MSDS 7333 QTW - Unit 10 Case Study - Spam Classification

Armand Post, Mustafa Sakarwala, Andrew Abbott

March 20, 2018

## Abstract

In this case study, we have examined more than 6,000 email messages to develop a model and test spam filters via classification trees and recursive partitioning. Spam filters are utilized by email providers to scan all emails and are designed to examine content and flag any unwanted messages, known as spam and not flag any wanted messages known as ham. Spam data is classified via machine learning techniques trained with messages known to be either ham or spam by SpamAssassin (<http://spamassassin.apache.org>) specifically for creating and testing these spam filters. The R statistical language is an excellent tool for identifying spam and processing text using approaches such as decision trees. The decision tree approach uses message characteristics from the email content to derive sets of features for email classification.

For the purpose of this casestudy, we are only using the classification tree approach. Recursive partitioning is the method of choice for building decision trees from derived features. To achive this we are using the rpart partitioning algorithm. Several parameters and arguments are explored by using the rpart algorithm through the rpart.control function, which enables control of each parameter input. The rpart.control function does not accept a list of values for each parameter to generate all possible combinations of the argument values. This limitation of rpart.control is addressed with custom code. After tweaking parameters values and calculating prediction accuracies for various possibilities, we find that resulting trees make sense with our understanding of how the parameters are used and we are able to improve overall prediction compared to default results.

## Introduction

The purpose of this study is to put forward a spam mail rule based on management theory by utilizing data mining to manage the information and to explore options on automating classification and filtering of unwanted and potentially harmful email messages, referred to hereafter as spam.

Spam data used in this study comes in from 6,000 messages previously classified by SpamAssassin (<http://spamassassin.apache.org>) specifically for creating and testing spam filters. The information contained in the messages must be organized and processed in order to correctly quantify for further analysis. Most anti-spam researched focused on developing a precise filter to classify emails. We will use the R statistical language to conduct text processing for this initial step and the second step uses either text mining for common word filters or decision trees. The text mining approach looks for words and compares associated frequencies in ham versus spam, while the decision tree approach uses message characteristics to derive variables for email classification. For the purposes of this case study, only the decision tree approach is used.

To read the 6,000 email messages into R, each file is read as its own message. Various parts are then identified within each message as one of the following: (1) the header (sender and subject information); (2) the message body; or (3) an attachment. However, prior to designing data extraction, we investigate email structures to detail and understand the general message format. Then the headers and message bodies can be processed for extraction. Examples of key information gathered include identifying excessive amounts of punctuation and capitalization in the subject line or character count and frequency in the body. The decision tree is then used to derive variables associated with certain characteristics to classify the messages.

We are Answering Q.19 from the *Data Science in R* textbook (Pg. 168): “Consider the other parameters that can be used to control the recursive partitioning process. Read the documentation for them in the rpart.control() documentation. Also, carry out an Internet search for more information on how to tweak the rpart() tuning parameters. Experiment with values for these parameters. Do the trees that result make sense with your understanding of how the parameters are used? Can you improve the prediction using them?”

## Literature Review

Aside from e-mail, spam exists in a number of areas of the internet including social networks, web pages, and phone communication. In their paper on Spam detection on Twitter, McCord and Chuah described using four traditional Machine Learning classifiers for spam detection: Random Forest, Support Vector Machine, Naïve Bayesian, and K-Nearest Neighbor. The author was able to achieve 95.7% precision via Random Forest. At the time the paper was written, Twitter was using a number of ad hoc methods to deal with spam. These included having users self report spam and counting frequency of spam messages with usernames associated. However these approaches allowed for Twitter users to falsely flag other users as spammers and allowed spammers to tag regular users in their messages which resulted in then being incorrectly identified as spammers.

Spam detection continues to evolve to allow real time monitoring of phone calls. In their paper *Spam Detection For Outgoing And Incoming Calls*, Jacobson and Toksoz explain how software is being developed to monitor outgoing calls live on a server equipped with machine learning algorithms. This is to prevent fraud that happens when a consumer buys an international phone calling card and one of the routing carriers in the chain diverts the call to a scam number which usually consists of an audio recording designed to cause the customer to waste credits which boosts revenue for the carriers. If the call is intercepted, the customer can be notified by an audio cue such as a beep.

