Derek Hughes
Assignment #3
Predict 454 – Sec 55
Winter 2016

## Introduction

Technology, the Internet, and ultimately the creation of electronic mail can be a wonderful and powerful medium in which to communicate. Specifically, electronic mail makes it easier to share experiences with others, obtain knowledge, and even to facilitate business relationships. Unfortunately, as with many things, when abused by others electronic mail can become more of a nuisance, especially when it comes to the propagation of spam (unsolicited and generally unwanted bulk emails).

This study aims to discover words, characters, and/or patterns that can be used to identify spam email compared to regular, acceptable email, so as to remove the individual daily burden of sifting through a sea of unsolicited emails. Furthermore, the effective identification of spam emails will reduce the data load that must be carried by Internet Service Providers, which reduces the available bandwidth available for customers trying to use the Internet for more constructive means. Finally, some spam emails contain electronic virus' formatted in HTML code that execute when the email is opened, so the improved identification of spam should reduce the propagation of malicious code over email.

The problem is approached by collecting emails from Hewlett-Packard Lab's postmaster and individuals who identified received emails as spam. Non-spam emails were obtained from a collection of work and personal emails from the Hewlett-Packard data set creators. By using traditional statistical methods to find correlations between these words, characters, and patterns within each email, the data reveal interesting findings that we can subsequently use to build models that predict if an email is classified as spam.

A couple of these interesting findings are that the use of characters such as "!" and "$" in abundance can be strong indictors of spam. Additionally, the simple presence or lack of presence of the words "remove," "free," and "money" can be strong indicators of a spam or not spam email. While knowing these findings does not do much towards helping any individual to identify spam in their inbox, they can be very helpful for developing predictive models that can be used as automated filters for email inboxes.

## Modeling Problem

The problem for this study is to develop a model that best predicts if an email is spam. This model will consider that false positives (the labeling of good email as spam email) are very undesirable. Along the way we will learn interesting relationships between the variables and between the variables and spam email. These inferences are used to create a larger understanding of the data and improve the value of the study, but the main goal is to develop a predictive model that predicts spam email based on the frequency of specific words, the frequency of specific characters, and particular grammatical patterns in the email. Hence, this is a classification modeling problem.

The data available consist of continuous variables representing the frequency that the words and characters are found in emails, in terms of percentage, and the average consecutive

length, longest consecutive length, and total number of capital letters found in a given email. The dependent response variable is a binomial integer value that, when we get to the modeling phase of the project, we convert to a factor. For clarity, we label the spam level of 0 as “NotSpam” and the spam level of 1 as “Spam.”

The techniques we use in the study are implemented in a systematic fashion in order to initially understand the data and then, based on those findings, let the data guide us to developing the most predictive model. Initially, we check the quality of the data and clean it if necessary. This initial step includes investigating the structure of the data, if we have missing values and if they should be imputed, if the values in the data are valid, if the range of values is appropriate, the presence of any outliers, the distributions of the data within each variable, etc.

Second, we apply traditional EDA to look for relationships in the data based on this being a classification problem. We look to compare the means and distributions between variables using box and whisker plots. We confirm any potential visual differences in mean values with T-tests comparing the means of a variable when it is classified as spam and not spam. Furthermore we look for correlations between the continuous variables. We also run a naïve decision tree to guide our EDA investigation and support any of our previous findings.

For creating models, we use the training paradigm to determine goodness of fit of our models. We do this by dividing the data into a training and test set based on a 70% to 30% division, respectively. Each model is fitted to the training data set and built based on the training set. The performance of the model is based on the accuracy between the predicted email type and the actual email type of the training set. Finally, the model is fitted to the test data set and the accuracy is determined and used as the ultimate indicator of model performance for comparing between models. The false positive rate (the error of identifying good/wanted emails as spam) is also considered.

When we are ready to build our models, for easier interpretation of the results, we manually transform the response variable into a factor and attach the labels “NotSpam” and “Spam” as appropriate. We then build four different models: logistic regression, decision tree, support vector machine, and random forest. We build multiple versions of each model by mainly using a simple model with the variable “char_$,” a full model, a model with categorized versions of five of the most significant models (remove, money, free, char_!, and char_$), and backward and stepwise selected variables. Not every model uses the same variables to create different versions of their model, and support vector machines only runs two different variable models but uses four different kernels to find the optimum support vector machine model. Finally, our data quality check identifies some variables that are not found in any spam emails, so we try one decision tree model using only those variables found in spam emails.

Finally, after each new subset of variables is applied to each model, its accuracy on the in-sample training set and the out-sample test set is evaluated. Each model’s accuracy values on the test set are used as the final performance metric for comparing between models. However, since misclassifying good email as spam is strongly discouraged, a model’s false positive rate will be considered as well when making a final determination on which model is best.

## Data Quality Check

The data have continuous variables and a binary response variable and are made up 4600 observations and 58 variables.

| VARIABLE | CLASS | DESCRIPTION |
|---|---|---|
| make | numeric | % email with word: **make** |
| address | numeric | % email with word: **address** |
| all | numeric | % email with word: **all** |
| w_3d | numeric | % email with word: **w_3d** |
| our | numeric | % email with word: **our** |
| over | numeric | % email with word: **over** |
| remove | numeric | % email with word: **remove** |

Table 1 – Data Description (full table in appendix)

The numeric data is comprised of fifty-eight variables. All are continuous except the response variable which is a binomial integer which we change to a two-class factor for modeling. There are not any missing values and all values appear valid.

In terms of range, the maximum range seems suspect for the variables 3d and char_!. For at least one observation of each variable, the percent of the email with the word 3d and "!" is 42% and 32% respectively. That would essentially mean almost every other word is "3d" or every three characters is "!" which seems unlikely, but a viable possibility considering the nature of spam formats. The minimum range value was zero for every variable, notwithstanding the three capital letter measurement values (cap_avg, cap_long, cap_total).

Based on values outside the 99$^{th}$ percentile, most variables have 30 or more outliers. Outliers play a much larger role in this analysis given the sparsity of the data, as most words or characters were not found or were sparsely found in a given email. This results in the mean percent of the words equaling the word or character under evaluation to be less than 1% of the total words in the email in almost every variable. Only "you" has a mean above 1% at 1.6% Thus, in most cases, merely having 1.5% or 2% of the total words in an email being those under evaluation might approach the definition of being an outlier. Hence, as opposed to being a nuisance, outliers or lack of outliers can be used as strong indicators of the email classification in this situation.

| VARIABLE | IS.NA | NOT.0 | %.ALL | %.SPAM | SPAM.RT | MIN | Q95 | Q99 | MAX | MEAN | SD | OUT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| make | 0 | 1053 | 0.23 | 0.14 | 0.61 | 0 | 0.651 | 1.260 | 4.540 | 0.105 | 0.305 | 45 |
| address | 0 | 897 | 0.20 | 0.14 | 0.70 | 0 | 0.671 | 3.271 | 14.280 | 0.213 | 1.291 | 46 |
| all | 0 | 1887 | 0.41 | 0.24 | 0.59 | 0 | 1.250 | 2.271 | 5.100 | 0.281 | 0.504 | 46 |
| 3d | 0 | 47 | 0.01 | 0.01 | 1.00 | 0 | 0.000 | 0.040 | 42.810 | 0.065 | 1.395 | 46 |
| our | 0 | 1747 | 0.38 | 0.25 | 0.66 | 0 | 1.490 | 2.941 | 10.000 | 0.312 | 0.673 | 46 |
| over | 0 | 999 | 0.22 | 0.15 | 0.68 | 0 | 0.630 | 1.210 | 5.880 | 0.096 | 0.274 | 46 |
| remove | 0 | 807 | 0.18 | 0.17 | 0.94 | 0 | 0.740 | 1.960 | 7.270 | 0.114 | 0.391 | 43 |
| internet | 0 | 824 | 0.18 | 0.13 | 0.72 | 0 | 0.600 | 1.620 | 11.110 | 0.105 | 0.401 | 46 |

Table 2 – Numeric Data Descriptive Statistics Summary Chart (full table in appendix)

Exploring further, we created a column for the percent of emails that contained a variable, the percent of spam emails that contained that variable, and a ratio of the latter divided by the former to give us a "spam ratio," which based on those percentages gives us an approximate idea if the email will be spam if that word is found in any email at all (Figure 1). Hence, we can see that some variables, such as "remove," have a .94 spam ratio and, therefore, if found in any email, that email is likely to be a spam email. This gives us even deeper insights into each variable.

