0.1 Question 1

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 looking through my own spam folder and noticed many common phrases and combinations of words that generally advertised discounts or limited time offers, which is why most of my words fit that general theme. I then expanded to include features that had anything to do with money since I also noticed that emails from insurance companies were also incredibly common so I used words fit those general themes. Additionally I noticed that spam emails tended to use much more punctuation to make their product or service seem more desirable so characters such as '!', '<' and '>' proved to be very effective.

2. Initially trying to create a theme behind some of the words I chose proved very effective, however I noticed that some themes worked better than others. For example, words that tended to adevertise a sale such as 'limited', 'off' and 'offer' proved incredibly effective. Additionally words that tended to advertise ways to make or save money such as 'insurance', 'debt' or 'paid' also proved to be effective. Other words that implied that the emails were subscription based such as 'subscribe' or 'unsubscribe' proved to be generally very ineffective and at time lowered my accurcacy which was a bit surprising.

3. One thing that really surprised me was that words that tended to advertise anything regarding romance proved to be ineffective. In extreme cases with words such as 'hot', 'singles', 'near', 'you' and 'sexy' proved to actually significantly lower my accuracy. This surprised me as I assumed words along these lines, especially in such a combination would be fairly common in certain types of spam emails.

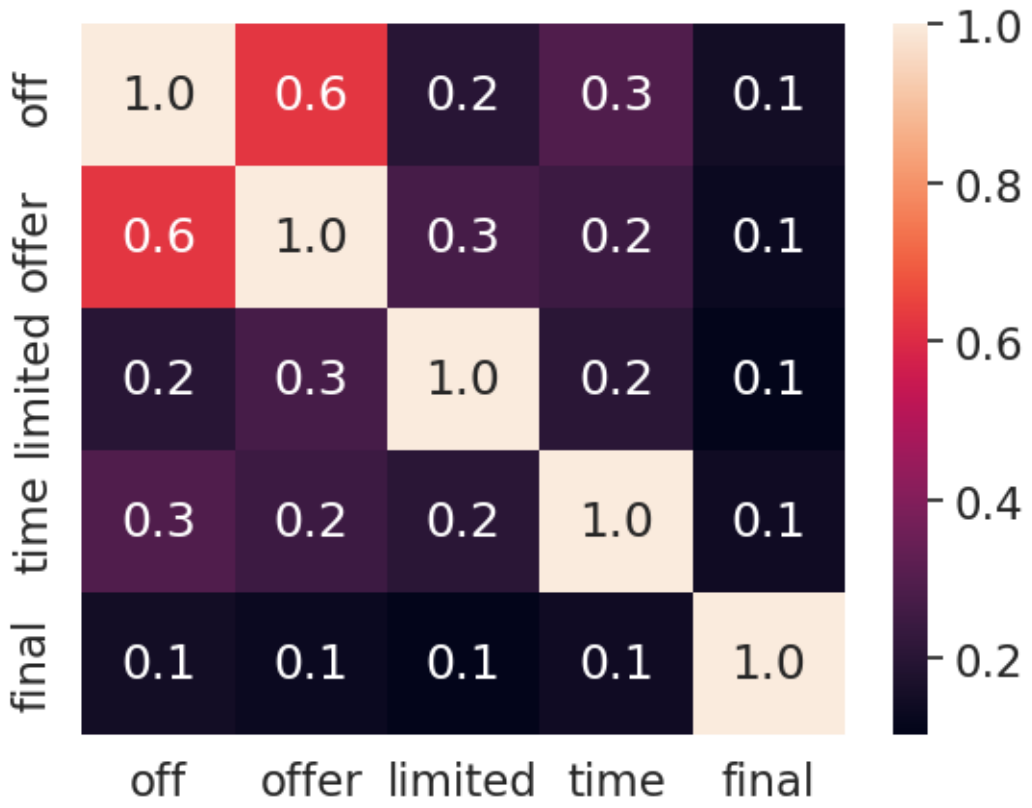
0.2 Question 2a

Generate your visualization in the cell below.