## Data Scrubbing

The most time consuming, yet important part of this anlysis was the data scrubbing. This data scrubbing included seperating the emails into various sections which include the header, body, and attachment. Among the final steps in the data scrubbing was the generation of a list of keywords that occur commonly in spam emails such as “money”, “prescription”, and “credit”. These keywords were scanned for in the emails to generate the estimate of spam or ham.

The first step was to load the email file locations into the fullDirNames variable. The emails are sorted into five folders according to the classification assigned by the SpamAssassin website.

## [1] "SpamAssassin/easy\_ham" "SpamAssassin/easy\_ham\_2"  
## [3] "SpamAssassin/hard\_ham" "SpamAssassin/spam"   
## [5] "SpamAssassin/spam\_2"

Function were then created to process the email. The processEmail function calls other functions which read the emails, split the emails into header, body, and attachment, and then processes each section into the emailStruct variable.

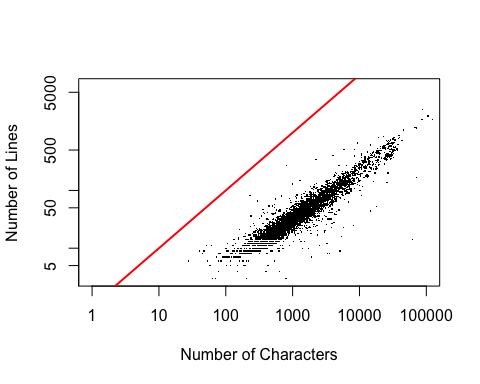
After organizing the emails into a structure more suitable for analysis our next step was to derive the variables which represent the set of features we use to classify the emails as spam or ham. Those features are listed here as seen on page 151 of *Data Science in R*:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Definition |
| isRe | logical | TRUE if Re: appears at the start of the subject. |
| numLines | integer | Number of lines in the body of the message. |
| bodyCharCt | integer | Number of characters in the body of the message. |
| underscore | logical | TRUE if email address in the From field of the header contains an underscore. |
| subExcCt | integer | Number of exclamation marks in the subject. |
| subQuesCt | integer | Number of question marks in the subject. |
| numAtt | integer | Number of attacments in the message. |
| priority | logical | TRUE is a Priority key is present in the header. |
| numRec | numeric | Number of recipients of the message, including CCs. |
| perCaps | numeric | Percentage of capitals among all letters in the message body, excluding attachments. |
| isInReplyTo | logical | TRUE if the IN-Reply-To key is present in the header. |
| sortedRec | logical | TRUE if the recipients’ email addresses are sorts. |
| subPunc | logical | TRUE if words in the subject have punctuation or numbers embedded i them, e.g., w!se. |
| hour | numeric | Hour of the day in the Date field. |
| multipartText | logical | TRUE if the MIME type is multipart/text. |
| hasImages | logical | TRUE if the message contains images. |
| isPGPsigned | logical | TRUE if the message contains a PGP signature. |
| perHTML | numeric | Percentage of characters in HTML tags in the message body in comparison to all characters. |
| subSpamWords | logical | TRUE if the subject contains one of the words in a spam word vector. |
| subBlanks | numeric | Percentage of blanks in the subject. |
| noHost | logical | TRUE if there is no hostname in the Message-Id key in the header. |
| numEnd | logical | TRUE if the email sender’s address (before the @) ends in a number. |
| isYelling | logical | TRUE if the subject is all capital letters. |
| forwards | numeric | Number of forward symbols in a line of the body, e.g., >>> xxx contains 3 forwards. |
| iOrigMsg | logical | TRUE if the message body contains the phrase ‘original message’. |
| isDear | logical | TRUE if the message body contains the word ‘dear’. |
| isWrote | logical | TRUE if the message contains the phrase ‘wrote’. |
| avgWordLen | numeric | The average length of the words in the message. |
| numDlr | numeric | Number of dollar signs in the message body. |