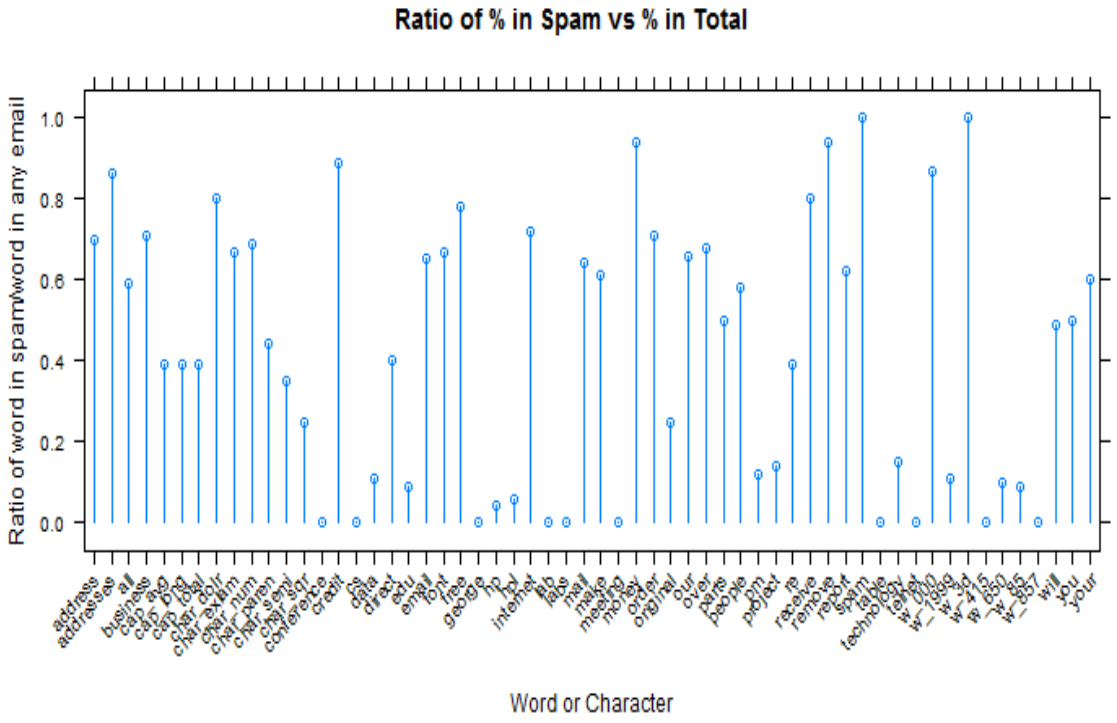


Figure 1 – Ratio of % in Spam vs. % in Total Emails (All Variables)

The distribution of most dependent variables is highly right-skewed, as the overwhelming majority of values are zero (Figure 2). One way to handle this is to categorize a variable. Through our EDA, we will create three dummy variables for five of the more significant variables that identify spam, and those three dummy variables will categorize each variable based on its values being equal to zero, zero to one, and greater than one. In the modeling section, these dummy variables will be used in our four models in an attempt to improve accuracy.

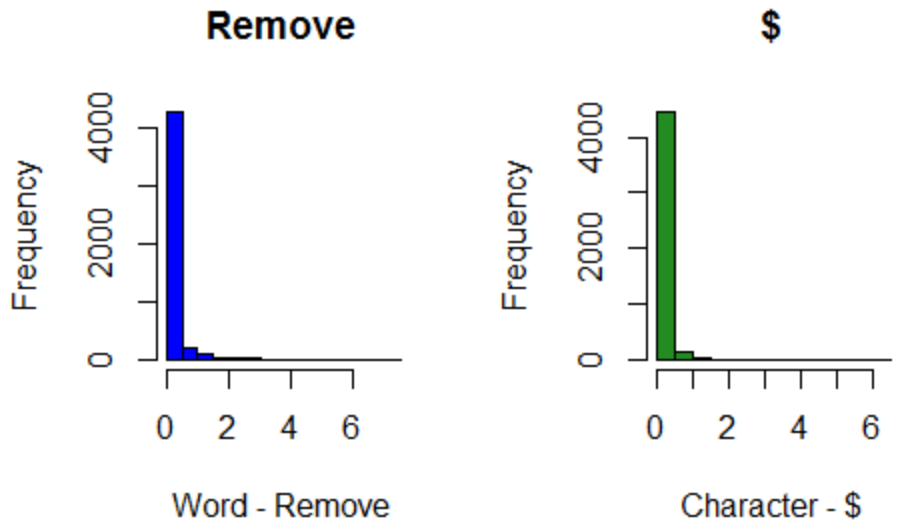


Figure 2 – Distribution of Dependent Variables (Remove, Char_$)

## Exploratory Data Analysis

Exploratory data analysis is the process of looking for relationships in the data driven by the type of problem. In our case, we look for relationships related to the response variable, spam.

Using correlation to compare a binary factor response variable to continuous variables is not appropriate so we use box and whisker plots to compare the means of a given variable to whether the variable is classified as spam or not spam. We visually inspect the means and outliers to look for significant differences.

Here (Figure 3) we can see how, although difficult to visually see the difference because the mean is so close to zero, the mean values are different for spam and not spam emails when we find the word “remove” or character “!” in the email. Furthermore, we can see how much higher and wider the range of outliers are for the spam emails compared to the not spam emails for these two variables.

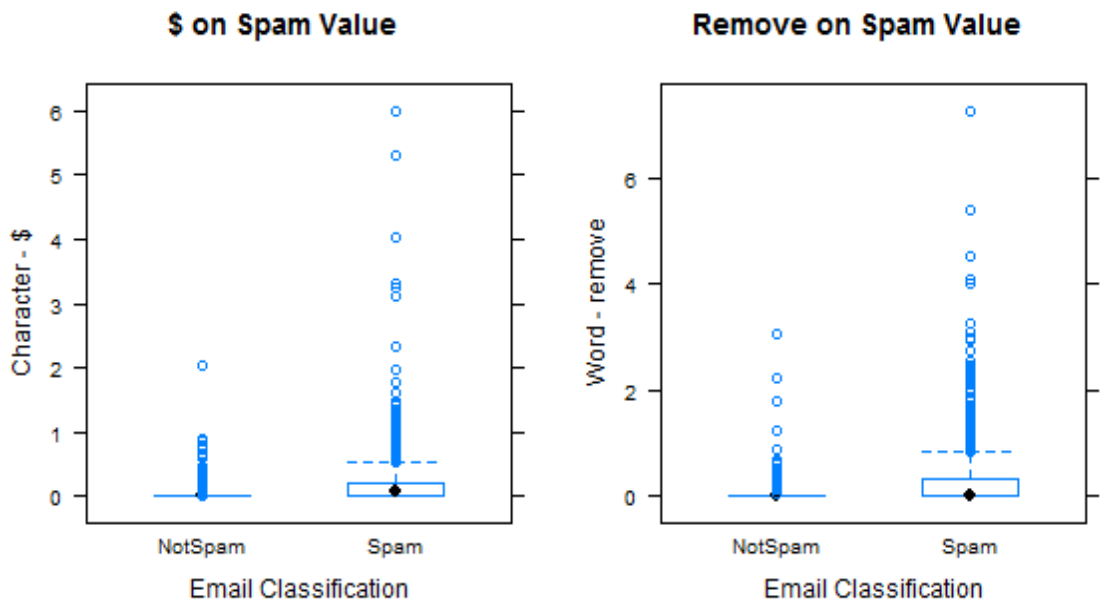


Figure 3 – Box Whisker Plot of Variables vs Spam Classification (Remove and char_!)

Since the means looks significantly different between the spam and not spam values for each variable, we run a t-test to check for significance and verify our visual inspection (Table 3 and Table 4).

| Welch Two Sample t-test |
| --- |
| data: mean.char_dolr$char_dolr and mean.char_dolr0$char_dolr |
| t = 19.0062, df = 1899.175, p-value < 2.2e-16 |

Table 3 – T-Test of Char_! on Spam Classification

| Welch Two Sample t-test |
| --- |
| data: mean.remove$remove and mean.remove0$remove |
| t = 19.5649, df = 1899.066, p-value < 2.2e-16 |

Table 4 – T-Test of Remove on Spam Classification

The t-tests tell us both variables have significant differences in their means. We repeate this process and find numerous variables with significant differences between their means for spam and not spam emails, such as: money, your, char_!, free, business, cap_long, and cap_avg.

Finally we run a naïve decision tree on spam using all variables to see what variables are selected as significant. Not surprisingly considering the relationships we discovered with our box whisker plots, it uses "$" for the main split and nine other variables to finish with thirteen terminal nodes.

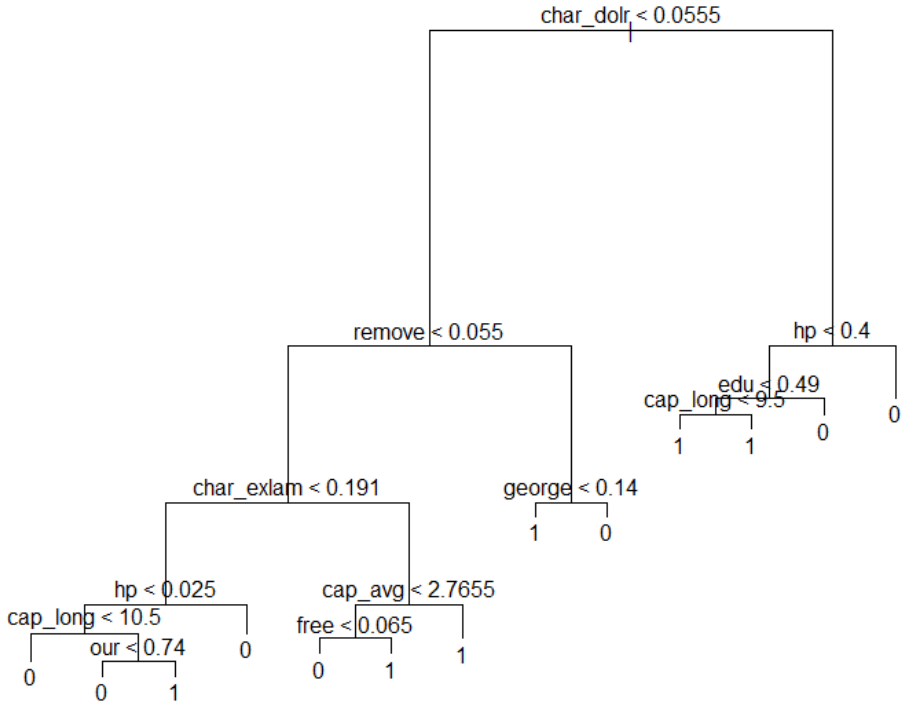


Figure 4 – Tree Diagram Using Spam and All Variables

## Modeling

### Logistic Regression Models

| Model | # Variables | Formula | Train Acrcy | Train FP | Train FP% | Test Acrcy | Test FP | Test FP% |
|---|---|---|---|---|---|---|---|---|
| logistic regression | simple-1 | spam~char_dolr | 0.783918 | 73 | 0.02 | 0.7810007 | 35 | 0.03 |
| logistic regression | full-58 | spam~. | 0.9329401 | 81 | 0.03 | 0.9253082 | 48 | 0.03 |
| logistic regression | bkwd select - 43 | spam~bkwd | 0.9292145 | 88 | 0.03 | 0.9224075 | 48 | 0.03 |
| logistic regression | stepwise - 43 | spam~stepwise | 0.9292145 | 88 | 0.03 | 0.9224075 | 48 | 0.03 |
| logistic regression | 5 binned - 68 | spam~5 binned | 0.9348029 | 97 | 0.03 | 0.9325598 | 48 | 0.03 |

