
0.1 Question 1

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

Type your answer here, replacing this text.

0.2 Question 2a

Generate your visualization in the cell below.

In [23]:
```
train = pd.read_csv('train.csv')
```

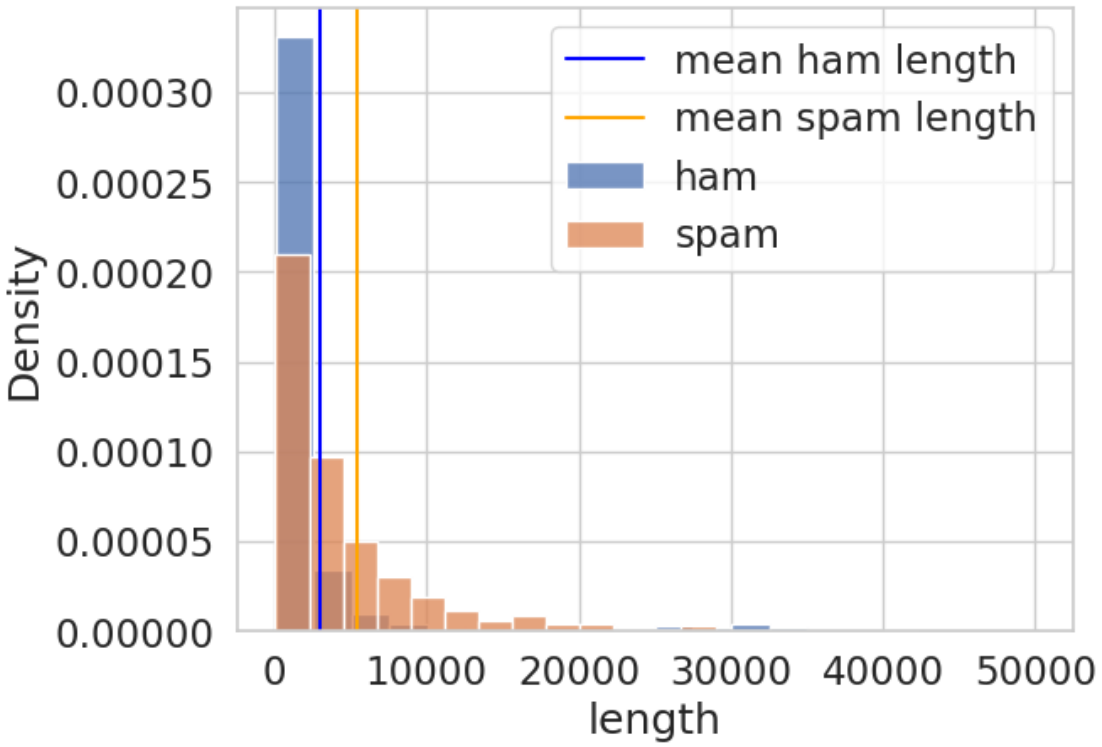
In [24]:
```
train[train.spam == 0].email.values[165]
```

Out[24]:
```
'On Tue 30 Jul 2002 10:28, David Neary wrote:\n \n > I have 3 or 4 email addresses (which get u
```

0.3 Measuring the length of an email

In [25]:
```
train['length'] = train['email'].apply(lambda x: len(x))

        sns.histplot(data = train[(train.length < 50000) & (train.spam == 0)], x = 'length', label = 'h
        sns.histplot(data = train[(train.length < 50000) & (train.spam == 1)], x = 'length', label = 's

        plt.axvline(np.mean(train[train.spam == 0]['length']), label = 'mean ham length', color = 'blue
        plt.axvline(np.mean(train[train.spam == 1]['length']), label = 'mean spam length', color = 'ora

        plt.legend()
```

Out[25]:
```
<matplotlib.legend.Legend at 0x7fba2424bf90>
```



0.4 Word Selection

I found a website that had a bunch of spam phrases and words. Source: https://www.activecampaign.com/blog/spam-words

```
In [26]: sample_spam_words = pd.read_csv('spam_texts.csv')
         sample_spam_words.columns = ['spam_phrases']
         sample_spam_words
```

Out[26]:

	spam_phrases
0	100% more
1	100% free
2	100% satisfied
3	Additional income
4	Be your own boss
..	…
182	Trial
183	Unlimited

```
184           Warranty
185        Web traffic
186     Work from home

[187 rows x 1 columns]
```

In [27]:
```
word = (sample_spam_words.spam_phrases[0:2])
```

```
train[word] = words_in_texts(word, train['email'])
```

In [28]:
```
existing_words = [
    "Free", "Winner", "Guaranteed", "Urgent", "Offer",
    "Discount", "Limited", "Risk-free", "Prize", "Congratulations",
    "Exclusive", "Act now", "Buy now", "Special", "Promotion",
    "Wealth", "Cheap", "Save", "Investment", "Credit",

'Miracle',
'Instant',
'Breakthrough',
'Secret',
'Money',
'Offer',
'Profits',
'Amazing',
'Incredible',
'Revolutionary',
'Sensational',
'Easy',
'Opportunity',
'Cash',
'Bargain',
'Best price',
'One time',
'Exclusive deal',
'Limited time',
'Urgent',
'Luxury',
'Elite',
'Premium',
'Fortune',
'Free trial',
'Subscribe',
'Membership',
'No obligation',
'Payout',
'Bonus',
]

words = sample_spam_words.spam_phrases.tolist() + existing_words
```

```
new_train = train.copy()

indicator_data = []

for word in words:
    indicator_array = words_in_texts([word], new_train['email'])
    indicator_data.append(indicator_array[:, 0])  # Convert to 1D and append

# Create a DataFrame from the indicator data
indicator_df = pd.DataFrame(indicator_data).transpose()
indicator_df.columns = words

# Concatenate the new DataFrame with the original one
new_train = pd.concat([new_train, indicator_df], axis=1)
```

In [29]:
```
# display(new_train.iloc[:, range(-1 * len(words), 0)])
# display(new_train)

cols = ['spam'] + words

dat = new_train[cols].melt('spam')
dat.value_counts()
# dispaly(dat)
spam_dict = {0: 'ham', 1: 'spam'}
dat['label'] = dat['spam'].map(spam_dict)

pt = pd.pivot_table(dat, index = 'variable', columns = 'label', aggfunc = 'mean').reset_index()
pt.columns = ['variable', 'spam', 'spam', 'ham_perc', 'spam_perc']
pt['differential'] = pt.spam_perc - pt.ham_perc
# display(pt.head())
best_words = pt.sort_values('differential', ascending = False).variable.values[0:20]
best_words
```

Out[29]:
```
array(['Free', 'Ad', 'Offer', 'Credit', 'Save', 'Click here',
       'Guaranteed', 'Money', 'Rates', 'Special', 'Refinance', 'Debt',
       'Cash', 'Quote', 'Easy', 'Loans', 'Secret', 'Limited', 'Removal',
       'Promotion'], dtype=object)
```

In [30]:
```
pt.sort_values('differential', ascending = False)
```

Out[30]:

	variable	spam	spam	ham_perc	spam_perc	differential
68	Free	0	1	0.058956	0.178037	0.119081
5	Ad	0	1	0.125966	0.217290	0.091323
139	Offer	0	1	0.003866	0.074299	0.070433
41	Credit	0	1	0.001128	0.069159	0.068031
167	Save	0	1	0.005799	0.070093	0.064294
..	…	…	…	…	…	…
34	Clearance	0	1	0.002899	0.002336	-0.000563

```
170        Score     0     1  0.000966   0.000000     -0.000966
58         Elite     0     1  0.003061   0.001869     -0.001191
181    Subscribe     0     1  0.013209   0.010748     -0.002461
149  Please read     0     1  0.009826   0.001869     -0.007957

[212 rows x 6 columns]
```

0.5 HMTL Analysis

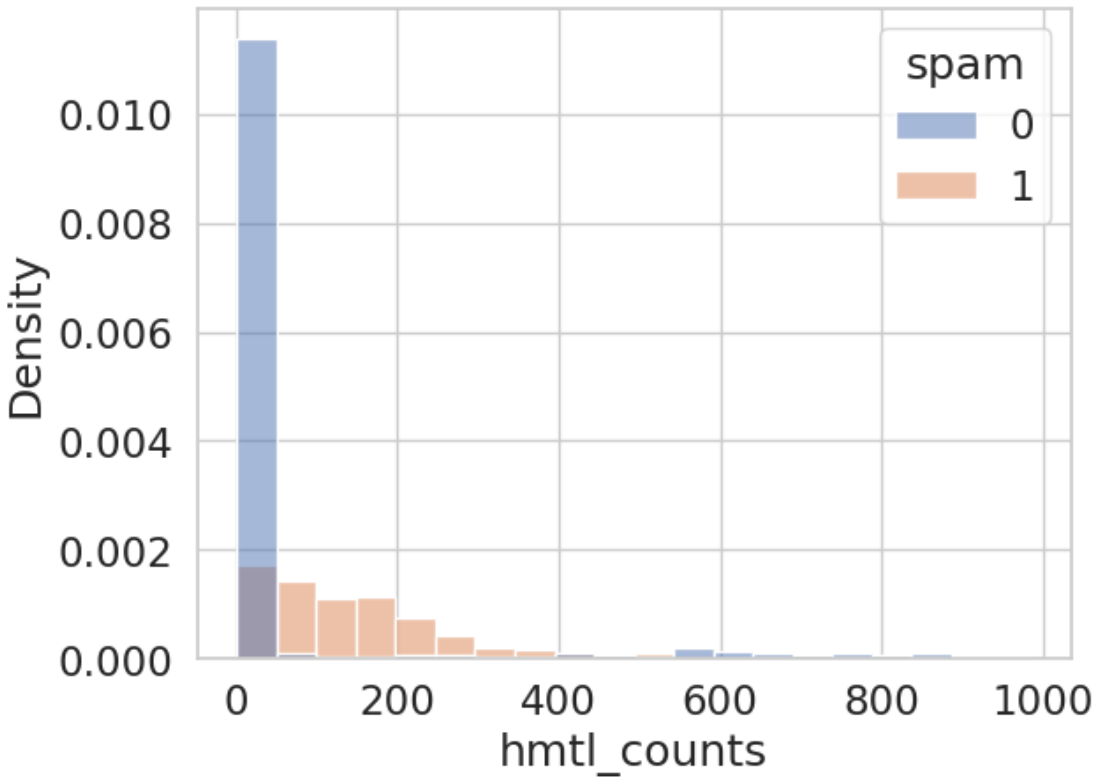
In [31]:
```
# train.email[2]
```

In [32]:
```
new_train.head()
```

Out[32]:
```
   id                                             subject  \
0   0  Subject: A&L Daily to be auctioned in bankrupt…
1   1  Subject: Wired: "Stronger ties between ISPs an…
2   2  Subject: It's just too small                  …
3   3                      Subject: liberal defnitions\n
4   4  Subject: RE: [ILUG] Newbie seeks advice - Suse…

                                               email  spam  length  100% more  \
0  URL: http://boingboing.net/#85534171\n Date: N…     0     359          0
1  URL: http://scriptingnews.userland.com/backiss…     0     278          0
2  <HTML>\n <HEAD>\n </HEAD>\n <BODY>\n <FONT SIZ…     1     444          0
3  Depends on how much over spending vs. how much…     0    1500          0
4  hehe sorry but if you hit caps lock twice the …     0    2018          0

   100% free  100% more  100% free  100% satisfied  …  Luxury  Elite  \
0          0          0          0               0  …       0      0
1          0          0          0               0  …       0      0
2          0          0          0               0  …       0      0
3          0          0          0               0  …       0      0
4          0          0          0               0  …       0      0

   Premium  Fortune  Free trial  Subscribe  Membership  No obligation  Payout  \
0        0        0           0          0           0              0       0
1        0        0           0          0           0              0       0
2        0        0           0          0           0              0       0
3        0        0           0          0           0              0       0
4        0        0           0          0           0              0       0

   Bonus
0      0
1      0
2      0
3      0
4      0

[5 rows x 244 columns]
```

In [33]:
```
"""
First, we are detecting the number of HTML tags that exist in each email. It seems that way mo
in spam emails rather than regular emails.
"""


html_stuff = train.email.str.extractall('<([^>]+)>').reset_index()
html_gb = html_stuff.groupby('level_0').agg('count')

new_train = train.copy()
new_train['hmtl_counts'] = html_stuff.groupby('level_0').agg('count').iloc[:, 0]
new_train.fillna(0)

sns.histplot(data = new_train[new_train.hmtl_counts < 1000], hue = 'spam', x = 'hmtl_counts', l
```

Out[33]: <Axes: xlabel='hmtl_counts', ylabel='Density'>



Trying to understand how many 'head' and 'body' formatting to see if there's a pattern

In [34]:
```
html_stuff['new_lowered'] = html_stuff[0].apply(lambda string: string.lower())
html_stuff
html_stuff[(html_stuff.level_0 == 7) & ((html_stuff.new_lowered.str.contains('body')) | (html_s
```

Out[34]:

	level_0	match	0	new_lowered
16	7	0	body lang=EN-US	body lang=en-us
18	7	2	p class=MsoBodyText style='text-align:justify'	p class=msobodytext style='text-align:justify'
30	7	14	p class=MsoBodyText style='text-align:justify'	p class=msobodytext style='text-align:justify'
38	7	22	p class=MsoBodyText style='text-align:justify'	p class=msobodytext style='text-align:justify'
42	7	26	p class=MsoBodyText style='text-align:justify'	p class=msobodytext style='text-align:justify'
44	7	28	p class=MsoBodyText style='text-align:justify'	p class=msobodytext style='text-align:justify'
46	7	30	p class=MsoBodyText style='text-align:justify'	p class=msobodytext style='text-align:justify'
48	7	32	p class=MsoBodyText style='text-align:justify'	p class=msobodytext style='text-align:justify'
52	7	36	p class=MsoBodyText style='text-align:justify'	p class=msobodytext style='text-align:justify'
54	7	38	p class=MsoBodyText	p class=msobodytext
57	7	41	/body	/body

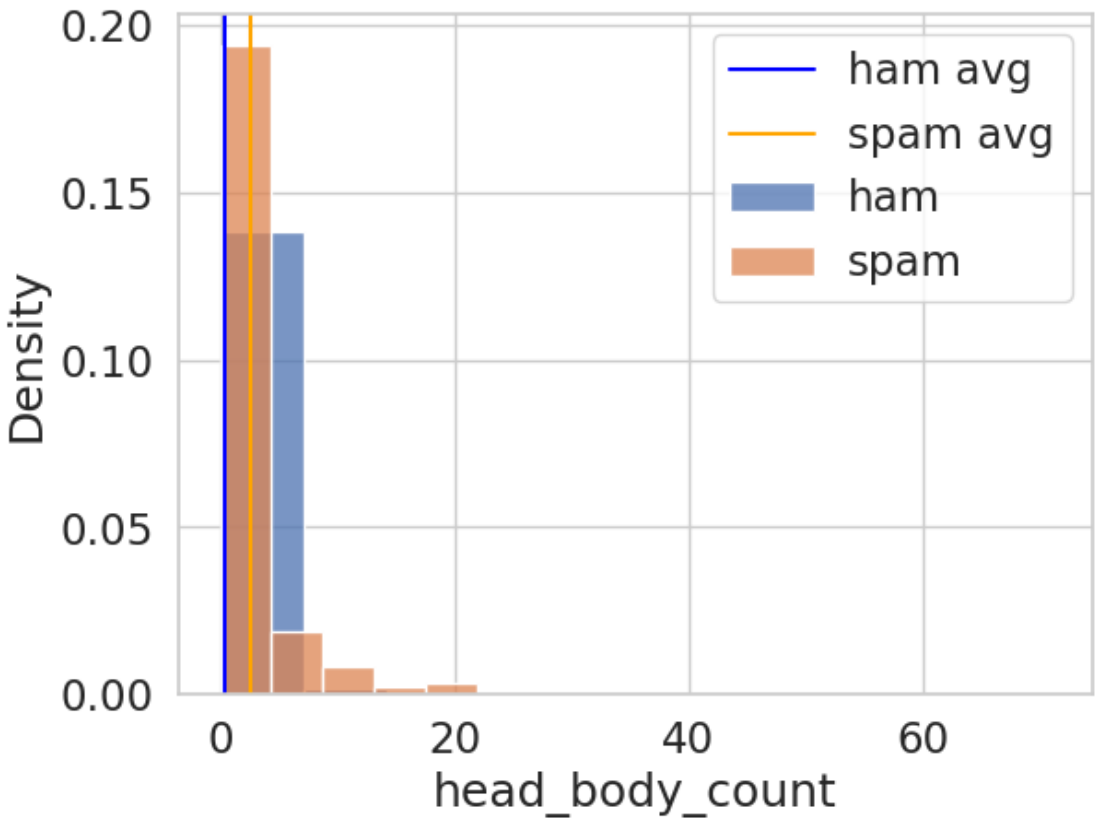
In [35]:
```
def head_body_count(df):
    return len(df[((df.new_lowered.str.contains('body')) | (df.new_lowered.str.contains('head'

df_test = html_stuff[html_stuff.level_0 == 1]
head_body_count(df_test)

hb_counts = html_stuff.sort_values('level_0', ascending = True).groupby('level_0').apply(lambda
# hb_counts
```

In [36]:
```
head_body_count_arr = []

for i in range(len(train)):
    if i in hb_counts.index:
        head_body_count_arr.append(hb_counts[i])
    else:
        head_body_count_arr.append(0)

train['head_body_count'] = head_body_count_arr
```

In [37]:
```
sns.histplot(data = train[train.spam == 0], x = 'head_body_count', stat = 'density', bins = 10
```

```
sns.histplot(data = train[train.spam == 1], x = 'head_body_count', stat = 'density', bins = 10

plt.axvline(np.mean(train[train.spam == 0].head_body_count), label = 'ham avg', color = 'blue'
plt.axvline(np.mean(train[train.spam == 1].head_body_count), label = 'spam avg', color = 'oran

plt.legend()
```

Out[37]: <matplotlib.legend.Legend at 0x7fba2407bf90>



0.6 Attempting Sentiment Analysis

In [38]:
```
file_path = 'vader_lexicon.txt'

df = pd.read_csv(file_path, sep='\t', header=None)
df.columns = ['symbol', 'score', 'score_sd', 'sample_scores']
df.head()
```

Out[38]:

	symbol	score	score_sd	sample_scores
0	$:	-1.5	0.80623	[-1, -1, -1, -1, -3, -1, -3, -1, -2, -1]
1	%)	-0.4	1.01980	[-1, 0, -1, 0, 0, -2, -1, 2, -1, 0]
2	%-)	-1.5	1.43178	[-2, 0, -2, -2, -1, 2, -2, -3, -2, -3]
3	&-:	-0.4	1.42829	[-3, -1, 0, 0, -1, -1, -1, 2, -1, 2]
4	&:	-0.7	0.64031	[0, -1, -1, -1, 1, -1, -1, -1, -1, -1]

In [39]:
```
sentiment_dict = df.set_index('symbol')['score'].to_dict()
```

In [40]:
```
train.email.values[5]
```

Out[40]:
```
"URL: http://diveintomark.org/archives/2002/10/09.html#five\n Date: 2002-10-09T10:25:09-05:00\r
```

In [41]:
```
def strip_to_text(html_content):
    # List of punctuation characters to remove
    punctuation = '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'

    # Using regular expressions to remove HTML tags and URLs
    text = re.sub('<[^<]+?>', '', html_content)   # Remove HTML tags
    text = re.sub(r'http\S+', '', text)           # Remove URLs

    # Removing punctuation
    text = text.translate(str.maketrans('', '', punctuation))

    # Removing tabs and newline characters
    text = text.replace('\t', '').replace('\n', '')

    return text

train['email_clean'] = train['email'].apply(strip_to_text)
train['email_clean'][0]
```

Out[41]:
```
'URL  Date Not supplied  Arts and Letters Daily a wonderful and dense blog has folded up its t
```

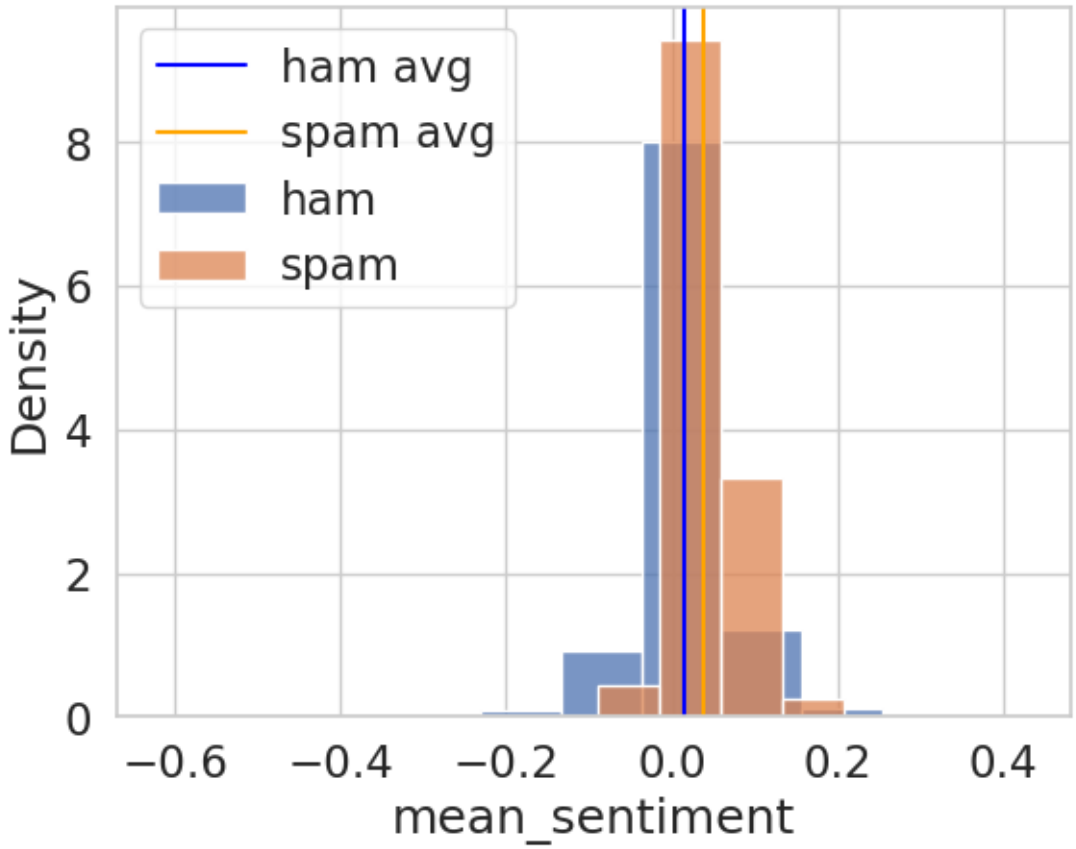
In [42]:
```
def apply_sentiment_score(text):
    sentences = text.split('. ')
    words_in_sentences = [sent.split() for sent in sentences]
    scores = []
    for sentence in words_in_sentences:
        sentence_scores = []
        for wrd in sentence:
            try:
                sentence_scores.append(sentiment_dict[wrd])
            except:
                sentence_scores.append(0)
```

```
            if len(sentence_scores) == 0:
                scores.append(0)
            else:
                scores.append(np.mean(sentence_scores))
        if len(scores) == 0:
            return 0
        else:
            return np.mean(scores)
```

In [43]:
```
train['mean_sentiment'] = train['email_clean'].apply(apply_sentiment_score)
```

In [44]:
```
sns.histplot(data = train[train.spam == 0], x = 'mean_sentiment', stat = 'density', bins = 10,
sns.histplot(data = train[train.spam == 1], x = 'mean_sentiment', stat = 'density', bins = 10,

plt.axvline(np.mean(train[train.spam == 0].mean_sentiment), label = 'ham avg', color = 'blue')
plt.axvline(np.mean(train[train.spam == 1].mean_sentiment), label = 'spam avg', color = 'orang

plt.legend()
```

Out[44]:
```
<matplotlib.legend.Legend at 0x7fba1fd29710>
```



```
In [45]: train.sample(n = len(train), replace = False).head()['spam']
```

```
Out[45]: 2918    0
         4826    1
         7482    1
         6837    1
         4030    0
         Name: spam, dtype: int64
```

```
In [46]: def count_capitals(text):
             count = 0
             for char in text:
                 if char.isupper():
                     count += 1
             return count

         train['num_caps'] = train['email'].apply(count_capitals)
         train
```

```
Out[46]:         id                                            subject  \
         0        0  Subject: A&L Daily to be auctioned in bankrupt…
         1        1  Subject: Wired: "Stronger ties between ISPs an…
         2        2  Subject: It's just too small                  …
         3        3                      Subject: liberal defnitions\n
         4        4  Subject: RE: [ILUG] Newbie seeks advice - Suse…
         …    …                                                  …
         8343  8343            Subject: Re: ALSA (almost) made easy\n
         8344  8344             Subject: Re: Goodbye Global Warming\n
         8345  8345                                  Subject: hello\n
         8346  8346  Subject: Your application is below. Expires Ju…
         8347  8347              Subject: Re: [SAtalk] CONFIDENTIAL\n

                                                        email  spam  length  \
         0     URL: http://boingboing.net/#85534171\n Date: N…     0     359
         1     URL: http://scriptingnews.userland.com/backiss…     0     278
         2     <HTML>\n <HEAD>\n </HEAD>\n <BODY>\n <FONT SIZ…     1     444
         3     Depends on how much over spending vs. how much…     0    1500
         4     hehe sorry but if you hit caps lock twice the …     0    2018
         …                                                   …   …      …
         8343  Thanks for this, I'm going to give them anothe…     0    2287
         8344  Thanks for the link - I'm fascinated by archae…     0    6463
         8345  WE NEED HELP.  We are a 14 year old fortune 50…     1     881
         8346  <html>\n \n \n <HEAD> \n <META charset=3DUTF-8…     1    2723
         8347  On Wed, 2002-08-21 at 06:42, Craig R.Hughes wr…     0     863

               100% more  100% free  head_body_count  \
```

```
0             0          0                0
1             0          0                0
2             0          0                4
3             0          0                0
4             0          0                0
…           …          …              …
8343          0          0                0
8344          0          0                0
8345          0          0                0
8346          0          0                7
8347          0          0                0

                                                email_clean  mean_sentiment  \
0     URL  Date Not supplied  Arts and Letters Daily…        0.065854
1     URL  Date Wed 25 Sep 2002 153310 GMT  Wired1 S…        0.033333
2          A man endowed with a 78 hammer is simply …        0.061538
3     Depends on how much over spending vs how much …       -0.044889
4     hehe sorry but if you hit caps lock twice the …        0.029545
…                                                   …            …
8343  Thanks for this Im going to give them another …        0.029412
8344  Thanks for the link  Im fascinated by archaeol…        0.025324
8345  WE NEED HELP  We are a 14 year old fortune 500…        0.035652
8346                              Your application …        0.038554
8347  On Wed 20020821 at 0642 Craig RHughes wrote  O…       -0.011224

      num_caps
0           22
1           18
2           51
3           26
4           90
…         …
8343        87
8344       167
8345       203
8346       371
8347        37

[8348 rows x 11 columns]
```

In [47]:
```python
def mean_diff(df, feature):
    return np.mean(df[df.shuffled_spam == 0][feature]) - np.mean(df[df.shuffled_spam == 1][fea

def shuffle_and_result(df, feature):
    new_label = 'shuffled_spam'
    df[new_label] = df.sample(n = len(df), replace = False)['spam'].values
    return df[['spam', new_label, feature]]

def ab_testing(df, feature):
    h0 = np.mean(df[df.spam == 0][feature]) - np.mean(df[df.spam == 1][feature])
    diffs = []
    for el in range(1000):
        s_df = shuffle_and_result(df, feature)
```

```
            diffs.append(mean_diff(s_df, feature))
        # plt.hist(diffs)
        # plt.axvline(h0)
        p_value = np.mean(h0 > diffs)
        print("P Value for " + feature + " " + str(p_value))
        return diffs, h0

    diffs, h0 = ab_testing(train, 'mean_sentiment')
```

```
P Value for mean_sentiment 0.0
```

```
In [48]: diffs, h0 = ab_testing(train, 'length')
```

```
P Value for length 0.0
```

```
In [49]: pattern = r'[!"#$%&\'()*+,-./:;<=>?@\[\\\]^_`{|}~]'

    # patten = r'[!?;#$€]'

    train['num_punc'] = train['email'].apply(lambda x: len(re.findall(pattern, x)))
    train.head()
```

Out[49]:

	id	subject	email	spam	length	100% more	100% free	head_body_count	email_clean	mean_sentiment
0	0	Subject: A&L Daily to be auctioned in bankrupt…	URL: http://boingboing.net/#85534171\n Date: N…	0	359	0	0	0	URL Date Not supplied	Arts and Letters Daily…	0.065854
1	1	Subject: Wired: "Stronger ties between ISPs an…	URL: http://scriptingnews.userland.com/backiss…	0	278	0	0	0			
2	2	Subject: It's just too small …	<HTML>\n <HEAD>\n </HEAD>\n <BODY>\n <FONT SIZ…	1	444	0	0	4			
3	3	Subject: liberal defnitions\n	Depends on how much over spending vs. how much…	0	1500	0	0	0			
4	4	Subject: RE: [ILUG] Newbie seeks advice - Suse…	hehe sorry but if you hit caps lock twice the …	0	2018	0	0	0			

```
1  URL  Date Wed 25 Sep 2002 153310 GMT  Wired1 S…        0.033333
2       A man endowed with a 78 hammer is simply …        0.061538
3  Depends on how much over spending vs how much …       -0.044889
4  hehe sorry but if you hit caps lock twice the …        0.029545

   num_caps  shuffled_spam  num_punc
0        22              0        41
1        18              0        40
2        51              0        66
3        26              0        71
4        90              0       127
```

```
In [50]: def cap_ratio(text):
             count = 0
             for char in text:
                 if char.isupper() == True:
                     count += 1
             return count/len(text)

         train['cap_ratio'] = train.email.apply(cap_ratio)

         diff, h0 = ab_testing(train, 'cap_ratio')
```

```
P Value for cap_ratio 0.0
```

```
In [51]: bruh = train.sample(n = 1)
         email = bruh.email
         print(bruh.spam)
         [print(el) for el in email]
```

```
1443    0
Name: spam, dtype: int64
Once upon a time, Chris wrote :

 > On Tue, 2002-10-08 at 10:36, Matthias Saou wrote:
 > > Hi there,
 > >
 > > Two new things today :
 > >
 > > 1) I've had to install a Red Hat Linux 6.2 server because of an old
 > > proprietary IVR software that doesn't work on newer releases :-( So
 > > I've recompiled both the latest apt and openssh packages for it, and
 > > they are now available with a complete "os, updates & freshrpms" apt
 > > repository at apt.freshrpms.net, for those who might be interested.
 >
 > Gack. Did you try 7.3 with the compat-glibc first? Or does it require an
 > antique kernel?
```

```
It requires a 2.2 kernel, plus antique just-about-everything :-/ Real crap!

Matthias

--
Clean custom Red Hat Linux rpm packages : http://freshrpms.net/
Red Hat Linux release 7.3 (Valhalla) running Linux kernel 2.4.18-10acpi
Load : 0.00 0.03 0.00

-------------------------------------------------
RPM-List mailing list <RPM-List@freshrpms.net>
http://lists.freshrpms.net/mailman/listinfo/rpm-list
```

Out[51]: [None]

In [52]:
```
len(train.iloc[1234, :].email_clean)/(len(train.iloc[1234, :].email))
```

Out[52]: 0.8274706867671692

In [53]:
```
def junk_char_ratio(row):
    return len(row.email_clean)/len(row.email)

train['junk_char_ratio'] = train.apply(lambda row: junk_char_ratio(row), axis = 1)
train.head()
```

Out[53]:

	id	subject	email	spam	length	100% more	100% free	head_body_count
0	0	Subject: A&L Daily to be auctioned in bankrupt…	URL: http://boingboing.net/#85534171\n Date: N…	0	359	0	0	0
1	1	Subject: Wired: "Stronger ties between ISPs an…	URL: http://scriptingnews.userland.com/backiss…	0	278	0	0	0
2	2	Subject: It's just too small …	<HTML>\n <HEAD>\n </HEAD>\n <BODY>\n <FONT SIZ…	1	444	0	0	4
3	3	Subject: liberal defnitions\n	Depends on how much over spending vs. how much…	0	1500	0		
4	4	Subject: RE: [ILUG] Newbie seeks advice - Suse…	hehe sorry but if you hit caps lock twice the …	0	2018	0		

```
3          0               0
4          0               0

                                              email_clean  mean_sentiment  \
0  URL  Date Not supplied  Arts and Letters Daily…        0.065854
1  URL  Date Wed 25 Sep 2002 153310 GMT  Wired1 S…        0.033333
2       A man endowed with a 78 hammer is simply …        0.061538
3  Depends on how much over spending vs how much …       -0.044889
4  hehe sorry but if you hit caps lock twice the …        0.029545

   num_caps  shuffled_spam  num_punc  cap_ratio  junk_char_ratio
0        22              1        41   0.061281         0.632312
1        18              0        40   0.064748         0.482014
2        51              1        66   0.114865         0.608108
3        26              0        71   0.017333         0.925333
4        90              0       127   0.044599         0.873637
```

```
In [54]: diffs, h0 = ab_testing(train, 'junk_char_ratio')
```

```
P Value for junk_char_ratio 1.0
```

```
In [55]: X_train.columns
```

```
Out[55]: Index(['Free', 'Ad', 'Offer', 'Credit', 'Save', 'Click here', 'Guaranteed',
                'Money', 'Rates', 'Special', 'Refinance', 'Debt', 'Cash', 'Quote',
                'Easy', 'Loans', 'Secret', 'Limited', 'Removal', 'Promotion',
                'mean_sentiment', 'num_punc', 'junk_char_ratio', 'num_caps', 'is_reply',
                'cap_ratio'],
               dtype='object')
```

```
In [56]: new_df = X_train.copy()
         new_df['spam'] = Y_train
         for col in X_train.columns:
             ab_testing(new_df, col)
```

```
P Value for Free 0.0
P Value for Ad 0.0
P Value for Offer 0.0
P Value for Credit 0.0
P Value for Save 0.0
P Value for Click here 0.0
P Value for Guaranteed 0.0
P Value for Money 0.0
P Value for Rates 0.0
```

```
P Value for Special 0.0
P Value for Refinance 0.0
P Value for Debt 0.0
P Value for Cash 0.0
P Value for Quote 0.0
P Value for Easy 0.0
P Value for Loans 0.0
P Value for Secret 0.0
P Value for Limited 0.0
P Value for Removal 0.0
P Value for Promotion 0.0
P Value for mean_sentiment 0.0
P Value for num_punc 0.0
P Value for junk_char_ratio 1.0
P Value for num_caps 0.0
P Value for is_reply 0.0
P Value for cap_ratio 0.0
```

```
In [57]: def detect_reply(text):
             if 'wrote:\n \n' in text:
                 return 0
             else:
                 return 1

         train['is_reply'] = train.email.apply(detect_reply)
         diffs, h0 = ab_testing(train, 'is_reply')
```

```
P Value for is_reply 0.0
```

0.7 Question 2b

Write your commentary in the cell below.

List of Features That I Implemented: 1. Word selection 2. Sentiment Analysis 3. Number of puncutation symbols 4. Number of extra characters that are not readable words : Total number of characters 6. If an email is a reply 7. Number of capital letters : Total number of characters

Word Selection I first acquired a list of about 237 words. I googled and searched for lists of words that could be useful in searching for good distinguishing words. I used the words_in_text function to figure out how the number of emails each word appears in. I then sorted the words based on the differential, or the percentage difference between the word existing in spam and not spam emails. I selected the top 20 words from this list to use as features for the model.

Sentiment Analysis: I used the vader_leixcon file from HW three to assess the mean sentiment score of each email. My approach was to average the total sentiment of each sentence. I formatted each email so that is removes all of the excess punctuation and URLs, and then I applied the sentiment analysis.

Number of Punctuation Symbols: I summed the number of punctuation symbols in each email.

Junk Character Ratio: Using each cleaned email, I calculated the number of characters in the cleaned email to the uncleaned email. This would highlight all of the extra stuff that's included in formatting a spam email.

Reply Detection: I found that a lot of ham emails have a piece of text that tell who exactly sent an email. I used the syntax to detect ham emails, and implemented this as a True/False column. I could go deeper into this, but for now the model seems fine.

Number of Capital Letters: Spam emails seem to have more capital letters, which might be used to attract the readers attention. I counted the number of captial letters in each email, and compared it to the total number of characters. This might highlight the junk characters/word that might be present in spam emails.

0.8 Feature Performance Analysis:

I wrote some code that performs AB testing on each individual feature. I selected features that have statistically significant p-values to ensure each feature is distinct in their spam and ham performance.

0.9 Question 3: ROC Curve

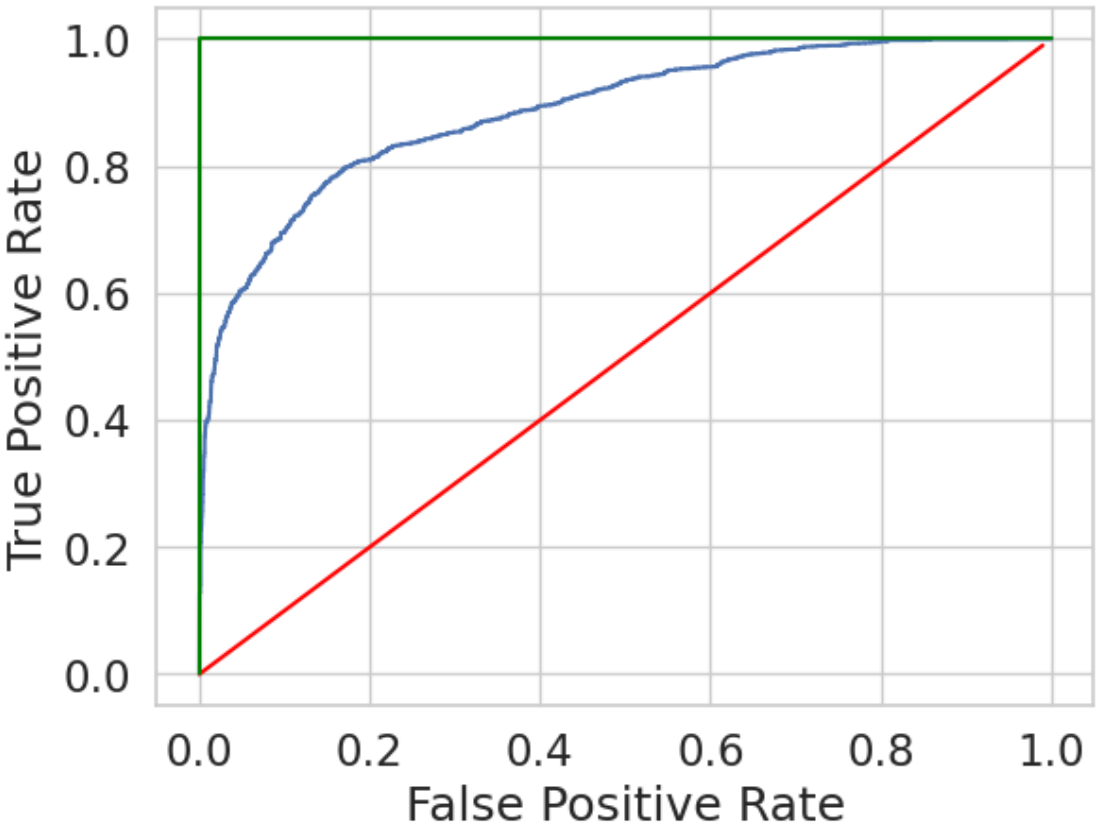
In most cases, we won't be able to get 0 false positives and 0 false negatives, so we have to compromise. For example, in the case of cancer screenings, false negatives are comparatively worse than false positives — a false negative means that a patient might not discover that they have cancer until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

Recall that logistic regression calculates the probability that an example belongs to a particular class. 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 threshold**: 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 Receiver Operating Characteristic (ROC) curve shows this trade-off for each possible cutoff probability. In the cell below, plot an ROC curve for your final classifier (the one you use to make predictions for Gradescope) on the training data. Refer to Lecture 23 to see how to plot an ROC curve.

Hint: You'll want to use the `.predict_proba` method for your classifier instead of `.predict` to get probabilities instead of binary predictions.

In [88]:
```
from sklearn import metrics

#define metrics
y_pred_proba = model.predict_proba(X_train)[::,1]
fpr, tpr, _ = metrics.roc_curve(Y_train,  y_pred_proba)

#create ROC curve
plt.plot(fpr,tpr)

x = np.arange(0, 1, 0.01)
x_ideal = [0, 0, 1]
y_ideal = [0, 1, 1]

plt.plot(x, x, color = 'red')
plt.plot(x_ideal, y_ideal, color = 'green')

plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```


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