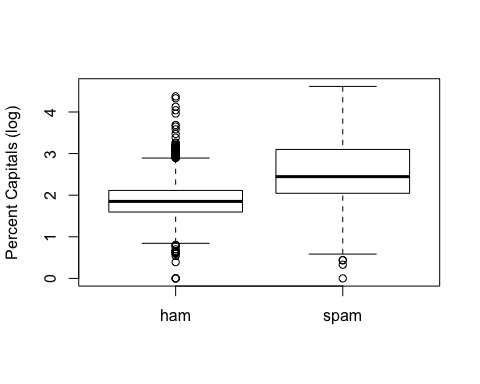
Each of the above variables was created for each email resulting the in final dataframe used for classification. The dataframe emailDF contains 6046 emails, each with 30 variables. Next we investigated some of the features individually through visualizations.

## Visualizations

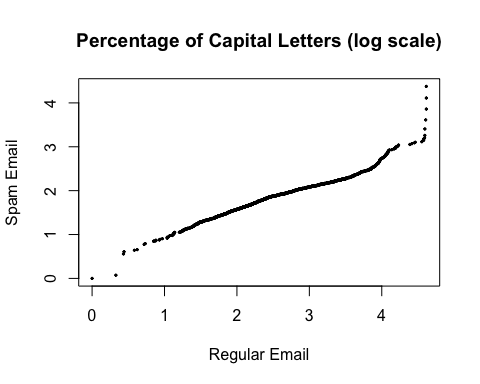
As a high level data integrity check, we plotted the number of lines and number of characters together. Intuitively, the number of lines shouldn’t exceed the number of characters assuming blank lines aren’t used excessively. We see this holds true in the following visualization. No parts of the scatter plot fall on the leftside of the redline, which is where the number of characters and lines are even.



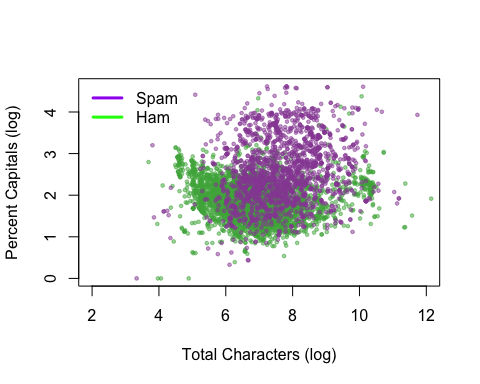
Next we created some boxplots that showed spam emails include a larger proportion of capital letters. This may be a useful way to classify the emails as ham or spam.



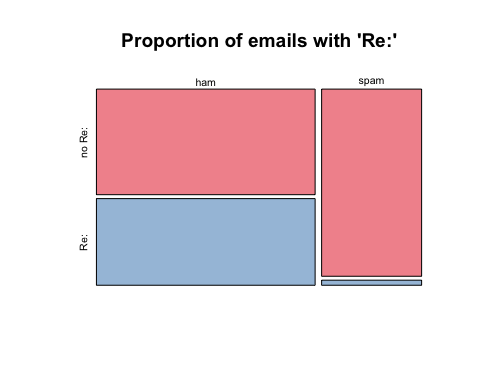
Assessing a quantile-quantile plot of the percent of capital letters categorized by email type yielded some interesting results. We see that the slope isn’t equal to one, which indicates that the distributions of percentage of capital letters is different for spam and ham. Although we have a couple of data points near 0,0 the intercept appears to be near zero for ham, but near the first quartile for spam. This indicates a shift in the mean of the distributions.



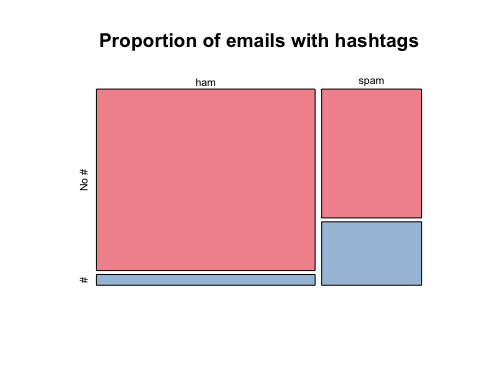
In the next visualization, we plot percent of capital letters along with total characters. We can clearly see that spam emails tend to be longer and have more capital letters.