Table 5 – Models Built by Logistic Regression

We run five types of logistic regression models to determine the effect of various variables in the model. The first model regresses "char_$" on spam. This is a simple model and we achieve an average accuracy of 78.1% and false positive (FP) of misclassifying good email as spam of 3%. Next, we run a full model regression. This improves our accuracy substantially to 92.5%. Then we run backward and stepwise variable selection with logistic regression and find the same variables selected for both methods. We achieve an accuracy of 92.2% with a FP of 3%. Finally, from our EDA we saw the distribution of many independent variables could be broken into categories of dummy variables representing 0 values, 0<=1 values, and >1 values. By cross-referencing the variables selected by our EDA decision tree on the full data set, our spam ratio values from our summary chart, and box whisker/t-test analysis, we selected and subsequently categorized the five variables we felt were the most significant to classifying the emails. Replacing each of those variables with

their three categorized dummy variable counterparts, we ran a final logistic regression on the full data set and achieved our highest accuracy of 93.3% and FP of 3%. Cleary the transformation into categorized dummy variables improved the model.

## Decision Tree Models

| Model | # Variables | Formula | Train Acrcy | Train FP | Train FP% | Test Acrcy | Test FP | Test FP% |
|---|---|---|---|---|---|---|---|---|
| decision tree | simple-1 | spam~char_dolr | 0.7938528 | 93 | 0.03 | 0.7940537 | 40 | 0.03 |
| decision tree | full-58 | spam~. | 0.9053089 | 63 | 0.02 | 0.8926759 | 39 | 0.03 |
| decision tree | 5 binned - 68 | spam~5 binned | 0.916796 | 90 | 0.03 | 0.9035533 | 48 | 0.03 |
| decision tree | bkwd select - 43 | spam~ bkwd | 0.9053089 | 63 | 0.02 | 0.8926759 | 36 | 0.03 |
| decision tree | only vars in spam - 48 | spam~vars in spam | 0.9022043 | 73 | 0.02 | 0.8905004 | 39 | 0.03 |

Table 6 – Models Built by Decision Trees

A decision tree is initially run with the same simple model as logistic regression. Accuracy is a meager 79.4% and FP of 3%. Next we run a full model and improve our accuracy to 89.3% with FP of 3%. We then run a model only using the backward selected variables used in the logistic regression. Accuracy was identical to the full model which shows the tree is still selecting the same variables for the splits, but with improved parsimony from 58 variables to 43 at the same FP %. Again we try a model based on our EDA by only using variables that were found in spam emails. We lose a very slight bit of accuracy (89.1% to 89.3%). Finally we use our categorized dummy variables we created in place of their original variables and, again, achieve the highest accuracy at 90.4% with a FP of 3%.

## Support Vector Machine Models

| Model | # Variables | Formula | Train Acrcy | Train FP | Train FP% | Test Acrcy | Test FP | Test FP% |
|---|---|---|---|---|---|---|---|---|
| support vector | linear - cost 10 | spam~. | 0.9323192 | 83 | 0.03 | 0.9296592 | 98 | 0.04 |
| support vector | linear - cost 10 | spam~ 5 binned | 0.9372866 | 43 | 0.03 | 0.9340102 | 50 | 0.03 |
| support vector | poly-deg3-coef2 | spam~. | 0.9580876 | 44 | 0.01 | 0.9441624 | 36 | 0.03 |
| support vector | poly-deg3-coef4 | spam~ 5 binned | 0.9726793 | 35 | 0.01 | 0.9354605 | 44 | 0.03 |
| support vector | radial-gam.1 | spam~. | 0.9720584 | 22 | 0.01 | 0.9144307 | 29 | 0.02 |
| support vector | sig-gam.1-coef.1 | spam~. | 0.8190003 | 299 | 0.09 | 0.8114576 | 135 | 0.1 |

Table 7 – Models Built by Support Vector Machines

Support Vector Machine models use the full model or the full model with the categorical dummy variables from the five significant variables we found from our EDA in every model. The main difference is the kernel, degree, coef0, and gamma values applied.

For the linear kernel we use a soft margin penalty cost of 10. Using the categorized dummy variables we achieved a test accuracy of 93.4% with FP of 3%. The best polynomial kernel uses the full model only with 3 degrees and a coef0 of 4. It achieves a test accuracy of

94.4% and FP of 3%, our best of any model thus far. We adjust the class weights to place more penalty on FP When we move to radial and sigmoid kernels with a gamma of .1our accuracy decreases substantially. This indicates that the data does not match the more complex kernels, although the radial kernel did achieve our lowest FP of 2% and perhaps the use a higher gamma value would improve our variance and, thus, the test error rate because our radial training set accuracy was extremely high (97.2%).

## Random Forest Models

| Model | # Variables | Formula | Train Acrcy | Train FP | Train FP% | Test Acrcy | Test FP | Test FP% |
|---|---|---|---|---|---|---|---|---|
| random forest | full-58 | spam~. | 0.9506364 | 62 | 0.02 | 0.9477883 | 28 | 0.02 |
| random forest | 5 binned - 68 | spam~5 binned | 0.950326 | 62 | 0.02 | 0.9535896 | 22 | 0.02 |
| random forest | bkwd select - 43 | spam~bkwd | 0.9509469 | 63 | 0.02 | 0.9463379 | 28 | 0.02 |

Table 8 – Models Built by Random Forests

Random forest is run using three models: one full model, one full model with the categorized dummy variables substituted for their original variables, and one using the backward selected variables. Using the pruning technique, for each model we run an initial model using 500 trees then another using the optimized "pruned" number of trees. Every model performs very well, achieving the highest test accuracy classifications and, in addition, the lowest FP rates. Similar to decision trees, our full model and backward selected variables model are almost identical on accuracy (94.7% to 94.6%) and FP of 2% each. Finally, once again, our model with the categorized dummy variables produces our best test accuracy of any model in the project with a 95.3% accuracy and, at the same time, achieves the lowest FP rate at 2%!

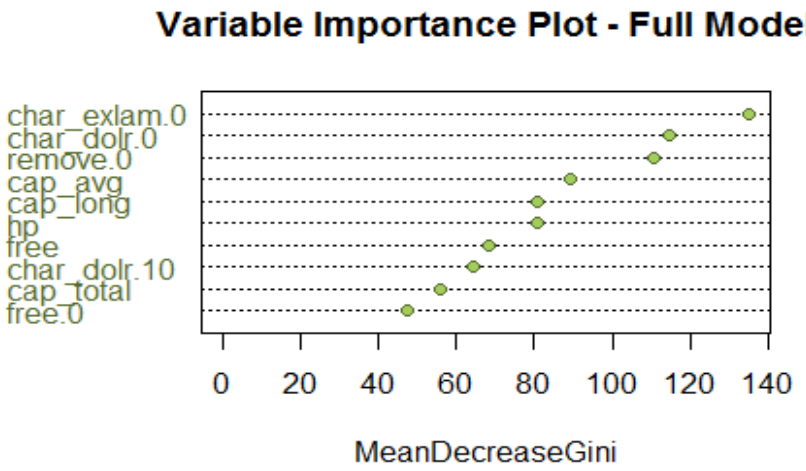


Figure 5 – Variable Importance for Best Random Forest Model

| Model | # Variables | Formula | Train Acrcy | Train FP | Train FP% | Test Acrcy | Test FP | Test FP% |
|---|---|---|---|---|---|---|---|---|
| logistic regression | simple-1 | spam~char_dolr | 0.783918 | 73 | 0.02 | 0.7810007 | 35 | 0.03 |
| logistic regression | full-58 | spam~. | 0.9329401 | 81 | 0.03 | 0.9253082 | 48 | 0.03 |
| logistic regression | bkwd select - 43 | spam~bkwd | 0.9292145 | 88 | 0.03 | 0.9224075 | 48 | 0.03 |
| logistic regression | stepwise - 43 | spam~stepwise | 0.9292145 | 88 | 0.03 | 0.9224075 | 48 | 0.03 |
| logistic regression | 5 binned - 68 | spam~5 binned | 0.9348029 | 97 | 0.03 | 0.9325598 | 48 | 0.03 |
| decision tree | simple-1 | spam~char_dolr | 0.7938528 | 93 | 0.03 | 0.7940537 | 40 | 0.03 |
| decision tree | full-58 | spam~. | 0.9053089 | 63 | 0.02 | 0.8926759 | 39 | 0.03 |
| decision tree | 5 binned - 68 | spam~5 binned | 0.916796 | 90 | 0.03 | 0.9035533 | 48 | 0.03 |
| decision tree | bkwd select - 43 | spam~ bkwd | 0.9053089 | 63 | 0.02 | 0.8926759 | 36 | 0.03 |
| decision tree | only vars in spam - 48 | spam~vars in spam | 0.9022043 | 73 | 0.02 | 0.8905004 | 39 | 0.03 |
| support vector | linear - cost 10 | spam~. | 0.9323192 | 83 | 0.03 | 0.9296592 | 98 | 0.03 |
| support vector | linear - cost 10 | spam~ 5 binned | 0.9372866 | 43 | 0.03 | 0.9340102 | 50 | 0.04 |
| support vector | poly-deg3-coef2 | spam~. | 0.9580876 | 44 | 0.01 | 0.9441624 | 36 | 0.03 |
| support vector | poly-deg3-coef4 | spam~ 5 binned | 0.9726793 | 35 | 0.01 | 0.9354605 | 44 | 0.03 |
| support vector | radial-gam.1 | spam~. | 0.9720584 | 22 | 0.01 | 0.9144307 | 29 | 0.02 |
| support vector | sig-gam.1-coef.1 | spam~. | 0.8190003 | 299 | 0.09 | 0.8114576 | 135 | 0.1 |
| random forest | full-58 | spam~. | 0.9506364 | 62 | 0.02 | 0.9477883 | 28 | 0.02 |
| random forest | 5 binned - 68 | spam~5 binned | 0.950326 | 62 | 0.02 | 0.9535896 | 22 | 0.02 |
| random forest | bkwd select - 43 | spam~bkwd | 0.9509469 | 63 | 0.02 | 0.9463379 | 28 | 0.02 |

Table 9 - Comparison of Model Performance

## Conclusions

It is clear that based on our exploratory data analysis and modeling methods that categorizing the variables into dummy variables achieved extra information. Specifically, our best model uses the absence the characters "!" and "$" and the word "remove" as its most important variables (Figure 5). Furthermore, the use of capital letters successively is also an important indicator of spam. Comparing the models, accuracy is most important but false positive rates are very important as well. In that sense, Random Forest performs the best on the test data set. However, Support Vector Machines are extremely strong on the training data, only to show some high variance when applied to the test data. It would be interesting to see if the Support Vector Machines can be improved upon in the future.

Some recommendations to improve these models are the more extensive use of pruning on the decision trees and the use of boosting on the random forest models. The FP rate for SVM models might be improved by experimenting with the class weight values some more. The gamma values for the radial kernel might be adjusted to help transfer the excellent training accuracy to the test data set. Also, considering how well random forest models used the dummy variables, categorizing more variables may make an even more significant impact as well. Finally, it should be noted that by using our EDA methods we, again, learn insights that we can apply and improve our models. Additionally it should be noted how well "traditional" approaches, such as logistic regression, can perform against the more complex models such as random forest when EDA insights are applied.

## Appendix

Hastie, T., James, G., Tibshirani, R., Witten, D. (2013). *An Introduction to Statistical Learning with Applications in R*. New York, NY: Springer.

Johnson, W. & Myatt, G. (2014). *Making Sense of Data I. A Practical Guide to Exploratory Data Analysis and Data Mining*. Hoboken, New Jersey: Wiley.

Lesmeister, C. (2015). *Mastering Machine Learning with R*. Birmingham, UK: Packt Publishing.

Mount, J. & Zumel, N. (2014). *Practical Data Science with R*. Shelter Island, NY: Springer.

TABLE 1

| | VAR.NAME | VAR.CLASS | DESCRIPTION |
|---|---|---|---|
| 1 | make | numeric | % email with word: make |
| 2 | address | numeric | % email with word: address |
| 3 | all | numeric | % email with word: all |
| 4 | w_3d | numeric | % email with word: w_3d |
| 5 | our | numeric | % email with word: our |
| 6 | over | numeric | % email with word: over |
| 7 | remove | numeric | % email with word: remove |
| 8 | internet | numeric | % email with word: internet |
| 9 | order | numeric | % email with word: order |
| 10 | mail | numeric | % email with word: mail |
| 11 | receive | numeric | % email with word: receive |
| 12 | will | numeric | % email with word: will |
| 13 | people | numeric | % email with word: people |
| 14 | report | numeric | % email with word: report |
| 15 | addresses | numeric | % email with word: addresses |
| 16 | free | numeric | % email with word: free |
| 17 | business | numeric | % email with word: business |
| 18 | email | numeric | % email with word: email |
| 19 | you | numeric | % email with word: you |
| 20 | credit | numeric | % email with word: credit |
| 21 | your | numeric | % email with word: your |
| 22 | font | numeric | % email with word: font |
| 23 | w_000 | numeric | % email with word: w_000 |
| 24 | money | numeric | % email with word: money |
| 25 | hp | numeric | % email with word: hp |

| | | | |
|---|---|---|---|
| 26 | hpl | numeric | % email with word: hpl |
| 27 | george | numeric | % email with word: george |
| 28 | w_650 | numeric | % email with word: w_650 |
| 29 | lab | numeric | % email with word: lab |
| 30 | labs | numeric | % email with word: labs |
| 31 | telnet | numeric | % email with word: telnet |
| 32 | w_857 | numeric | % email with word: w_857 |
| 33 | data | numeric | % email with word: data |
| 34 | w_415 | numeric | % email with word: w_415 |
| 35 | w_85 | numeric | % email with word: w_85 |
| 36 | technology | numeric | % email with word: technology |
| 37 | w_1999 | numeric | % email with word: w_1999 |
| 38 | parts | numeric | % email with word: parts |
| 39 | pm | numeric | % email with word: pm |
| 40 | direct | numeric | % email with word: direct |
| 41 | cs | numeric | % email with word: cs |
| 42 | meeting | numeric | % email with word: meeting |
| 43 | original | numeric | % email with word: original |
| 44 | project | numeric | % email with word: project |
| 45 | re | numeric | % email with word: re |
| 46 | edu | numeric | % email with word: edu |
| 47 | table | numeric | % email with word: table |
| 48 | conference | numeric | % email with word: conference |
| 49 | char_semi | numeric | % email with character: char_semi |
| 50 | char_paren | numeric | % email with character: char_paren |
| 51 | char_sqr | numeric | % email with character: char_sqr |
| 52 | char_exlam | numeric | % email with character: char_exlam |
| 53 | char_dolr | numeric | % email with character: char_dolr |
| 54 | char_num | numeric | % email with character: char_num |
| 55 | cap_avg | numeric | Capital letter average run length |
| 56 | cap_long | integer | Capital letter longest run length |
| 57 | cap_total | integer | Total number of captial letters |
| 58 | spam | factor | Spam |