In [13]:
```
#words = ['viagra', 'warning', 'urgent', 'cheap', 'shop','get','hot','near','home']
#master list
```

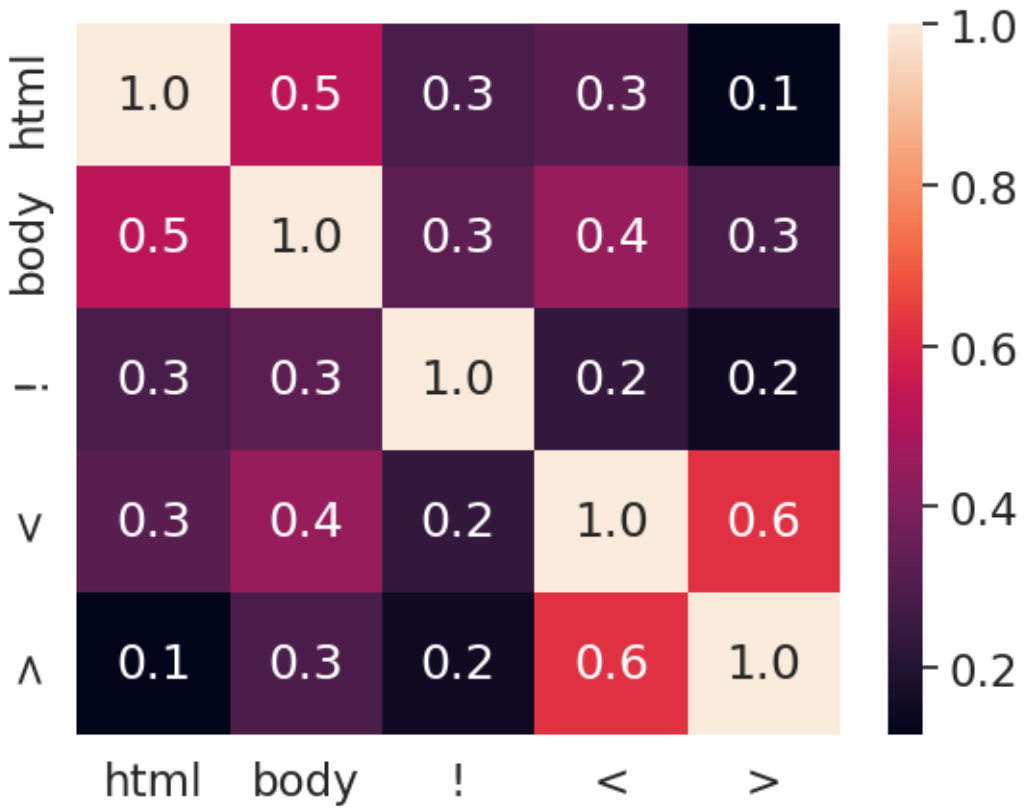
In [14]:
```
words_subarray2 = ['off','offer','limited','time','final']
words_subdf2 = pd.DataFrame(words_in_texts(words_subarray2, train['email']), columns = words_su
sns.heatmap(data = words_subdf2, annot = True, fmt=".1f")
```

Out[14]: <Axes: >



In [15]:
```
words_subarray = ['html','body','!','<','>']
words_subdf = pd.DataFrame(words_in_texts(words_subarray, train['email']), columns = words_suba
sns.heatmap(data = words_subdf, annot = True, fmt=".1f")
```

Out[15]: <Axes: >



In [16]:
```
words_subarray3 = ['brand','new','own','boss','work']
words_subdf3 = pd.DataFrame(words_in_texts(words_subarray3, train['email']), columns = words_su
sns.heatmap(data = words_subdf3, annot = True, fmt=".1f")
```

