Anomaly Detection Challenge 4

Vishal Bhalla Matriculation No: 03662226
Nithish Raghunandanan Matriculation No: 03667351

Spam Detection

January 11, 2016

1 Abstract

The goal of this challenge was to implement a machine learning algorithm (Binary Classification) to classify email messages into Spam or Ham. The email messages are in the format of an .eml file as defined in RFC822 in accordance to the recent standard of email, i.e., MIME (Multipurpose Internet Mail Extensions) in RFC2045-2049. The Training set consisted of the email messages along with the label identifying whether it was a Spam/Ham message.

2 Data Preprocessing & Analysis

Fields of Interest
We analyzed the distribution of different fields and their corresponding characteristics. Out of the many fields in each email, we found the Sender information, the date of the email, the message subject and body to be interesting. However, processing each of these fields is cumbersome due to the different formats of email. In particular, the email body is either of plain text or MIME type and contained multiple sections in it. We also took the SpamAssassinator classifications in the subject field that were present for some of the Emails. The presence of the PGP signature in the Email body was also considered.

3 Feature Engineering

There were many qualitative features in the data set which we had to map to numerical features. So we arrived at different metrics to compute the Spamminess/Hamminess of Messages.

3.1 Spam/Ham Measure of an email

The feature vectors for a message subject and body are encoded based on the Spam Measure of its constituent words. For each word, we constructed a dictionary of spamminess of the word [1] from the training set and the Spamminess of a word is defined as

$$\textbf{Spamminess(word)} = \frac{Number\ of\ occurences\ of\ that\ word\ in\ a\ SPAM\ email}{Total\ \ Occurrences\ of\ that\ word\ in\ both\ SPAM\ \&\ HAM\ emails}$$

Finally, to calculate the overall Spamminess of a message we multiply the individual words' Spamminess. Therefore,

$$\textbf{Message Spamminess} = \prod_{w \in Message} Spamminess(w)$$

Hamminess of a word is actually the complement of the word's spamminess.
Hamminess(word) = 1 - Spamminess(word)
Similarly, to calculate the overall Hamminess of a message we multiply the individual words' Hamminess. Therefore,

$$\textbf{Message Hamminess} = \prod_{w \in Message} Hamminess(w)$$

Variations

1. **Spamminess/Hamminess of Top Spam/Ham Words**
   The spamminess/hamminess values of the K most extreme words in the Email are considered to compute the spamicity of the message.

2. **Spaminess of Top K Spam Words in Emails**
   The spamminess values of the K most spam words in the Email are considered to compute the spamicity of the message.

3. **Ratio of Spam Words**
   The ratio of the Spam words to the total number of the words in the Email are considered.

3.2 Combination of Features

We tried incorporating the email subject, body & the combined fields of subject & body as features in all the classifiers. We also tried incorporating all features by considering combined fields of subject & body including the message length, presence of special words, capital lettered words & PGP with Spaminess as a metric in Random Forest classifier. We used the same features but with another set of metric functions to define the Spammicity of words i.e Ratio of Spam words & Top K Spam metrics. We also applied Static Rules after the each classifier.

4 Static Rules

4.1 Static Rule 1

There are SpamAssassin classifications in the subject for some of the emails. These emails are classified correctly to a high degree in the training data by the commercial spam filter (SpamAssasin). So, if the SpamAssassin classifies an email as spam in the test data, the rule classifies the email as spam.

4.2 Static Rule 2

Some of the emails contain PGP signatures in the body which confirms the identity of the sender. It is not present for the spam mails. So, if the email contains the PGP signature, the rule classifies the email as ham.

4.3 Static Rule 3

On checking the dates of the Emails, some of them were sent in the extreme past(0102) or extreme future(3601). These are spam in the training set. So, if the email date is not within the range 2000 & 2010, it is classified as spam.

4.4 Static Rule 4

Some of the emails have the sender information blanked out. There is a high possibility of these being spam. So, this rule predicted emails with no sender information as spam.