TABLE 2

| | AR.NAME | NOT.0 | PCT.USE | PCT.SPAM | RAT.SPAM | Q95 | Q99 | NA | NA | MAX | MEAN | OUT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | make | 1053 | 0.23 | 0.14 | 0.61 | 0.000 | 0.000 | 0.651 | 1.260 | 4.540 | 0.105 | 45 |
| 2 | address | 897 | 0.20 | 0.14 | 0.70 | 0.000 | 0.000 | 0.671 | 3.271 | 14.280 | 0.213 | 46 |
| 3 | all | 1887 | 0.41 | 0.24 | 0.59 | 0.000 | 0.420 | 1.250 | 2.271 | 5.100 | 0.281 | 46 |
| 4 | w_3d | 47 | 0.01 | 0.01 | 1.00 | 0.000 | 0.000 | 0.000 | 0.040 | 42.810 | 0.065 | 46 |
| 5 | our | 1747 | 0.38 | 0.25 | 0.66 | 0.000 | 0.383 | 1.490 | 2.941 | 10.000 | 0.312 | 46 |
| 6 | over | 999 | 0.22 | 0.15 | 0.68 | 0.000 | 0.000 | 0.630 | 1.210 | 5.880 | 0.096 | 46 |
| 7 | remove | 807 | 0.18 | 0.17 | 0.94 | 0.000 | 0.000 | 0.740 | 1.960 | 7.270 | 0.114 | 43 |
| 8 | internet | 824 | 0.18 | 0.13 | 0.72 | 0.000 | 0.000 | 0.600 | 1.620 | 11.110 | 0.105 | 46 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | order | 773 | 0.17 | 0.12 | 0.71 | 0.000 | 0.000 | 0.651 | 1.330 | 5.260 | 0.090 | 46 |
| 10 | mail | 1302 | 0.28 | 0.18 | 0.64 | 0.000 | 0.160 | 1.330 | 2.650 | 18.180 | 0.239 | 46 |
| 11 | receive | 709 | 0.15 | 0.12 | 0.80 | 0.000 | 0.000 | 0.380 | 1.020 | 2.610 | 0.060 | 46 |
| 12 | will | 2324 | 0.51 | 0.25 | 0.49 | 0.100 | 0.800 | 2.180 | 4.101 | 9.670 | 0.542 | 46 |
| 13 | people | 852 | 0.19 | 0.11 | 0.58 | 0.000 | 0.000 | 0.610 | 1.440 | 5.550 | 0.094 | 45 |
| 14 | report | 357 | 0.08 | 0.05 | 0.62 | 0.000 | 0.000 | 0.200 | 1.490 | 10.000 | 0.059 | 45 |
| 1 | addresses | 336 | 0.07 | 0.06 | 0.86 | 0.000 | 0.000 | 0.171 | 1.610 | 4.410 | 0.049 | 37 |
| 16 | free | 1240 | 0.27 | 0.21 | 0.78 | 0.000 | 0.100 | 1.340 | 3.260 | 20.000 | 0.249 | 46 |
| 17 | business | 963 | 0.21 | 0.15 | 0.71 | 0.000 | 0.000 | 0.820 | 2.320 | 7.140 | 0.143 | 46 |
| 18 | email | 1037 | 0.23 | 0.15 | 0.65 | 0.000 | 0.000 | 1.201 | 2.230 | 9.090 | 0.185 | 45 |
| 19 | you | 3226 | 0.70 | 0.35 | 0.50 | 1.310 | 2.640 | 4.760 | 7.310 | 18.750 | 1.662 | 46 |
| 20 | credit | 424 | 0.09 | 0.08 | 0.89 | 0.000 | 0.000 | 0.320 | 2.540 | 18.180 | 0.086 | 46 |
| 21 | your | 2422 | 0.53 | 0.32 | 0.60 | 0.220 | 1.270 | 3.170 | 5.260 | 11.110 | 0.810 | 40 |
| 22 | font | 117 | 0.03 | 0.02 | 0.67 | 0.000 | 0.000 | 0.000 | 6.750 | 17.100 | 0.121 | 45 |
| 23 | w_000 | 679 | 0.15 | 0.13 | 0.87 | 0.000 | 0.000 | 0.731 | 1.620 | 5.450 | 0.102 | 45 |
| 24 | money | 735 | 0.16 | 0.15 | 0.94 | 0.000 | 0.000 | 0.570 | 1.310 | 12.500 | 0.094 | 45 |
| 25 | hp | 1090 | 0.24 | 0.01 | 0.04 | 0.000 | 0.000 | 3.060 | 8.690 | 20.830 | 0.550 | 42 |
| 26 | hpl | 811 | 0.18 | 0.01 | 0.06 | 0.000 | 0.000 | 1.690 | 4.160 | 16.660 | 0.265 | 44 |
| 27 | george | 780 | 0.17 | 0.00 | 0.00 | 0.000 | 0.000 | 2.730 | 20.000 | 33.330 | 0.767 | 20 |
| 28 | w_650 | 463 | 0.10 | 0.01 | 0.10 | 0.000 | 0.000 | 0.801 | 2.811 | 9.090 | 0.125 | 46 |
| 29 | lab | 372 | 0.08 | 0.00 | 0.00 | 0.000 | 0.000 | 0.500 | 2.322 | 14.280 | 0.099 | 46 |
| 30 | labs | 469 | 0.10 | 0.00 | 0.00 | 0.000 | 0.000 | 0.660 | 2.320 | 5.880 | 0.103 | 45 |
| 31 | telnet | 293 | 0.06 | 0.00 | 0.00 | 0.000 | 0.000 | 0.280 | 1.560 | 12.500 | 0.065 | 46 |
| 32 | w_857 | 205 | 0.04 | 0.00 | 0.00 | 0.000 | 0.000 | 0.000 | 1.200 | 4.760 | 0.047 | 45 |
| 33 | data | 405 | 0.09 | 0.01 | 0.11 | 0.000 | 0.000 | 0.451 | 2.350 | 18.180 | 0.097 | 45 |
| 34 | w_415 | 215 | 0.05 | 0.00 | 0.00 | 0.000 | 0.000 | 0.000 | 1.201 | 4.760 | 0.048 | 46 |
| 35 | w_85 | 485 | 0.11 | 0.01 | 0.09 | 0.000 | 0.000 | 0.620 | 2.270 | 20.000 | 0.105 | 46 |
| 36 | tech | 599 | 0.13 | 0.02 | 0.15 | 0.000 | 0.000 | 0.650 | 1.720 | 7.690 | 0.097 | 45 |
| 37 | w_1999 | 829 | 0.18 | 0.02 | 0.11 | 0.000 | 0.000 | 0.930 | 1.920 | 6.890 | 0.137 | 44 |
| 38 | parts | 83 | 0.02 | 0.01 | 0.50 | 0.000 | 0.000 | 0.000 | 0.190 | 8.330 | 0.013 | 45 |
| 39 | pm | 384 | 0.08 | 0.01 | 0.12 | 0.000 | 0.000 | 0.491 | 1.720 | 11.110 | 0.079 | 46 |
| 40 | direct | 453 | 0.10 | 0.04 | 0.40 | 0.000 | 0.000 | 0.361 | 1.580 | 4.760 | 0.065 | 46 |
| 41 | cs | 148 | 0.03 | 0.00 | 0.00 | 0.000 | 0.000 | 0.000 | 1.440 | 7.140 | 0.044 | 45 |
| 42 | meeting | 341 | 0.07 | 0.00 | 0.00 | 0.000 | 0.000 | 0.540 | 3.840 | 14.280 | 0.132 | 41 |
| 43 | original | 375 | 0.08 | 0.02 | 0.25 | 0.000 | 0.000 | 0.270 | 1.111 | 3.570 | 0.046 | 46 |
| 44 | project | 327 | 0.07 | 0.01 | 0.14 | 0.000 | 0.000 | 0.240 | 2.041 | 20.000 | 0.079 | 46 |
| 45 | re | 1311 | 0.28 | 0.11 | 0.39 | 0.000 | 0.110 | 1.411 | 4.160 | 21.420 | 0.301 | 45 |
| 46 | edu | 517 | 0.11 | 0.01 | 0.09 | 0.000 | 0.000 | 1.020 | 4.162 | 22.050 | 0.180 | 46 |
| 47 | table | 63 | 0.01 | 0.00 | 0.00 | 0.000 | 0.000 | 0.000 | 0.040 | 2.170 | 0.005 | 46 |
| 48 | confnce | 203 | 0.04 | 0.00 | 0.00 | 0.000 | 0.000 | 0.000 | 0.900 | 10.000 | 0.032 | 46 |
| 49 | char_semi | 790 | 0.17 | 0.06 | 0.35 | 0.000 | 0.000 | 0.172 | 0.645 | 4.385 | 0.039 | 46 |
| 50 | char_paren | 2715 | 0.59 | 0.26 | 0.44 | 0.065 | 0.188 | 0.530 | 1.030 | 9.752 | 0.139 | 46 |
| 51 | char_sqr | 529 | 0.12 | 0.03 | 0.25 | 0.000 | 0.000 | 0.093 | 0.284 | 4.081 | 0.017 | 46 |
| 52 | ar_exlam | 2257 | 0.49 | 0.33 | 0.67 | 0.000 | 0.314 | 1.116 | 2.631 | 32.478 | 0.269 | 46 |
| 53 | char_dolr | 1400 | 0.30 | 0.24 | 0.80 | 0.000 | 0.052 | 0.377 | 0.894 | 6.003 | 0.076 | 46 |
| 54 | char_num | 750 | 0.16 | 0.11 | 0.69 | 0.000 | 0.000 | 0.147 | 0.763 | 19.829 | 0.044 | 46 |

```
55 cap_avg  4600    1.00     0.39     0.39  2.276   3.705    8.857   44.758  1102.500   5.192  46
56cap_long  4600    1.00     0.39     0.39 15.000  43.000  181.000  669.000  9989.000  52.171  41
57cap_total 4600    1.00     0.39     0.39 95.000 265.250 1223.300 3027.020 15841.000 283.290  46
58    spam  1812    0.39     0.39     1.00  0.000   1.000    1.000    1.000     1.000   0.394   0
```

## R Code

```
#### FIGURE 1 ####
# plot showing which words, if found in any email, will more likely indicate spam
xyplot(my.summary.df$RATIO.SPAM ~ my.summary.df$VAR.NAME, data=my.summary.df,
      type = c("p","h"),scales=list(x = list(rot = 45)),
      xlab="Word or Character", ylab="Ratio of word in spam/word in any email",
      main="Ratio of % in Spam vs % in Total")

#### FIGURE 2 #######
#histogram of numeric distributions
par(mfrow=c(1,2))
hist(mydata$remove, xlab="Word - Remove", main="Remove", col="blue")
hist(mydata$char_dolr, xlab="Character - $", main="$", col="forestgreen")

######### FIGURE 3 #############
par(mfrow=c(2,2))
a<-bwplot(money ~ spam, data=mydata,
      xlab="Email Classification", ylab="Word-Money",
      main="Money on Spam Value")
b<-bwplot(remove ~ spam, data=mydata,
      xlab="Email Classification", ylab="Word - remove",
      main="Remove on Spam Value")
c<-bwplot(free ~ spam, data=mydata,
      xlab="Email Classification", ylab="Word - free",
      main="Free on Spam Value")
d<-bwplot(char_dolr ~ spam, data=mydata,
      xlab="Email Classification", ylab="Character - $",
      main="$ on Spam Value")

# four panel lattice layout using print.trellis
print(a, position = c(0, 0.5, 0.5, 1), more = TRUE)
print(b, position = c(0.5, 0.5, 1, 1), more = TRUE)
print(c, position = c(0, 0, 0.5, 0.5), more = TRUE)
print(d, position = c(0.5, 0, 1, 0.5))

# two panel lattice layout using print.trellis
print(d, position = c(0, 0, 0.5, 1), more = TRUE)
print(b, position = c(0.5, 0, 1, 1))

#######FIGURE 4########
# tree on entire data set no-modified training data set
spam.fit <- tree(factor(spam) ~ ., data=mydata) #left-hand side must be factor for classification tree
summary(spam.fit) #summarize findings

plot(spam.fit) # plot model
text(spam.fit, pretty=0)
```

```
#####FIGURE 5#######

### RF full model ###
set.seed(71)
rf.full = randomForest(spam~., data=spam.train)
print(rf.full)

#determine how many trees for min error rate
plot(rf.full)
which.min(rf.full$err.rate[,1])

#retry model with min number of trees
set.seed(71)
rf.full.2 = randomForest(spam~., data=spam.train, ntree=98)
print(rf.full.2)
varImpPlot(rf.full.2, n.var=10, main="Variable Importance Plot - Full Model",
          color = "darkolivegreen", gcolor = par("fg"), lcolor = "black")


#######FIGURE 6########
tree.fit <- tree(price~., data=train)
summary(tree.fit) # summarize findings

plot(tree.fit) # plot model
text(tree.fit, pretty=0)
# tree price~. selects carat, ideal:No, BlueNile:No
# tree logprice~. selects carat only
# tree logprice~ logcarat selects logcarat only



######TABLE 1########
###### data description table ######
VAR.NAME = colnames(mydata)

variable.class <- function(df){
  var.class = rep(1,ncol(df),0)
    for(i in 1:ncol(df)){
        var.class[i] = class(df[,i])
    }
return(var.class)
}

VAR.CLASS <- variable.class(mydata)

var_descrip_48 <- paste("% email with word: ", colnames[1:48])
var_descrip_54 <- paste("% email with character: ", colnames[49:54])
var_descrip_57 <- c("Capital letter average run length", "Capital letter longest run length", "Total number of
captial letters","Spam")

descrip_attach <- append(var_descrip_48,var_descrip_54)
DESCRIPTION <- append(descrip_attach, var_descrip_57)

data.descrip= data.frame(VAR.NAME, VAR.CLASS, DESCRIPTION)

head(data.descrip)
###### end - data description table ######
```

```
#########TABLE 2############
########## numeric summary ##########
my.summary <- function(in.orig.df){

 # check and create df with only numeric columns for the summary function
 col.to.add = numeric()  # initialize column vector

 for (i in 1:ncol(in.orig.df)){
  if(is.numeric(in.orig.df[,i]) == 'TRUE'){
    col.to.add = append(col.to.add,i)  # add column number to vector
  }
 }

 in.df <-in.orig.df[col.to.add]  # using '<<' adds object to global env for use outside function
 VAR.NAME <- colnames(in.df);   # get variable names from df

 # count not-NAs function
 not.na <- function(x){
  out <- sum(-1*(is.na(x)-1));
  return(out);
 }
 NOT.NA <- apply(X=in.df,MARGIN=2,FUN='not.na');

 # count NAs function
 count.na <- function(x){
  out <- sum(is.na(x));
  return(out);
 }

 # count # emails in which this word appears
 not.0 <- function(x){
  out <- sum(x!=0);
  return(out);
 }

 # count # emails in which this word appears
 not.0.spam <- function(x){
  out <- sum((x!=0) & (mydata['spam']==1));
  return(out);
 }

 ###### OUTLIER FUNCTION #####
 find.outliers <- function(x){

  # count outliers in each variable based on outside Q01 or Q99 (these limits are adjustable)

  # create quantile of input
  quantile.99=quantile(x,c(0.99), na.rm=T)
  #quantile.01=quantile(x,c(0.01), na.rm=T)

  # create min and max of input
  #min= min(x)
  max= max(x)

  # initial outlier value for this column loop
  outlier = 0

  for(j in 1:length(x)){  # loop through each value in each column

    # 1) find observation value
```

```
      obs=x[j]

      # 2) find observation and outlier barrier ratios
      #obs.min.ratio = obs/min
      obs.max.ratio = max/obs
      #outlier.min.ratio = quantile.01/min
      outlier.max.ratio = max/quantile.99

      # 3) compare if observation ratio is less than outlier ratios
      #result.min = obs.min.ratio < outlier.min.ratio #if observation/min ratio less than outlier/min ratio = outlier
      result.max = obs.max.ratio < outlier.max.ratio #if max/obs ratio less than max/outlier ratio = outlier

      # result.min
#       if(result.min == 'TRUE' || result.max == 'TRUE'){
#         outlier = outlier + 1
#       }
        # using this outlier check for spam database because 0 is a common min value and
        # produces NaN and Inf values in the minimum ratios needed in this function
        # hence, only looking for extreme max values here
        if(result.max=='TRUE'){
          outlier=outlier+1
        }
    }
    return(outlier)
  }
  ###### END OUTLIER SECTION #####

  #### combine functions and apply() to the numeric data frame #####
  IS.NA <- apply(X=in.df,MARGIN=2,FUN='count.na');  # number of NAs
  #PCT.NA <- round(IS.NA/(NOT.NA+IS.NA),digits=3);   # percent that are NAs
  NOT.0 <- apply(X=in.df,MARGIN=2,FUN='not.0')   # number of emails in which this word appeared
  NOT.0.SPAM <- apply(X=in.df,MARGIN=2,FUN='not.0.spam')   # number of emails in which this word
appeared
  PCT.USE <- round((NOT.0/NOT.NA),digits=2);   # number of emails in which this word appeared
  PCT.SPAM <- round((NOT.0.SPAM/NOT.NA),digits=2);   # number of spam emails in which this word
appeared
  RATIO.SPAM <-round(PCT.SPAM/PCT.USE, digits=2);   # ratio of spam emails with word compared to
ratio of any email with word
  # Note that we need to account for computations with NA values;

  p <- c(0.50,0.75,0.95,0.99); # set quantiles we want to use
  Q <- round(apply(X=in.df,MARGIN=2,FUN='quantile',p,na.rm=TRUE),digits=3); # quantiles
  tQ.df <- as.data.frame(t(Q));  # transpose quantile vector and turn into data frame to display properly later
  colnames(tQ.df) <- c('Q50','Q75','Q95','Q99');  # change col names in data frame to be more readable

  MIN <- round(apply(X=in.df,MARGIN=2,FUN='min',na.rm=TRUE),digits=3); # min value
  MAX <- round(apply(X=in.df,MARGIN=2,FUN='max',na.rm=TRUE),digits=3); # max value
  MEAN <- round(apply(X=in.df,MARGIN=2,FUN='mean',na.rm=TRUE),digits=3); # mean value
  SD <- round(apply(X=in.df,MARGIN=2,FUN='sd',na.rm=TRUE),digits=3); # Standard deviation value
  OUT = apply(X=in.df,MARGIN=2,FUN='find.outliers')  # outlier values for each variable stored in vector

  # combine all results into one data frame for display
  wide.df <-
cbind(VAR.NAME,IS.NA,NOT.0,PCT.USE,PCT.SPAM,RATIO.SPAM,MIN,tQ.df,MAX,MEAN,SD,OUT);
  rownames(wide.df) <- seq(1,dim(wide.df)[1],1);   # rename rownames to numeric values

  # Write CSV in R
  write.csv(wide.df, file = "summary_numeric_data_spam.csv")

  return(wide.df);
```

```
}

my.summary.df = my.summary(in.orig.df=mydata)  # test my.summary() function with spam data set

# unique additions to this mysummary():
# the % emails with word in it
# the % spam emails with word in it
########## end - numeric summary ##########


#######TABLE 3 and 4 #########

mean.remove<- mydata %>%
  filter(spam==1) %>%
  select(remove)

mean.remove0<- mydata %>%
  filter(spam==0) %>%
  select(remove)

t.test(mean.remove$remove, mean.remove0$remove, p.adjust="bonferroni")

mean.char_dolr<- mydata %>%
  filter(spam==1) %>%
  select(char_dolr)

mean.char_dolr0<- mydata %>%
  filter(spam==0) %>%
  select(char_dolr)

t.test(mean.char_dolr$char_dolr, mean.char_dolr0$char_dolr, p.adjust="bonferroni")


########################




##### EDA ##########


##### categorizing highly influential variables #####
### into variables 0, 0-1, 1+ for each varaible ###

#mydata$char_exlam
for(i in 1:length(mydata$char_exlam)){
   if(mydata$char_exlam[i]==0){
      mydata$char_exlam.0[i]=1}
        else{
          mydata$char_exlam.0[i]=0
   }
   if(mydata$char_exlam[i]>0 & mydata$char_exlam[i]<=1){
      mydata$char_exlam.10[i]=1}
        else{
          mydata$char_exlam.10[i]=0
   }
   if(mydata$char_exlam[i]>1){
      mydata$char_exlam.1plus[i]=1}
        else{
```

```
        mydata$char_exlam.1plus[i]=0
    }
}
mydata$char_exlam=NULL
sum(mydata$char_exlam.1plus==0)

# mydata$money
for(i in 1:length(mydata$char_dolr)){
  if(mydata$char_dolr[i]==0){
    mydata$char_dolr.0[i]=1}
  else{
    mydata$char_dolr.0[i]=0
  }
  if(mydata$char_dolr[i]>0 & mydata$char_dolr[i]<=1){
    mydata$char_dolr.10[i]=1}
  else{
    mydata$char_dolr.10[i]=0
  }
  if(mydata$char_dolr[i]>1){
    mydata$char_dolr.1plus[i]=1}
  else{
    mydata$char_dolr.1plus[i]=0
  }
}
mydata$char_dolr=NULL
sum(mydata$char_dolr.1plus==0)


# mydata$remove
for(i in 1:length(mydata$remove)){
  if(mydata$remove[i]==0){
    mydata$remove.0[i]=1}
  else{
    mydata$remove.0[i]=0
  }
  if(mydata$remove[i]>0 & mydata$remove[i]<=1){
    mydata$remove.10[i]=1}
  else{
    mydata$remove.10[i]=0
  }
  if(mydata$remove[i]>1){
    mydata$remove.1plus[i]=1}
  else{
    mydata$remove.1plus[i]=0
  }
}
mydata$remove=NULL
sum(mydata$remove.1plus==0)

# mydata$your
for(i in 1:length(mydata$your)){
  if(mydata$your[i]==0){
    mydata$your.0[i]=1}
  else{
    mydata$your.0[i]=0
  }
  if(mydata$your[i]>0 & mydata$your[i]<=1){
    mydata$your.10[i]=1}
  else{
    mydata$your.10[i]=0
```

```
  }
  if(mydata$your[i]>1){
    mydata$your.1plus[i]=1}
  else{
    mydata$your.1plus[i]=0
  }
}
mydata$your=NULL
sum(mydata$your.1plus==0)

# mydata$free
for(i in 1:length(mydata$free)){
  if(mydata$free[i]==0){
    mydata$free.0[i]=1}
  else{
    mydata$free.0[i]=0
  }
  if(mydata$free[i]>0 & mydata$free[i]<=1){
    mydata$free.10[i]=1}
  else{
    mydata$free.10[i]=0
  }
  if(mydata$free[i]>1){
    mydata$free.1plus[i]=1}
  else{
    mydata$free.1plus[i]=0
  }
}
mydata$free=NULL
sum(mydata$free.1plus==0)
###### end - segment or binning or trimming ##


########################################



############# MODELING AND EVALUATIONS ###################################

###### Logisitic Regression with variable selection ####

##### logistic simple model ####
log.simp <- glm(spam~char_dolr,family=binomial,data=spam.train)
# Uses 1 variable: AIC 3383

##### full model ####
log.full <- glm(spam~.,family=binomial,data=spam.train)
vif(log.full)
# Uses 58 variables: AIC 1337
# Uses 68 variables: AIC 1195

##### backward #####
log.full.bkwd <- step(log.full,direction="backward",trace=FALSE)
# Uses 42 variables: AIC 1318
# ***  our, remove, free, business, your, w_000, hp, george, re, edu, char_exlam, char_dolr, cap_long
# **       cap_total, char_num, internet, w_85, technology, meeting
# *        credit, money, data, project, conference, char_semi

##### stepwise #####
log.full.step <- step(log.full,direction="both",trace=FALSE)
```

```
# Uses 42 variables: AIC 1318

##### top 13 most sign variables from bkwd selection ####
log.sig.vars <- glm(spam~our+remove+free+business+your+
                    w_000+hp+george+re+edu+char_exlam+
                    char_dolr+cap_long,family=binomial,data=spam.train)
# Uses 13 variables: AIC 1563

##### vars with spam.ratio > .6 ####
log.spam.ratio.6 <- glm(factor(spam)~make+address+w_3d+our+over+remove+internet+order+
                      mail+receive+report+addresses+free+business+email+
                      credit+your+font+w_000+money+char_exlam+char_dolr+
                      char_num,family=binomial,data=spam.train)
# Uses 23 variables: AIC 2088


###### LogReg Predictions - train/test data set ######

### log.simp model ###
prob.log.simp <- predict(log.simp, type="response")
prob.test.log.simp <- predict(log.simp, type="response", newdata=spam.test)

contrasts(spam.train$spam) # view how response factor is treated in R

pred.log.simp=rep("NotSpam",3221) # training data set size
pred.log.simp[prob.log.simp > .5]="Spam"  # set threshold at 50%
pred.test.log.simp=rep("NotSpam",1379) # training data set size
pred.test.log.simp[prob.test.log.simp > .5]="Spam"  # set threshold at 50%

conf.train= table(pred.log.simp, spam.train$spam)
conf.train.perc = round(conf.train/dim(spam.train)[1], digits=2)
conf.test= table(pred.test.log.simp, spam.test$spam)
conf.test.perc = round(conf.test/dim(spam.test)[1], digits=2)

accuracy.train=sum(diag(table(pred.log.simp,spam.train$spam)))/length(pred.log.simp)
accuracy.test=sum(diag(table(pred.test.log.simp,spam.test$spam)))/length(pred.test.log.simp)
#accuracy train = 0.783918  FP = 73, .02
#accuracy test = 0.7810007  FP = 35, .03



### log.full model ###
prob.log.full <- predict(log.full, type="response")
prob.test.log.full <- predict(log.full, type="response", newdata=spam.test)

contrasts(spam.train$spam) # view how response factor is treated in R

pred.log.full=rep("NotSpam",3221) # training data set size
pred.log.full[prob.log.full > .5]="Spam"  # set threshold at 50%
pred.test.log.full=rep("NotSpam",1379) # training data set size
pred.test.log.full[prob.test.log.full > .5]="Spam"  # set threshold at 50%

conf.train= table(pred.log.full, spam.train$spam)
conf.train.perc = round(conf.train/dim(spam.train)[1], digits=2)
conf.test= table(pred.test.log.full, spam.test$spam)
conf.test.perc = round(conf.test/dim(spam.test)[1], digits=2)

accuracy.train=sum(diag(table(pred.log.full,spam.train$spam)))/length(pred.log.full)
accuracy.test=sum(diag(table(pred.test.log.full,spam.test$spam)))/length(pred.test.log.full)
#accuracy train = 0.9329401  FP = 81, .03
```

```
#accuracy test = 0.9253082  FP = 48, .03
#accuracy train bin = 0.9348029  FP = 97, .03
#accuracy test bin = 0.9325598  FP = 48, .03


### log.bkwd model ###
prob.log.full.bkwd <- predict(log.full.bkwd, type="response")
prob.test.log.full.bkwd <- predict(log.full.bkwd, type="response", newdata=spam.test)

contrasts(spam.train$spam) # view how response factor is treated in R

pred.log.full.bkwd=rep("NotSpam",3221) # training data set size
pred.log.full.bkwd[prob.log.full.bkwd > .5]="Spam"  # set threshold at 50%
pred.test.log.full.bkwd=rep("NotSpam",1379) # training data set size
pred.test.log.full.bkwd[prob.test.log.full.bkwd > .5]="Spam"  # set threshold at 50%

conf.train= table(pred.log.full.bkwd, spam.train$spam)
conf.train.perc = round(conf.train/dim(spam.train)[1], digits=2)
conf.test= table(pred.test.log.full.bkwd, spam.test$spam)
conf.test.perc = round(conf.test/dim(spam.test)[1], digits=2)

accuracy.train=sum(diag(table(pred.log.full.bkwd,spam.train$spam)))/length(pred.log.full.bkwd)
accuracy.test=sum(diag(table(pred.test.log.full.bkwd,spam.test$spam)))/length(pred.test.log.full.bkwd)
#accuracy train = 0.9292145  FP = 88, .03
#accuracy test = 0.9224075  FP = 48, .03




############ end - LogReg predictions train and test ###########

################ end - Logistic Regression ####################



###################### Decision Tree ######################

#### tree - simple model ####
tree.simp <- tree(spam ~ char_dolr, data=spam.train) #left-hand side must be factor for classification tree
summary(tree.simp) #summarize findings

plot(tree.simp) # plot model
text(tree.simp, pretty=0)
# tree selects variables: char_exlam, remove, money, george, hp,
# cap_long, our, cap_avg, char_dolr, business


#### tree - backward var model ####
tree.bkwd <- tree(spam ~ make + address + all + w_3d + our + over + remove + internet +
  order + report + addresses + free + business + credit + your +
  w_000 + money + hp + hpl + george + w_650 + lab + telnet +
  data + w_85 + technology + parts + pm + cs + meeting + original +
  project + re + edu + table + conference + char_semi + char_exlam +
  char_dolr + char_num + cap_avg + cap_long + cap_total, data=spam.train)
summary(tree.bkwd) #summarize findings

plot(tree.bkwd) # plot model
text(tree.bkwd, pretty=0)
```

```
#### tree - var in spam model ####
tree.in.spam <- tree(spam ~ make+address+all+w_3d+our+over+remove+internet+order+
            mail+receive+will+people+report+addresses+free+business+
            email+you+credit+you+font+w_000+money+hp+hpl+w_650+telnet+
            data+w_85+technology+w_1999+parts+pm+direct+original+project+
            re+edu+char_semi+char_paren+char_sqr+char_exlam+char_dolr+
            char_num+cap_avg+cap_long+cap_total, data=spam.train)
summary(tree.in.spam) #summarize findings

plot(tree.in.spam) # plot model
text(tree.in.spam, pretty=0)



###### Decision Tree Predictions - train/test data set ######

### tree simple model ###
pred.tree.simp.train <- predict(tree.simp, newdata=spam.train, type="class")
pred.tree.simp.test <- predict(tree.simp, newdata=spam.test, type="class")

# confusion matrix for training and test data
conf.train <- table(pred.tree.simp.train, spam.train$spam)
conf.test <- table(pred.tree.simp.test, spam.test$spam)
conf.train.perc = round(conf.train/dim(spam.train)[1], digits=2)
conf.test.perc = round(conf.test/dim(spam.test)[1], digits=2)

# Accuracy for train and test sets
accuracy.train=sum(diag(table(pred.tree.simp.train,spam.train$spam)))/length(pred.tree.simp.train)
accuracy.test=sum(diag(table(pred.tree.simp.test,spam.test$spam)))/length(pred.tree.simp.test)
# accuracy train = 0.7938528  FP = 93, .03
# accuracy test = 0.7940537  FP = 40, .03


### tree bkwd model ###
pred.tree.bkwd.train <- predict(tree.bkwd, newdata=spam.train, type="class")
pred.tree.bkwd.test <- predict(tree.bkwd, newdata=spam.test, type="class")

# confusion matrix for training and test data
conf.train = table(pred.tree.bkwd.train, spam.train$spam)
conf.test =table(pred.tree.bkwd.test, spam.test$spam)
conf.train.perc = round(conf.train/dim(spam.train)[1], digits=2)
conf.test.perc = round(conf.test/dim(spam.test)[1], digits=2)

# Accuracy for train and test sets
accuracy.train=sum(diag(table(pred.tree.bkwd.train,spam.train$spam)))/length(pred.tree.bkwd.train)
accuracy.test=sum(diag(table(pred.tree.bkwd.test,spam.test$spam)))/length(pred.tree.bkwd.test)