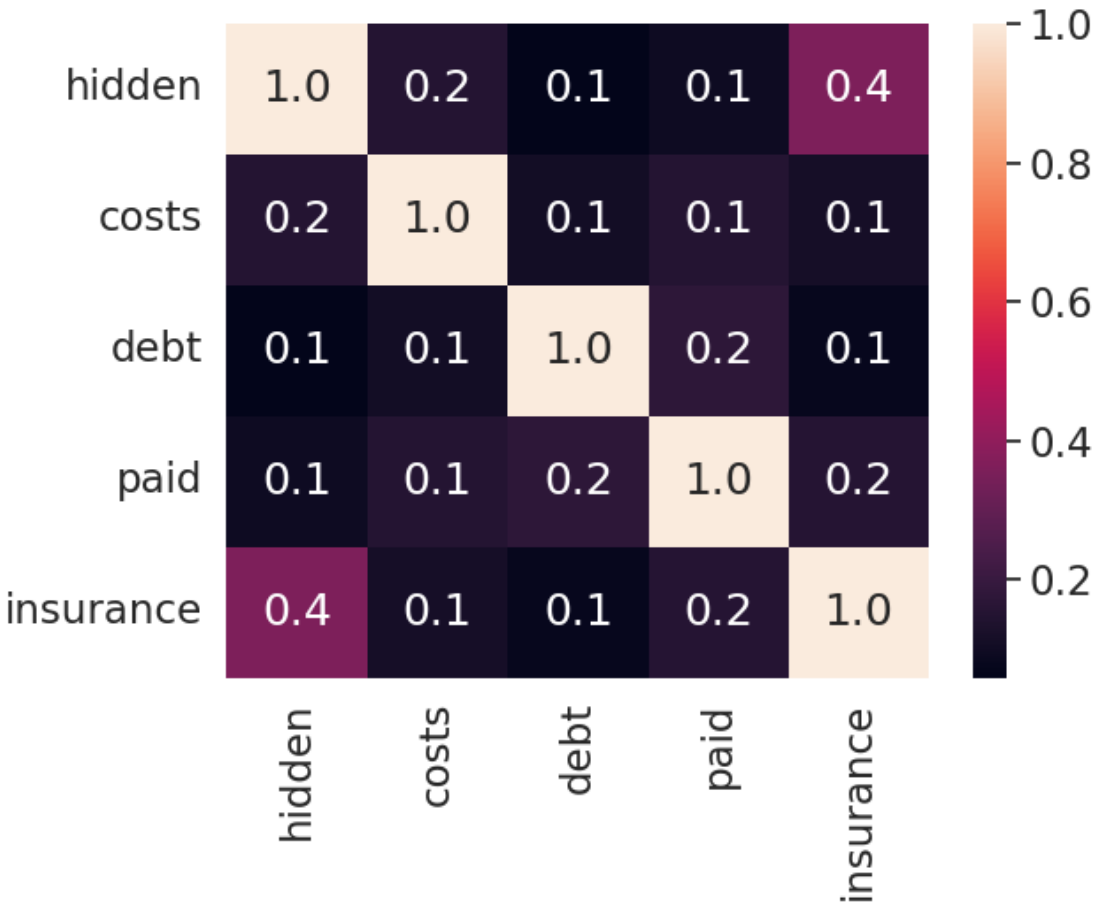
Out[16]: <Axes: >

In [17]:
```
words_subarray4 = ['sign','up','call','now','order']
words_subdf4 = pd.DataFrame(words_in_texts(words_subarray4, train['email']), columns = words_su
sns.heatmap(data = words_subdf4, annot = True, fmt=".1f")
```

Out[17]: <Axes: >

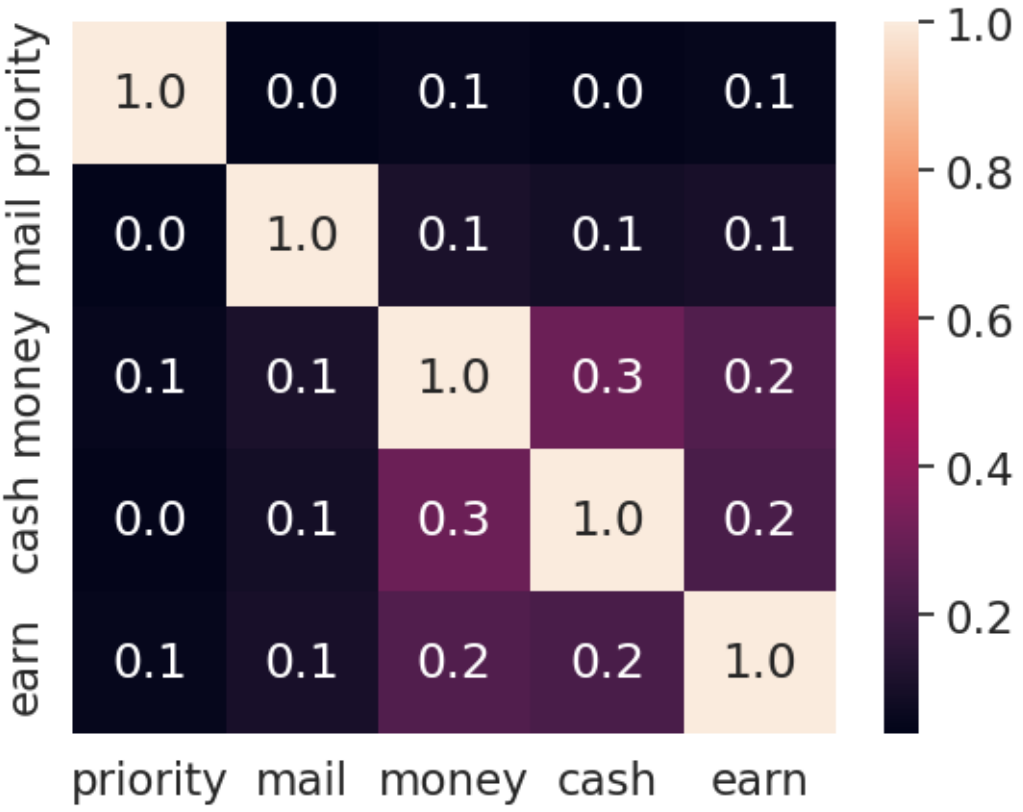

```
In [18]: words_subarray5 = ['hidden','costs','debt','paid','insurance']
         words_subdf5 = pd.DataFrame(words_in_texts(words_subarray5, train['email']), columns = words_su
         sns.heatmap(data = words_subdf5, annot = True, fmt=".1f")
```

```
Out[18]: <Axes: >
```



