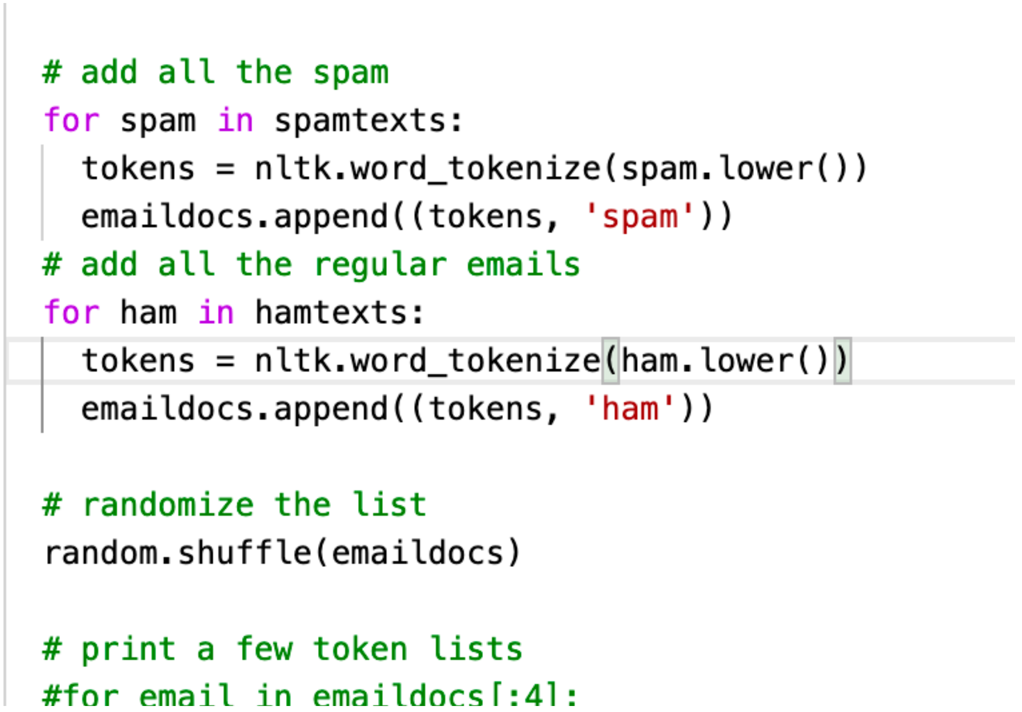
IST 664
Spam-Ham Detection
Seshu Miriyala

Data Processing

The data source contains emails presorted into spam and ham categories. We are using the physical separation of emails into spam and ham folders to identify the category.
When loading the emails, the folder name is taken as the category they belong as shown in the following screenshot.

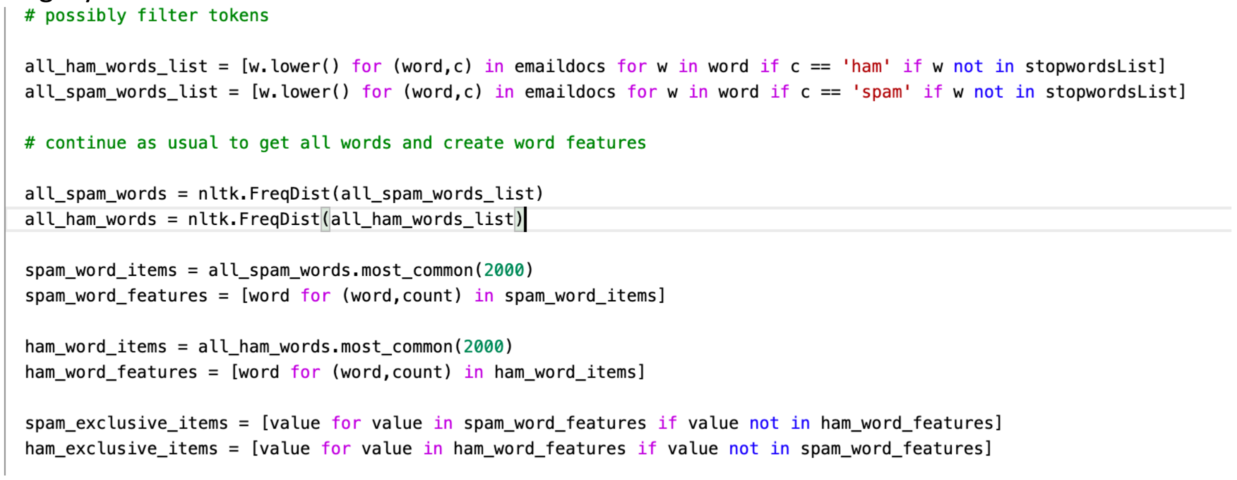


```
# add all the spam
for spam in spamtexts:
  tokens = nltk.word_tokenize(spam.lower())
  emaildocs.append((tokens, 'spam'))
# add all the regular emails
for ham in hamtexts:
  tokens = nltk.word_tokenize(ham.lower())
  emaildocs.append((tokens, 'ham'))

# randomize the list
random.shuffle(emaildocs)

# print a few token lists
#for email in emaildocs[:4]:
```