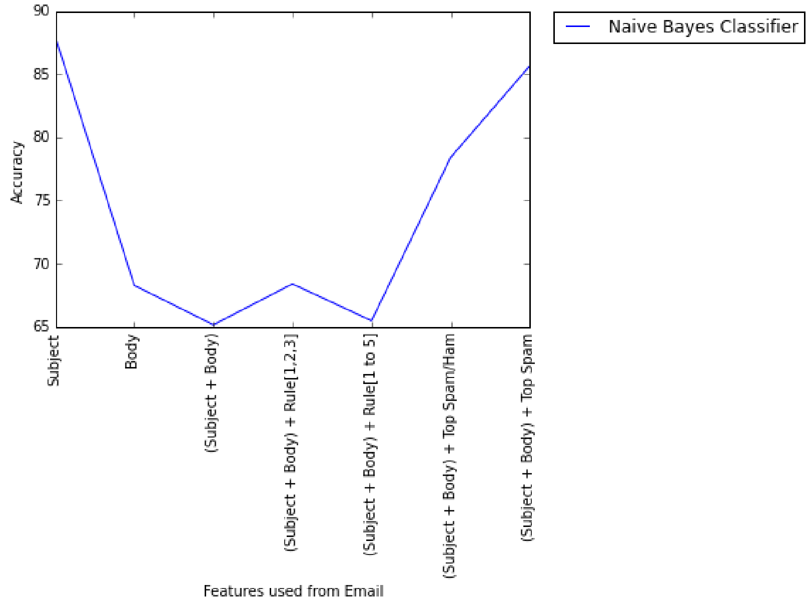
4.5 Static Rule 5

Usually, spam mails contain a lot of capitalized words in them. This rule classifies a mail as spam if there are capitalized words greater than a threshold in the email.

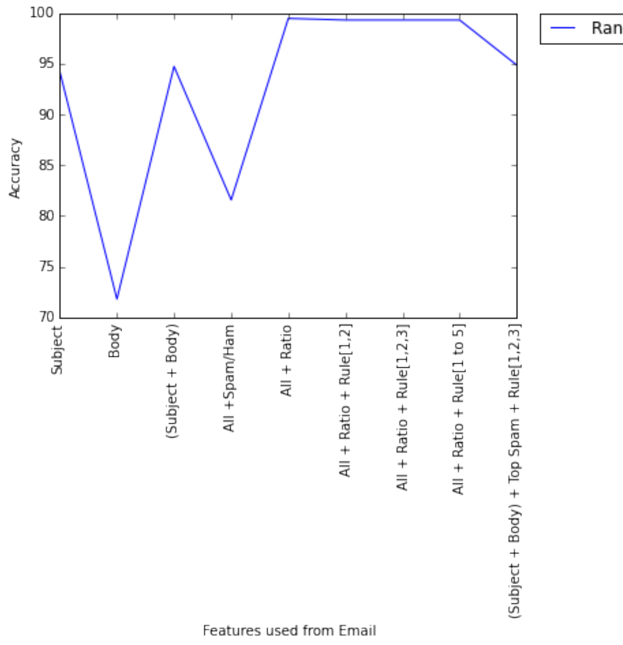
5 Model

We tried fitting different models to our data viz. Random Forest, Ada Boost, Naive Bayes, Vowpal Wabbit and KNN Classifiers. Some of the observations and conclusions during Model Selection over the cross-validation dataset are as below:

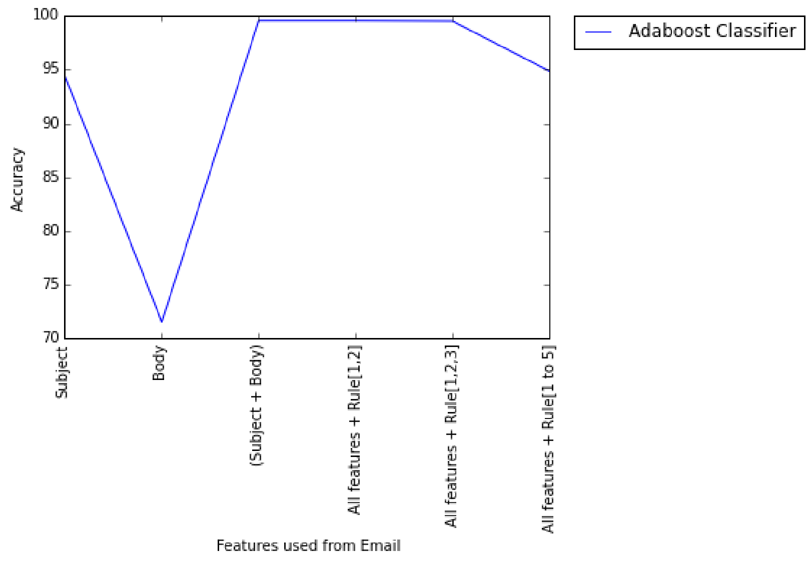
1. We used Random Forest with Entropy and 200 trees as a criterion.
2. Naive Bayes
   We tried Naive Bayes with the classification criteria of an email being classified as spam if its spamminess was more than its hamminess in the case of spamminess/hamminness being taken as feature. In the case of top k spam words as the feature, the spamminness



