top=0.95);


p2 = sns.jointplot(
    x='html',
    y='body',
    data=html_body_spam,
    kind="reg",
    ratio=4,
    space=0,
    scatter_kws={
        's': 3,
        'alpha': 0.25
    },
    line_kws={
```

```
        'color': 'black'
    }
)

p2.fig.suptitle("The association between 'html' vs 'body' in spam email")
p2.fig.tight_layout()
p2.fig.subplots_adjust(top=0.95);
```




The association between 'html' vs 'body' in spam email
body
html
0
10
20
30
40
50
0
20
40
60
80

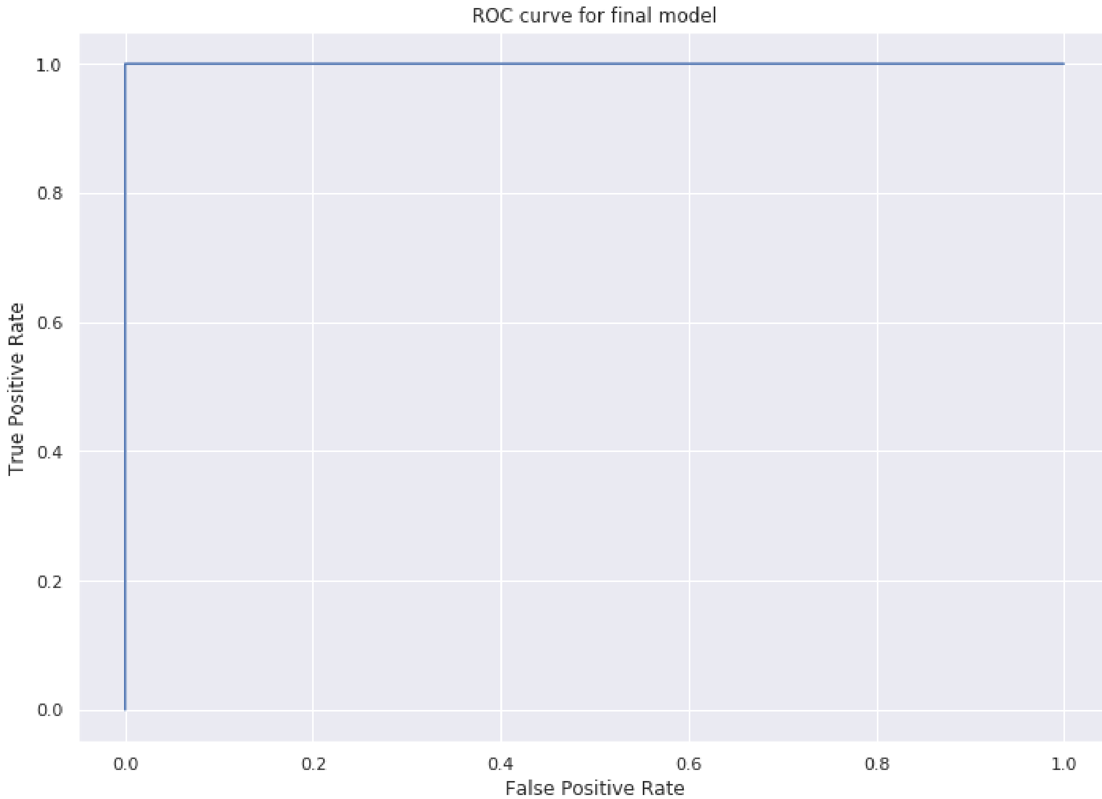
0.0.8 Question 9: ROC Curve

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 [65]: from sklearn.metrics import roc_curve
         import plotly.express as px

         # Note that you'll want to use the .predict_proba(...) method for your classifier
         # instead of .predict(...) so you get probabilities, not classes

         words_list_model_probabilities = model.predict_proba(feature_df_with_extra_features)[:, 1]
         fpr, tpr, threshold = roc_curve(Y_train, words_list_model_probabilities, pos_label=1)

         plt.plot(fpr, tpr)
         plt.xlabel("False Positive Rate")
         plt.ylabel("True Positive Rate")
         plt.title("ROC curve for final model");
```


ROC curve for final model
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate