[3]:

```
import numpy as np
import pandas as pd
import sys

import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
sns.set(style = "whitegrid",
        color_codes = True,
        font_scale = 1.5)

from datetime import datetime
from IPython.display import display, HTML
```

[5]:

```python
original_training_data = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

# Convert the emails to lowercase as the first step of text processing.
original_training_data['email'] = original_training_data['email'].str.lower()
test['email'] = test['email'].str.lower()

original_training_data.head()
```

[5]:

```
   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
0  url: http://boingboing.net/#85534171\n date: n…     0
1  url: http://scriptingnews.userland.com/backiss…     0
2  <html>\n <head>\n </head>\n <body>\n <font siz…     1
3  depends on how much over spending vs. how much…     0
4  hehe sorry but if you hit caps lock twice the …     0
```

[6]:

```python
# Fill any missing or NAN values.
print('Before imputation:')
print(original_training_data.isnull().sum())
original_training_data = original_training_data.fillna('')
print('------------')
print('After imputation:')
print(original_training_data.isnull().sum())
```

```
Before imputation:
id          0
subject     6
email       0
spam        0
```

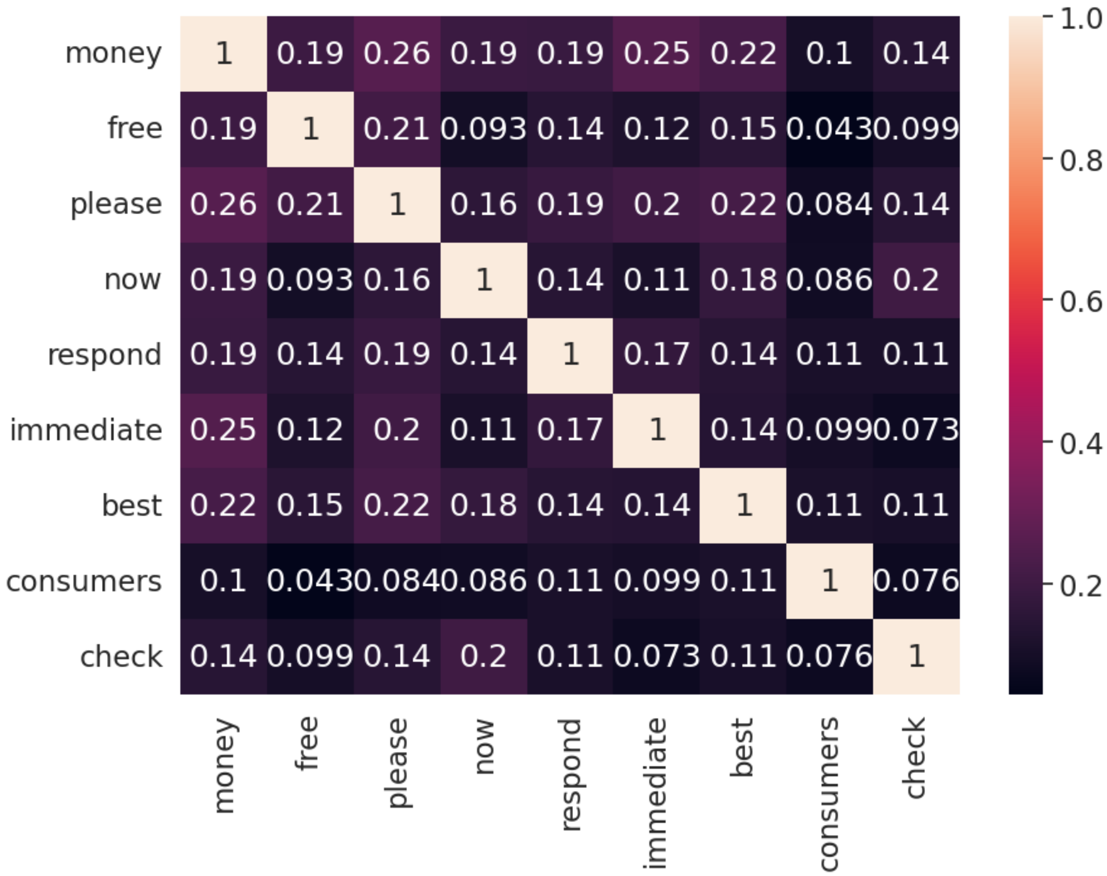
2.4 EDA and Basic Classification

```
[9]: some_words = ['drug', 'bank', 'prescription', 'memo', 'private']

     X_train =
     Y_train =

     X_train[:5], Y_train[:5]
```

```
[9]:



            [0, 0, 0, 0, 0],
            [0, 0, 0, 1, 0]]),
     array([0, 0, 0, 0, 0]))
```

```
[10]: from sklearn.linear_model import LogisticRegression

      simple_model = LogisticRegression()
      simple_model.fit

      training_accuracy =
      print("Training Accuracy: ", training_accuracy)
```

```
Training Accuracy:  0.7576201251164648
```


money 1 0.19 0.26 0.19 0.19 0.25 0.22 0.1 0.14
free 0.19 1 0.21 0.093 0.14 0.12 0.15 0.043 0.099
please 0.26 0.21 1 0.16 0.19 0.2 0.22 0.084 0.14
now 0.19 0.093 0.16 1 0.14 0.11 0.18 0.086 0.2
respond 0.19 0.14 0.19 0.14 1 0.17 0.14 0.11 0.11
immediate 0.25 0.12 0.2 0.11 0.17 1 0.14 0.099 0.073
best 0.22 0.15 0.22 0.18 0.14 0.14 1 0.11 0.11
consumers 0.1 0.043 0.084 0.086 0.11 0.099 0.11 1 0.076
check 0.14 0.099 0.14 0.2 0.11 0.073 0.11 0.076 1
money
free
please
now
respond
immediate
best
consumers
check
1.0
0.8
0.6
0.4
0.2

you may only use the packages we've imported for you in the cell below or earlier in this notebook. In addition, **you are only allowed to train logistic regression models**. No decision trees, random forests, k-nearest-neighbors, neural nets, etc.

Please consider the ideas mentioned above when choosing features. We have not provided any code to do this, so feel free to create as many cells as you need to tackle this task.

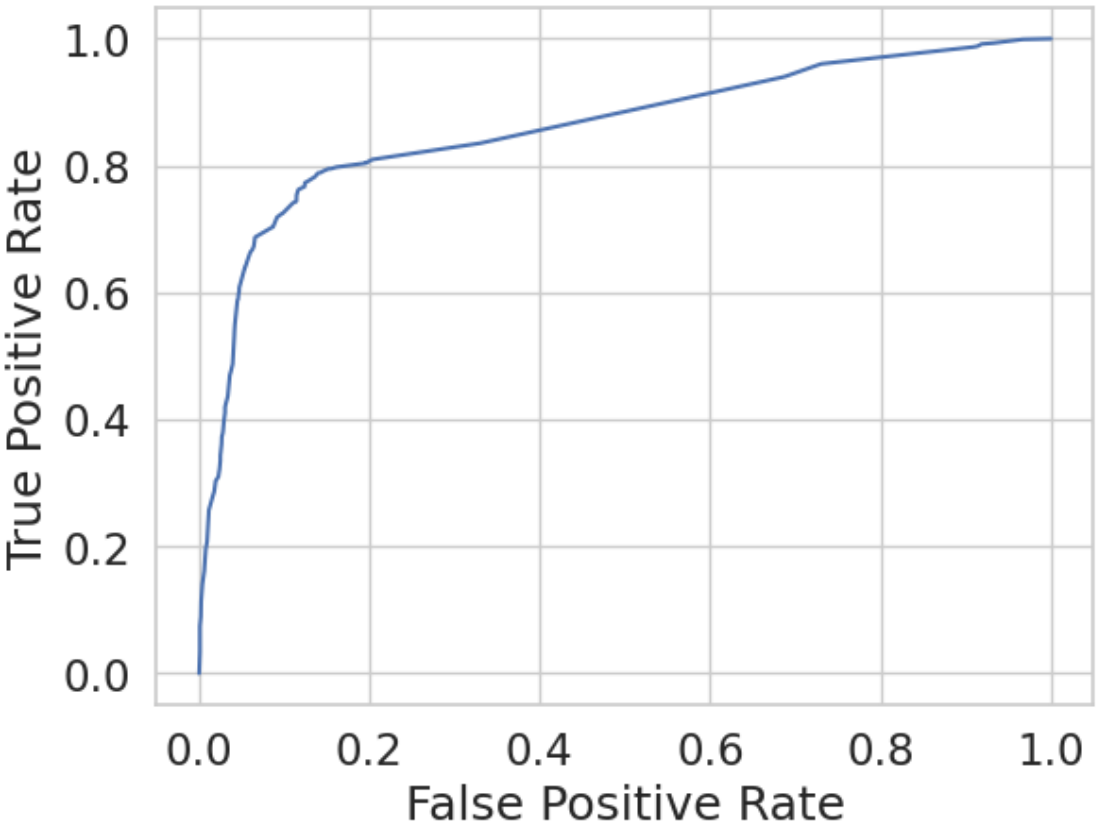
[18]:
```
# import libraries
# You may use any of these to create your features.
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_curve, confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn.decomposition import PCA
import re
from collections import Counter
```

[19]:
```
# Define your processing function, processed data, and model here.
# You may find it helpful to look through the rest of the questions first!
#xt = train[['email length']]
#yt = train['spam']
some_words = [redacted]

xt = [redacted]
yt =
model = LogisticRegression()
model.fit(xt, yt)
```

[19]:
```
LogisticRegression()
```

[24]:
```
#credits to lecture 23 coding for the skeleton to my code below
def predict_threshold(model, X, T):
    [redacted]
    return

def tpr_threshold[redacted]


def fpr_threshold[redacted]


thresholds = np.linspace(0, 1, 100)

tprs = [redacted]
fprs =

plt.plot(fprs,tprs)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

[24]:
```
Text(0, 0.5, 'True Positive Rate')
```


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