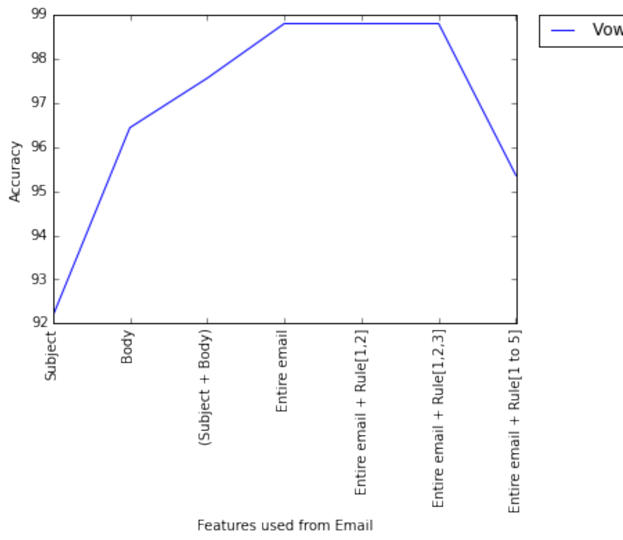
(a) Naive Bayes



(b) Random Forest



(c) Adaboost



(d) Vowpal Wabbit

Figure 1: Classification Models

of the mail being higher than a high threshold(0.7-0.8) was the criteria for a mail to be classified as spam. The top spam words as feature performed better on the combination of the email body & subject as it reduced the probability of a mail being falsely classified as ham due to the presence of a large number of hammy words(Example:the, you and, etc).

3. Vowpal Wabbit
   Vowpal Wabbit (VW) is an open source fast out-of-core learning system library [2]. It was originally developed at Yahoo! Research, and currently at Microsoft Research. It is written in C++ and provides an efficient scalable implementation of online machine learning.
   We tried Vowpal Wabbit on the email body, email subjects, combination of email body and the subject and also the entire email with the headers as a string. The best performance was observed on the latter due to the presence of a large amount of features. It uses an online adaptive learning algorithm that tries to reduce the square loss. It trains on the training set one by one and then uses the model to predict for the test set.

4. Static Rules
   In general static rules helped in improving the classification accuracy. In particular static rules 1, 2 & 3 helped the most in improving the accuracy.

5. Splitting the training samples in the ratio of 4:1 and using Stratified K-Fold Cross-Validation as a model evaluation metric as depicted in fig. 2, we infer Vowpal Wabbit to be the best classifier.

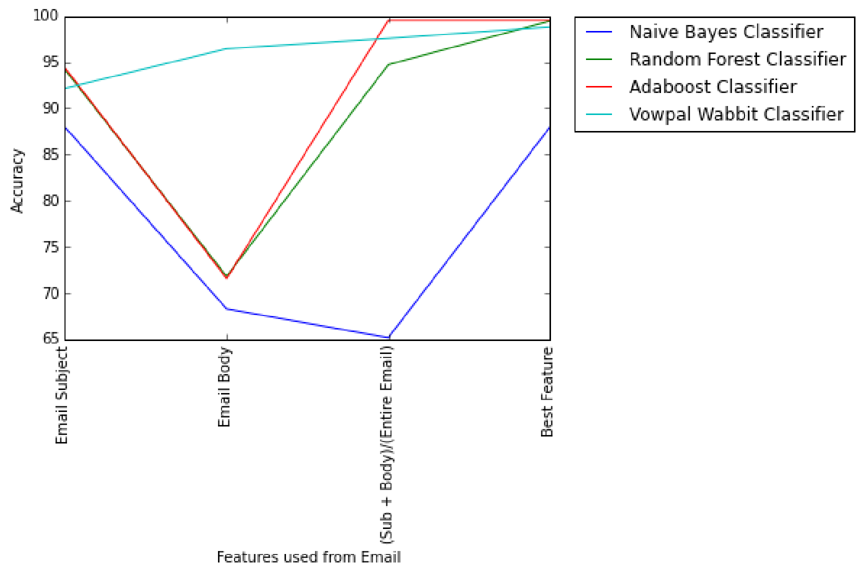


Figure 2: Model Comparison

6 Results

The performance is evaluated by computing the Categorization Accuracy i.e. the percentage of correct predictions. The final evaluation results from Kaggle using different models on the dataset are depicted in the graph fig. 3

7 Conclusion

By trying out different models on our training set, best results were observed for the classification by Vowpal Wabbit(using adaptive learning and squared loss function) along with the application of static rules 1, 2 & 3 on the entire email as a raw string with a Public classification accuracy of 96.166% (96.171% Private Score). The application of static rules

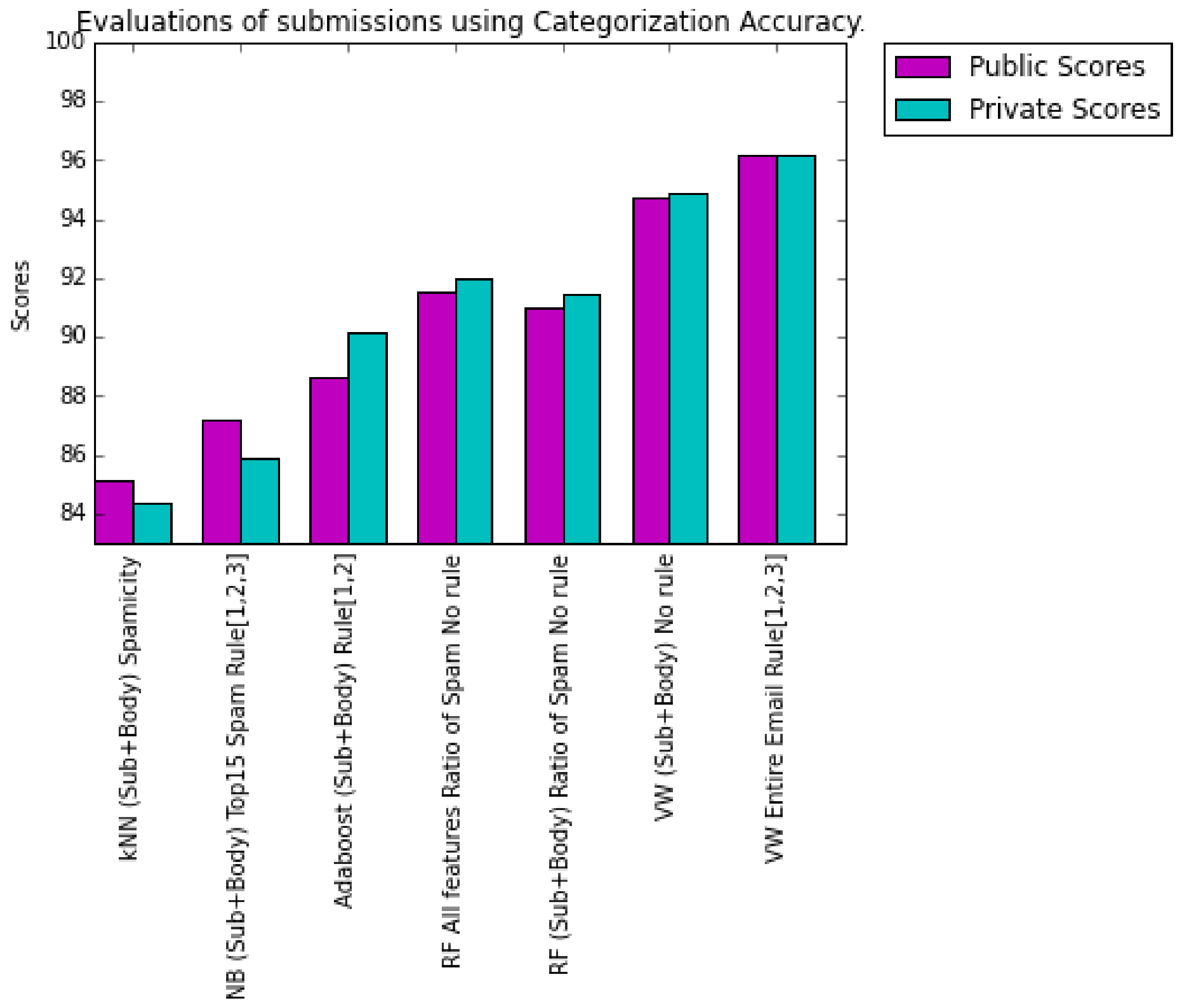


Figure 3: Kaggle Evaluation Results on different models : Public vs Private Scores

4 & 5 were not very useful in the vowpal wabbit classification.

The Naive Bayes Classification accuracy depended on the selection of proper spamminess/hamminess metric. The classification accuracy on the combination of Email body & the subject improved by changing the spamminess metric to top k spam words in the message instead of taking all the words' spamminess into account. When all words in the message are taken into account, the presence of lots of words with low spamminess classifies the mail as ham.

Random Forest with all features and using static rules 1, 2 & 3 gave the best accuracy OF 99.45% on the cross-validation dataset. However it was highly over fitted as verified by its Kaggle score of 91.566% (92.013% Private Score).

The reason why Vowpal Wabbit gave better results compared to the other models was that it operated on the whole raw email data. The other models did not get all the data as features & were provided only a subset of the whole feature space.

References

[1] Awad, W. A., and S. M. ELseuofi. "Machine Learning methods for E-mail Classification." International Journal of Computer Applications (09758887) 16.1 (2011).

[2] Langford, John, L. Li, and A. Strehl. "Vowpal wabbit". `https://github.com/JohnLangford/vowpal_wabbit/wiki` (2011).