Report

Team ID: 005

Name: Samiul Ehsan Chowdhury

Approach
Spam emails contains many key words (congratulations, Nigeria, million etc) in its body. Hence, my classifying spam and ham emails I looked at email body words.

I did not use email subjects as they have low significance and most of the spam emails, subject words and body words were similar.

During preprocess, I removed all the special characters and only kept words.

In train function, it calculates all the spam or ham words and most frequent words were taken based on those words.

During prediction, the function calculates how many words matches in the given email with the frequent spam and ham words from training data. If more matches are found with ham emails, it predicts test email as ham or else as spam.

The word percentage is given at 7% but it can be changed accordingly. The model is then trained using the entire dataset and then it is used to predict for the test dataset.

Experimental setup
After the preprocessing, the dataset contains columns like Label and email. Using sklearn test_train_split, the dataset was divided into train and validation sets. Validation size is set to 20.

Results
This model is able to detect a spam E-mail 80.7% of the time.

Of the times that the model does say that the email is spam it is right 80.7% of the time.

Even though I have given the macro- and micro-averaged results, I am choosing The micro-averaged results, because the training data provided was very balanced in an almost 50-50 split.

Model	Accuracy	Macro Precision	Micro Precision	FPR (Macro)	FPR (Micro)
Simple Email Classifier	*0.807*	*0. 0.82*	*0.807*	*0.305*	*0.040*

Discussion (max. 200 words)

Besides using email body, other features such as email recipients, sender can be used. I have used the email recipients word length, although there were no significant improvement on accuracy.
Hence, further improvement is required and can be made filtering return path and cc copy list.

Advanced classifier

As observed from the table underneath, the precision of both Naive Bayes and SVM expanded utilizing TF-IDF as opposed to simply TF or word check.

Out of the two classifiers put under a magnifying glass, SVM gave the best outcomes.
I also tested with Decision Tree Classifier as it a good classifier for classification.
When testing Decision Tree classifier, it demonstrated better outcomes.

Since its an email classifier, the false positive rate can be utilized to decide how great the classifier performed, and Decision tree showed signs of improvement.
Results
At first the data was divided to validation set just for testing.
I used the train_test_split function from sklearn to split the training data into a train and a validation set.
The data was split as 80% train and 20% validation. After validation, actual test data used, and results are shown below:

Algorithm	Term weighting	Acc.	Prec.	FPR
Naive Bayes	Count	*0.867*	*0.928*	*0.086*
Naive Bayes	TF	*0.89*	*0.843*	*0.24*
Naive Bayes	TF-IDF	*0.907*	*0.866*	*0.205*
SVM	Count	*0.954*	*0.943*	*0.079*
SVM	TF	*0.941*	*0.956*	*0.057*
SVM	TF-IDF	*0.947*	*0.934*	*0.092*
Decision Tree	Count	*0.996*	*0.998*	*0.003*
Decision Tree	TF	*0.997*	*0.997*	*0.004*
Decision Tree	*TF-IDF*	*0.997*	*0.997*	*0.003*

Experimental approaches (max. 500 words)

Approaches taken for the purpose of email classifications are as follows:
- At first the whole email was extracted using email Library function. I used email body first as it contributes most of email, it was processed with stemming, removing stopwords and foreign words.

Furthermore I started extracting features, such as “Subject” and “return-path”, ‘Body word count’.
Many emails do not have body, so subject and return path are valuable additions for determining.

- From “subject”, I tried to count number of special characters as I found out that spams contain many characters while hams don’t.

- For “return-path” I have converted them as Boolean ‘1’ or ‘0’. Most of the cases spams do not have return path.

- For “body word” I have taken length of each body to determine.
- Alternative algorithms used for this classification was Decision Tree.

Text pre-processing

At first, I extracted the body from the content of the file provided. Then cleaned the body of the email i.e removal of any stop words as they don’t contribute towards similarity, then stemmed the word to their lowest form to make it easier for classifier to detect similarities, Removing any word which is not english and coverting all the words to lower case.

Results

The Features extracted from the data to obtain the results are listed as follow.
It can be seen from the results that one long string yielded with Count.

- Feature1: Return-path
- Feature2: Special character length in subject
- Feature3: Body word count

Algorithm	Features	Acc.	Prec.	FPR
Naïve-Bayes	*Feature1*	*0.897*	*0.932*	*0.086*
SVM	*Feature1*	*0.987*	*0.989*	*0.013*
Decision Tree	*Feature1*	*0.988*	*0.989*	*0.014*

Naïve-Bayes	*Feature2*	*0.880*	*0.904*	*0.126*
SVM	*Feature2*	*0.961*	*0.965*	*0.04*
Decision Tree	*Feature2*	*0.945*	*0.947*	*0.07*

Naïve-Bayes	*Feature3*	*0.886*	*0.926*	*0.085*
SVM	*Feature3*	*0.933*	*0.944*	*0.07*
Decision Tree	*Feature3*	*0.942*	*0.956*	*0.057*

Naïve-Bayes	*all Feature*	*0.897*	*0.907*	*0.12*
SVM	*all Feature*	*0.933*	*0.991*	*0.01*
Decision Tree	*all Feature*	*0.988*	*0.989*	*0.014*

Discussion (max. 300 words)

It is seen that in all cases Naïve-bayes performed poorly than other classifiers. In general, Decision Tree performed best.

Moreover, taking “Subject count” as feature did not help much rather lower the predictions.

Final result for Kaggle was taken using “term count”. The score was 0.97500. “TF-IDF” didn’t increase score, maybe taking new features can improve the score.

In both cases, Decision Tree performed well classifying the emails.