Once the data is loaded into variable, I am extracting the ham tokens and spam tokens into separate variables. And then filtering the stop words for individual tokens.
I have included word “Subject” into the stop words list as almost all the emails have the word.

Most common words

I have taken the top 2000 most common words from ham and spam categories each. And then excluded the common words to get the unique set of words that could be used to identify the category of the email.



```
# possibly filter tokens

all_ham_words_list = [w.lower() for (word,c) in emaildocs for w in word if c == 'ham' if w not in stopwordsList]
all_spam_words_list = [w.lower() for (word,c) in emaildocs for w in word if c == 'spam' if w not in stopwordsList]

# continue as usual to get all words and create word features

all_spam_words = nltk.FreqDist(all_spam_words_list)
all_ham_words = nltk.FreqDist(all_ham_words_list)

spam_word_items = all_spam_words.most_common(2000)
spam_word_features = [word for (word,count) in spam_word_items]

ham_word_items = all_ham_words.most_common(2000)
ham_word_features = [word for (word,count) in ham_word_items]

spam_exclusive_items = [value for value in spam_word_features if value not in ham_word_features]
ham_exclusive_items = [value for value in ham_word_features if value not in spam_word_features]
```

Features selection

I used the following methods to build the featuresets that could be used by the classification model.

Most common exclusive words:

I used the exclusive ham and spam words that I extracted using the method mentioned in the above data processing section to generate the featureset. As the words like http, url are most commonly used in spam emails, they could be used to identify the emails as spam easily.

Count of special characters:

I used the frequency of each individual special character in the email as one of the featureset. Generally, spam emails tend to have more special characters like #,@ than ham emails.

Count of all special and non-special characters

I used the count of all special characters and non-special characters in the email as another featureset.
Generally, spam emails tend to have more special characters than ham emails.

Count of numbers:

I added a feature set called "numberCount" to represent the numerical words present in each email.
In this corpus, the ham emails tend to have more numerical words than spam emails. So I added it as a feature.

Words longer than 6 characters and shorter than 3 characters:

I noticed that the spam emails have more lengthy words than ham words. So included a feature of frequency of words more than 6 characters and frequency of words less than 3 characters. And also included a featureset to capture the average length of the words in each email.

Count of POS tags:

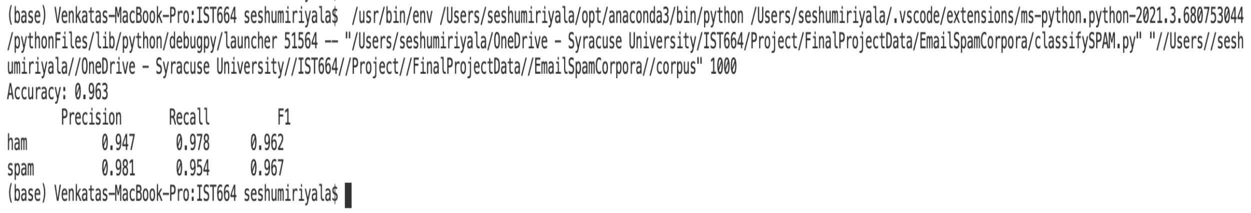
I then used the frequency of POS tags like Nouns, Verbs, Adjectives, Adverbs in each email as the features.
I assumed spam emails to have more adjective phrases than ham emails. But I was wrong as the accuracy was dropped before and after this feature set is added.