Another useful visualization was checking to see how often “Re:” was included in the emails, indicating that it had been replied to at some point. We can see that spam emails rarely get replied to from the below mosaic plot. This should prove useful when training the classification models.



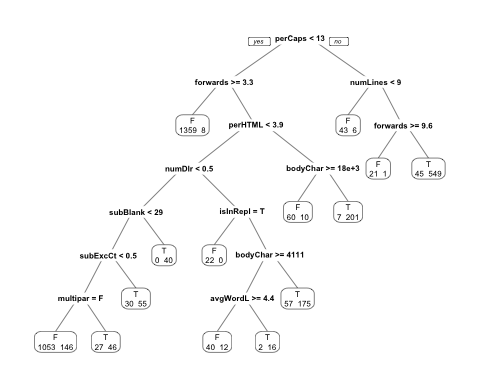
Likewise, we see that ham emails typically do not contain hashtags.



## Fitting the R Part model

Now that we have a good feel for what parameters we can change to better train our spam classifier, we proceeded to fit the model. We performed this by creating test and training datasets for both ham and spam that were split to contain 30% of the records in the test set and 70% in the training set. The iterations of the models were assessed for accuracy including the number of Type I and Type II errors.

After fitting the model, we assessed the tree. We noticed that the a large number of ham emails, 1,359, fall into the leaf of the tree where percentage of caps is less than 13% and the number of forwards is less than 3.3. Only eight misclassfications fall into this area of the tree, indicating that captial letter proportions and number of forwards are good predictors. On the otherhand, we see a large proportion of the spam emails falling where the tree splits on the number of forwards greater than or equal to 9.6.



This tree could then be used to predict the class of the emails in the test set. To evaluate the performance of the classification tree we assessed the rates of Type I and Type II errors. Type I and Type II errors are defined as stated below: - Type I : The proportion of ham messages misclassified as spam. - Type II : The proportion of spam messages misclassified as ham.

The type I error was found by dividing the number of the ham emails in the test set that were misclassified as spam by the number of ham emails in the test set.

## F T   
## 1290 93

6.7245119% of the ham emails in the test set were misclassified as spam. The type II error rate, the percentage of spam emails in the test set that were misclassified as ham, was 17.4050633%.

We observed a significantly higher Type II error rate. One way to improve those results could be to optimize the recursive partitioning function parameters.

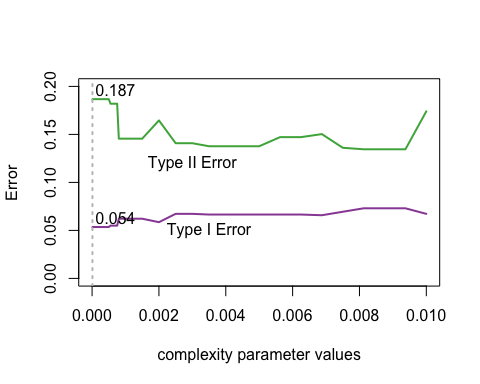
## Parameter testing

The rpart.control() function takes multiple arguments as parameters. Those arguments are listed below and are found in the [rpart.control documentation](https://www.rdocumentation.org/packages/rpart/versions/4.1-12/topics/rpart.control).

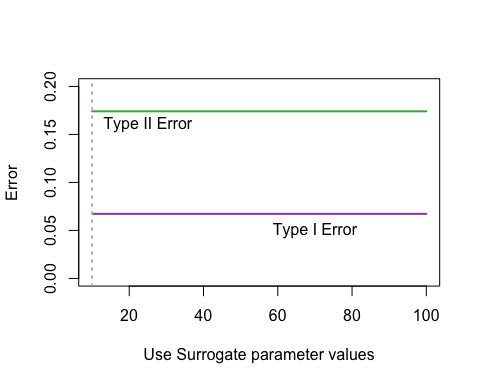
|  |  |
| --- | --- |
| Argument | Description |
| minsplit | the minimum number of observations that must exist in a node in order for a split to be attempted. |
| minbucket | the minimum number of observations in any terminal node. If only one of minbucket or minsplit is specified, the code either sets minsplit to minbucket\*3 or minbucket to minsplit/3, as appropriate. |
| cp | complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. For instance, with anova splitting, this means that the overall R-squared must increase by cp at each step. The main role of this parameter is to save computing time by pruning off splits that are obviously not worthwhile. Essentially,the user informs the program that any split which does not improve the fit by cp will likely be pruned off by cross-validation, and that hence the program need not pursue it. |
| maxcompete | the number of competitor splits retained in the output. It is useful to know not just which split was chosen, but which variable came in second, third, etc. |
| maxsurrogate | the number of surrogate splits retained in the output. If this is set to zero the compute time will be reduced, since approximately half of the computational time (other than setup) is used in the search for surrogate splits. |
| usesurrogate | how to use surrogates in the splitting process. 0 means display only; an observation with a missing value for the primary split rule is not sent further down the tree. 1 means use surrogates, in order, to split subjects missing the primary variable; if all surrogates are missing the observation is not split. For value 2 ,if all surrogates are missing, then send the observation in the majority direction. A value of 0 corresponds to the action of tree, and 2 to the recommendations of Breiman et.al (1984). |
| xval | number of cross-validations. |
| surrogatestyle | controls the selection of a best surrogate. If set to 0 (default) the program uses the total number of correct classification for a potential surrogate variable, if set to 1 it uses the percent correct, calculated over the non-missing values of the surrogate. The first option more severely penalizes covariates with a large number of missing values. |
| maxdepth | Set the maximum depth of any node of the final tree, with the root node counted as depth 0. Values greater than 30 rpart will give nonsense results on 32-bit machines. |

Having a higher Type II error rates is preferred to a higher Type I error rate. This is because it would be mose costly to have a legitmate email flagged and go unread than vice-versa.

First we investigated the complexity (cp) parameter as described above. We plot the Type I and Type II error rates across a range of values for the complexity parameter. The default value for cp is 0.01, so we can see that an improvement can be achieved at a lower complexity. In this use case, having a higher Type II error rate is preferred to a higher Type I error rate. This is because it would be more costly to have a legitimate email flagged and go unread than vice-versa. With this in mind, a complexity approaching zero yields the lowest Type I error rate.



Further exploration was done on the remaining rpart.control parameters. As seen below, demonstrating the error rates of a range of minimum split parameters, the other parameters did not influence our spam classification model.



## Conclusion

We have examined more than 6,000 email messages to develop a model and test spam filters via classification trees and recursive partitioning. We used a decision tree approach which uses message features from the email content to derive svariables for email classification. We have also experimented with values for the parameters for the recursive partitioning function. The decision tree model quickly and with around 6% type I error classify ham and spam email. The complexity parameter is the best candidate parameter for tweaking in our model and can improve the results.

## Future Work

For future work we would like to explore additional machine learning algorithms such as random forests and compare the algorithms in order ti improve results and speed. Additionally, a grid search technique could be used to determing the optimal combination iof parameter values in an automated way.

## References

1.) D. Lang and D. Nolan, *Data Science in R: A Case Studies Approach to Computation Reasoning and Problem Solving.* New York, New York: CRC Press. “Chapter 3: Using Statistics to Identify Spam”

2.) M McCord and M Chuah, *Spam Detection on Twitter Using Traditional Classifiers*, Lehigh University <https://www.tdcommons.org/dpubs_series/759/>

3.) Alex Jacobson and Tuna Toksoz, *Spam Detection For Outgoing And Incoming Calls*, Technical Disclosure Commons - Defensive Publications Series - Oct 16, 2017 <https://www.tdcommons.org/cgi/viewcontent.cgi?article=1821&context=dpubs_series>