In [19]:
```
words_subarray6 = ['priority','mail','money','cash','earn']
words_subdf6 = pd.DataFrame(words_in_texts(words_subarray6, train['email']), columns = words_su
sns.heatmap(data = words_subdf6, annot = True, fmt=".1f")
```

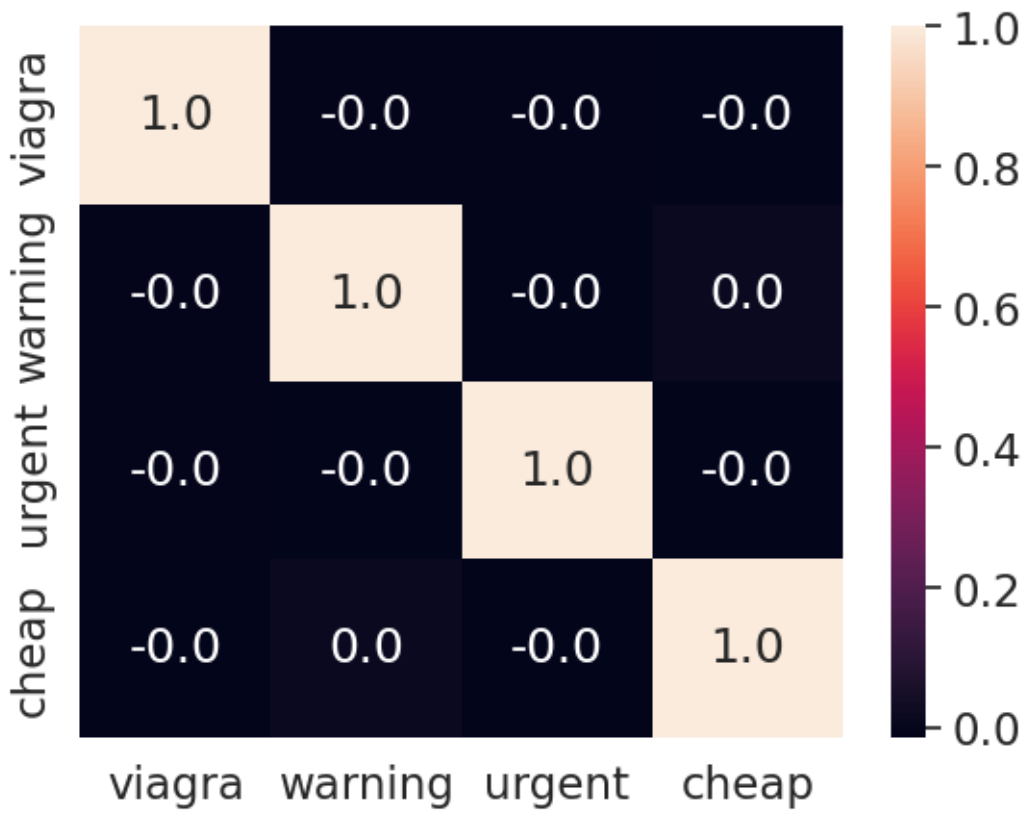
Out[19]: <Axes: >



In [ ]:

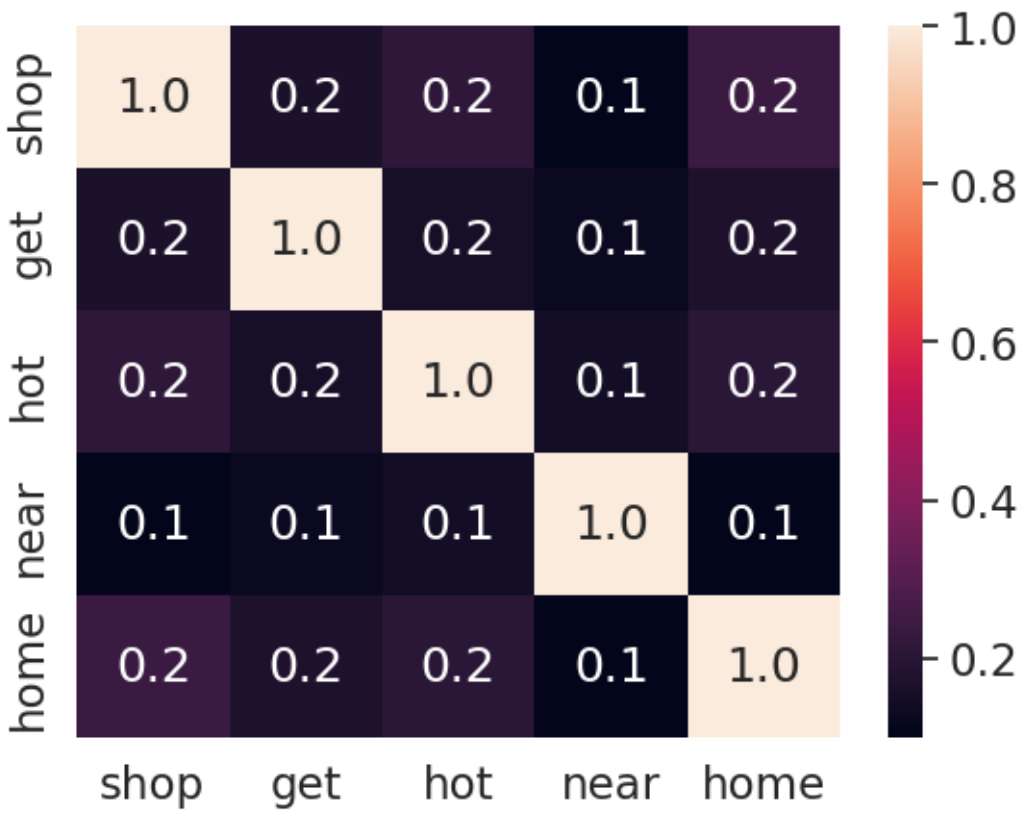
In [21]:
```
words_subarray7 = ['viagra', 'warning', 'urgent', 'cheap']
words_subdf7 = pd.DataFrame(words_in_texts(words_subarray7, train['email']), columns = words_su
sns.heatmap(data = words_subdf7, annot = True, fmt=".1f")
```

Out[21]: <Axes: >



In [22]:
```
words_subarray8 = ['shop','get','hot','near','home']
words_subdf8 = pd.DataFrame(words_in_texts(words_subarray8, train['email']), columns = words_su
sns.heatmap(data = words_subdf8, annot = True, fmt=".1f")
```

Out[22]: <Axes: >


shop
get
hot
near
home
1.0 0.2 0.2 0.1 0.2
0.2 1.0 0.2 0.1 0.2
0.2 0.2 1.0 0.1 0.2
0.1 0.1 0.1 1.0 0.1
0.2 0.2 0.2 0.1 1.0
shop
get
hot
near
home
1.0
0.8
0.6
0.4
0.2

0.3 Question 2b

Write your commentary in the cell below.

I chose to use heatmaps to help me see the correlation between certain words in the emails list. This helped me understand whether or not certain words worked well together and to what extent they did. For example, from the first heatmap we can see that both '<' and '>' had a high correlation together which makes sense as they are the primary characters used in html tags, while on the other hand words such as 'viagra' and 'urgent' really did not work well together at all. This visualization may help me better understand and filter out words that are less effective than others.

0.4 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, 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 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 a ROC curve for your final classifier (the one you use to make predictions for Gradescope) on the training data. Refer to Lecture 19 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 [31]:
```
from sklearn.metrics import roc_curve

        y_prob = model.2.predict_proba(x_train)[:, 1]
        fp, tp, threshold = roc_curve(y_train, y_prob)
        plt.figure(figsize = (8,6))
        plt.plot(fp, tp, color = 'blue', lw = 2)
        plt.xlabel('False Positive')
        plt.ylabel('True Positive')
        plt.title('')
```

Out[31]:
```
Ellipsis
```