Experiments

Naïve Bayes classification algorithm

With all features

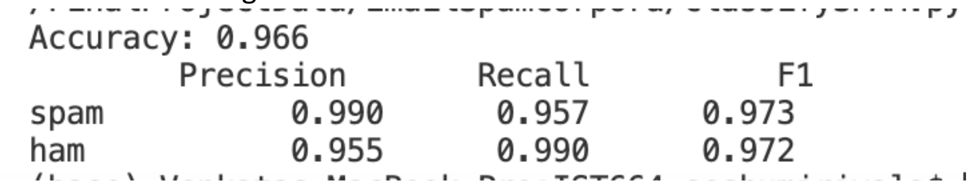
Accuracy of the classifier with all features included is 96.3%. I used the NLTK's naïve Bayes classification algorithm and generated the accuracy



```
(base) Venkatas-MacBook-Pro:IST664 seshumiriyala$  /usr/bin/env /Users/seshumiriyala/opt/anaconda3/bin/python /Users/seshumiriyala/.vscode/extensions/ms-python.python-2021.3.680753044
/pythonFiles/lib/python/debugpy/launcher 51564 -- "/Users/seshumiriyala/OneDrive - Syracuse University/IST664/Project/FinalProjectData/EmailSpamCorpora/classifySPAM.py" "//Users//sesh
umiriyala//OneDrive - Syracuse University//IST664//Project//FinalProjectData//EmailSpamCorpora//corpus" 1000
Accuracy: 0.963
        Precision       Recall          F1
ham          0.947       0.978       0.962
spam         0.981       0.954       0.967
(base) Venkatas-MacBook-Pro:IST664 seshumiriyala$ 
```

With the POS features removed

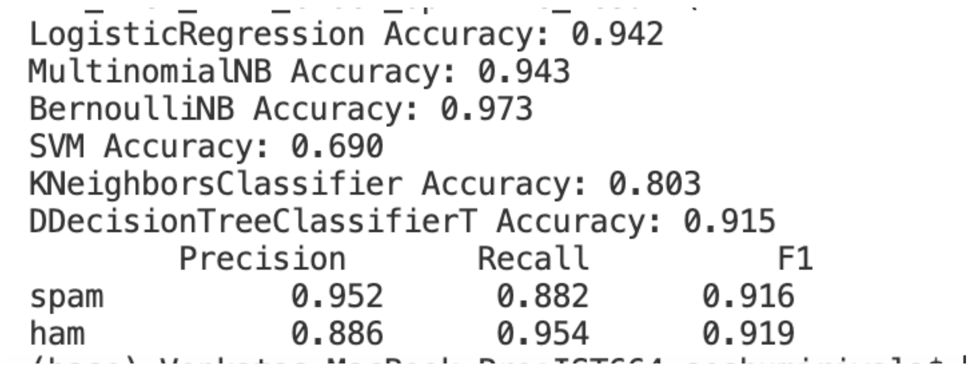
When I remove the POS features, the accuracy increased a little. May be there is some confusion in the algorithm when the POS counts are included.



```
Accuracy: 0.966
        Precision       Recall          F1
spam          0.990      0.957       0.973
ham           0.955      0.990       0.972
```

Comparing different algorithms in scikit learn

I compared the different classification algorithms in the scikit learn and noted the accuracy with 5-fold cross validation.

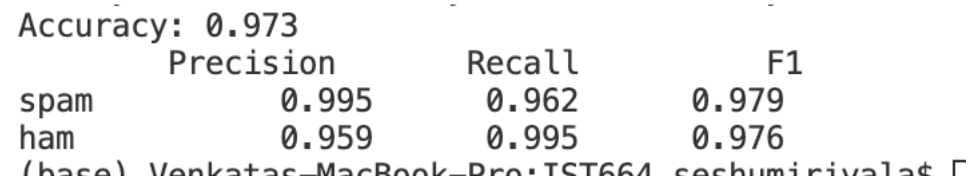


```
LogisticRegression Accuracy: 0.942
MultinomialNB Accuracy: 0.943
BernoulliNB Accuracy: 0.973
SVM Accuracy: 0.690
KNeighborsClassifier Accuracy: 0.803
DDecisionTreeClassifierT Accuracy: 0.915
        Precision       Recall          F1
spam          0.952      0.882       0.916
ham           0.886      0.954       0.919
```

Based on the accuracies, Naive Bayes classifier for multivariate Bernoulli models gives the most accuracy.

Final numbers

Using the multivariante Bernoulli model the precision recall and F1 scores are as below.



```
Accuracy: 0.973
        Precision       Recall          F1
spam          0.995      0.962       0.979
ham           0.959      0.995       0.976
(base) Venkatas-MacBook-Pro:IST664 seshumiriyala$
```