# accuracy train = 0.9053089  FP = 63, .02
# accuracy test = 0.8926759  FP = 36, .03


### tree vars in spam model ###
pred.tree.in.spam.train <- predict(tree.in.spam, newdata=spam.train, type="class")
pred.tree.in.spam.test <- predict(tree.in.spam, newdata=spam.test, type="class")

# confusion matrix for training and test data
conf.train = table(pred.tree.in.spam.train, spam.train$spam)
conf.test = table(pred.tree.in.spam.test, spam.test$spam)
conf.train.perc = round(conf.train/dim(spam.train)[1], digits=2)
conf.test.perc = round(conf.test/dim(spam.test)[1], digits=2)
```

```
# Accuracy for train and test sets
accuracy.train=sum(diag(table(pred.tree.in.spam.train,spam.train$spam)))/length(pred.tree.in.spam.train)
accuracy.test=sum(diag(table(pred.tree.in.spam.test,spam.test$spam)))/length(pred.tree.in.spam.test)
# accuracy train = 0.9022043  FP = 73, .02
# accuracy test = 0.8905004  FP = 39, .03

### end - decision tree predictions ######

################ end - Decision Tree ####################



################ Support Vector Machine ####################


### SVM full linear model ###
set.seed(71)
linear.tune.full = tune.svm(spam~., data=spam.train, kernel="linear", cost=c(0.001,0.01, 0.1, 1,5,10),
class.weights=c('Spam'=1,'NotSpam'=100))
summary(linear.tune)
best.linear.full = linear.tune.full$best.model

### SVM full polynomial model ###
set.seed(71)
poly.tune.full = tune.svm(spam~., data=spam.train, kernel="polynomial",degree=c(3,4,5),
coef0=c(0.1,0.5,1,2,3,4), class.weights=c('Spam'=1,'NotSpam'=10))
summary(poly.tune.full)
best.poly.full = poly.tune.full$best.model

### SVM full radial model ###
set.seed(71)
rbf.tune = tune.svm(spam~., data=spam.train, kernel="radial",gamma=c(0.1,0.5,1,2,3,4))
summary(rbf.tune)
best.rbf.full = rbf.tune$best.model

# try model p.249 PDSwR with class.weights=c('Spam'=1,'NotSpam'=10)
# appended to model - notspam=10 is no change, trying notspam=100

### SVM full sigmoid model ###
set.seed(71)
sigmoid.tune = tune.svm(spam~., data=spam.train, kernel="sigmoid",gamma=c(0.1,0.5,1,2,3,4),
coef0=c(0.1,0.5,1,2,3,4))
summary(sigmoid.tune)
best.sigmoid.full = sigmoid.tune$best.model



###### SVM Predictions - train/test data set ######

### SVM full linear model ###
tune.train.full = predict(best.linear.full, type="response")
tune.test.full = predict(best.linear.full, newdata=spam.test, type="response")

conf.train = table(tune.train.full, spam.train$spam)
conf.test = table(tune.test.full, spam.test$spam)
conf.train.perc = round(conf.train/dim(spam.train)[1], digits=2)
conf.test.perc = round(conf.test/dim(spam.test)[1], digits=2)

# Accuracy for train and test sets
```

```
accuracy.train=sum(diag(table(tune.train.full,spam.train$spam)))/length(tune.train.full)
accuracy.test=sum(diag(table(tune.test.full,spam.test$spam)))/length(tune.test.full)
# accuracy train = 0.9323192  FP = 83, .03
# accuracy test = 0.9296592  FP = 43, .03
# accuracy train bin = 0.9372866  FP = 98, .03
# accuracy test bin = 0.9340102 FP = 50, .04
# - best parameters:
#   cost - the soft margin that trades error for stability
#   large C gives low bias-high variance, bc penalizes
#   misclassificaiton a lot
#   10 - making this lower may improve FP (misclassifying non-spam as spam)

### SVM full poly model ###
tune.train.full = predict(best.poly.full, type="response")
tune.test.full = predict(best.poly.full, newdata=spam.test, type="response")

conf.train = table(tune.train.full, spam.train$spam)
conf.test = table(tune.test.full, spam.test$spam)
conf.train.perc = round(conf.train/dim(spam.train)[1], digits=2)
conf.test.perc = round(conf.test/dim(spam.test)[1], digits=2)

# Accuracy for train and test sets
accuracy.train=sum(diag(table(tune.train.full,spam.train$spam)))/length(tune.train.full)
accuracy.test=sum(diag(table(tune.test.full,spam.test$spam)))/length(tune.test.full)
# accuracy train = 0.9580876  FP = 44, .01
# accuracy test = 0.9441624  FP = 36, .03
# accuracy train bin = 0.9726793  FP = 35, .01
# accuracy test bin = 0.9354605  FP = 44, .03
# - best parameters:
#   degree coef0
#     3     2
# Degree is the degree of the polynomial
# Coef0 is a limit on the coeficent so doesn't go to infinity
# or continue to smaller numbers approaching zero?


### SVM full radial model ###
tune.train.full = predict(best.rbf.full)
tune.test.full = predict(best.rbf.full, newdata=spam.test)

conf.train = table(tune.train.full, spam.train$spam)
conf.test = table(tune.test.full, spam.test$spam)
conf.train.perc = round(conf.train/dim(spam.train)[1], digits=2)
conf.test.perc = round(conf.test/dim(spam.test)[1], digits=2)

# Accuracy for train and test sets
accuracy.train=sum(diag(table(tune.train.full,spam.train$spam)))/length(tune.train.full)
accuracy.test=sum(diag(table(tune.test.full,spam.test$spam)))/length(tune.test.full)
# accuracy train = 0.9720584  FP = 22, .01
# accuracy test = 0.9144307  FP = 29, .02
# best parameters:
#   gamma - free parameter of gaussian radial basis function
#   a radial kernal "raises" some obs above the 2D plane so a
#   hyperplane can be used to separate them. Small gamma means the
#   obs raised have sharp peaks, which means low bias-high variance
#   High gamma means obs have flatter, broader peaks which menas
#   higher bias-lower variance.
#   0.1

### SVM full sigmoid model ###
```

```
tune.train.full = predict(best.sigmoid.full)
tune.test.full = predict(best.sigmoid.full, newdata=spam.test)

conf.train = table(tune.train.full, spam.train$spam)
conf.test = table(tune.test.full, spam.test$spam)
conf.train.perc = round(conf.train/dim(spam.train)[1], digits=2)
conf.test.perc = round(conf.test/dim(spam.test)[1], digits=2)

# Accuracy for train and test sets
accuracy.train=sum(diag(table(tune.train.full,spam.train$spam)))/length(tune.train.full)
accuracy.test=sum(diag(table(tune.test.full,spam.test$spam)))/length(tune.test.full)
# accuracy train = 0.8190003  FP = 299, .09
# accuracy test = 0.8114576  FP = 135, .10
# best parameters:
#   gamma coef0
#   0.1   0.1

### end - Support Vector Machine predictions ######

################ end - Support Vector Machine###############




################ Random Forest ####################

### RF full model ###
set.seed(71)
rf.full = randomForest(spam~., data=spam.train)
print(rf.full)

#determine how many trees for min error rate
plot(rf.full)
which.min(rf.full$err.rate[,1])

#retry model with min number of trees
set.seed(71)
rf.full.2 = randomForest(spam~., data=spam.train, ntree=98)
print(rf.full.2)
varImpPlot(rf.full.2, main="Variable Importance Plot - Full Model")


### RF bkwd vars model ###
set.seed(71)
rf.bkwd = randomForest(spam ~ make + address + all + w_3d + our + over + remove + internet +
  order + report + addresses + free + business + credit + your +
  w_000 + money + hp + hpl + george + w_650 + lab + telnet +
  data + w_85 + technology + parts + pm + cs + meeting + original +
  project + re + edu + table + conference + char_semi + char_exlam +
  char_dolr + char_num + cap_avg + cap_long + cap_total, data=spam.train)
print(rf.bkwd)

#determine how many trees for min error rate
plot(rf.bkwd)
which.min(rf.bkwd$err.rate[,1])

#retry model with min number of trees
set.seed(71)
rf.bkwd2 = randomForest(spam ~ make + address + all + w_3d + our + over + remove + internet +
                        order + report + addresses + free + business + credit + your +
```

```
                w_000 + money + hp + hpl + george + w_650 + lab + telnet +
                data + w_85 + technology + parts + pm + cs + meeting + original +
                project + re + edu + table + conference + char_semi + char_exlam +
                char_dolr + char_num + cap_avg + cap_long + cap_total,
                data=spam.train, ntree=233)
print(rf.bkwd2)
varImpPlot(rf.bkwd2, main="Variable Importance Plot - Bkwd Model")



###### Random Forest Predictions - train/test data set ######

### RF full model ###
rf.full.train = predict(rf.full.2, type="response")
rf.full.test = predict(rf.full.2, newdata=spam.test, type="response")

conf.train = table(rf.full.train, spam.train$spam)
conf.test = table(rf.full.test, spam.test$spam)
conf.train.perc = round(conf.train/dim(spam.train)[1], digits=2)
conf.test.perc = round(conf.test/dim(spam.test)[1], digits=2)

# Accuracy for train and test sets
accuracy.train=sum(diag(table(rf.full.train,spam.train$spam)))/length(rf.full.train)
accuracy.test=sum(diag(table(rf.full.test,spam.test$spam)))/length(rf.full.test)
# accuracy train = 0.9506364  FP =  62, .02
# accuracy test = 0.9477883  FP = 28, .02
# accuracy train bin = 0.950326  FP =  62, .02
# accuracy test bin = 0.9535896  FP = 22, .02

### RF backward model ###
rf.bkwd.train = predict(rf.bkwd2, type="response")
rf.bkwd.test = predict(rf.bkwd2, newdata=spam.test, type="response")

conf.train = table(rf.bkwd.train, spam.train$spam)
conf.test = table(rf.bkwd.test, spam.test$spam)
conf.train.perc = round(conf.train/dim(spam.train)[1], digits=2)
conf.test.perc = round(conf.test/dim(spam.test)[1], digits=2)

# Accuracy for train and test sets
accuracy.train=sum(diag(table(rf.bkwd.train,spam.train$spam)))/length(rf.bkwd.train)
accuracy.test=sum(diag(table(rf.bkwd.test,spam.test$spam)))/length(rf.bkwd.test)
# accuracy train = 0.9509469  FP =  63, .02
# accuracy test = 0.9463379  FP = 28, .02


### end - Random Forest predictions ######


################ end - Random Forest ###############
```