# SPAM Case Study: Arellano, Clark, Shah, Vaughn

Samuel Arellano, Daniel Clark, Dhyan Shah, Chandler Vaughn

2020-06-17

## Table of Contents

## Abstract

The use of email has become the standard for communication in business and personal settings all over the world. As it's coming to take the place of traditional mail services, one side effect is the constant wave of unwanted messages that we receive daily. Spam email has become ubiquitous of our email experience and can lead to user frustration when it overshadows valid emails in one's inbox. Email servers have employed spam detection methods to help automatically detect spam emails and filter them from the user. At the same time, spam email writers are becoming increasingly savvy in breaching these detection methods. In this paper, we will explore a dataset of spam and valid emails and use a recursive partitioning method to learn from a training set and classify unlabeled data in a testing set. After some tuning, we were able to accurately classify emails as spam and valid while also identifying that forwards, perCaps and bodyCharCt were the key factors in deciphering a spam vs valid email.

## Introduction

According to the anti-malware company Malwarebytes, “Spam is any kind of unwanted, unsolicited digital communication, often an email, that gets sent out in bulk. Spam is a huge waste of time and resources.” At its most benign, spam is the digital equivalent of junk mail costing recipients nothing more than time and aggravation. However, spam emails can expose individuals and companies to a litany of attacks including phishing, ransomware, and most recently crypto-jacking. These cyber-attacks can result in significant costs to companies in the form of lost time, revenue, resources, and intellectual property. Without a good spam filter even the most sophisticated layered cyber defense system is vulnerable to a cyber-attack that originates from malicious email carelessly opened by a trusting employee. For this reason spam filters are just as important to cyber security as firewalls .

The simplest forms of filters are list-based filters which as the name implies, take lists of words from a combination of black lists, grey lists, and white lists and compares the words in an email to its list, to determine if the email should be blocked, flagged, or allowed, respectively. These filters require continuous list updates and result in both high false positive and false negative designations. A more effective filter is the heuristic filter that makes use of statistical methods and machine learning algorithms to determine the probability that an individual email is spam. In this study we will create a spam filter based on decision trees and compare it with a naive Bayes-based filter to determine the accuracy and precision of each.

## Background

Spam is generally defined as an email that has been sent on masse to many users with no commercial purpose. Spam emails can also include messages containing attachments that spread viruses through emails. Clearly, spam has become a major problem for users and businesses, which led to the advent of spam filters in email systems. In the past, these filters relied on keywords within the message to identify the spam. This can include list-based filters which can classify the sender, content-based filters, and collaborative filters where users report spam messages which are stored in the database.

Typically, researchers use a taxonomy based on these web filters to present spam from spreading, and there are many different types of classification methods used to detect spam. Methods such as random forests, support vector machines, naive Bayes and decision trees have been used to parse through these taxonomy filters and identify which play the greatest role in defining spam. Our project will leverage decision trees so that we can build a taxonomy flow chart to better understand how the different email features relate to one another in identifying spam.

A decision tree is a graphing and modeling tool that is designed to visualize, and structure models based on a given set of rules. We can use it to classify unlabeled data using the remaining labeled features to detect patterns. In the case of our email spam detection project, we will be primarily utilizing rpart (recursive partitioning package in R) as our primary modeling method. This method outputs a decision tree made up of the features and their values as the leaves. Recursive partitioning is useful because it can be done for both classification and regression and can easily utilize categorical and continuous variables.

Package rpart is one of the most commonly used machine learning packages available in R and it is easily implemented and can be easily tuned so that we can build an optimized model to improve our classification accuracy. Package rpart uses a non-parametric CART algorithm, which will use the Gini Index as the splitting criteria.

## Data

```
setwd("/Users/danielclark/Desktop/SMU/Quantifying_the_World/Unit
5/Week_5_Materials/MS7333_QTW-master/Case5")
#setwd("C:/Users/Akuma2099/MachineLearning/QTW_Project_3")
load("data.Rda")
load("resampling_binary")
load("spam.tuned.clf")

ls()

## [1] "emailDFrp"       "spam.resamp.bin" "spam.tuned.clf"
```

The data consists of 9348 observations. Each observation represents 1 email message with up to 30 variables. The first variable, "isSpam" is a 2-level categorical variable that denotes whether the message was known to be spam. This will serve as the response variable for the study. The remaining variables consist of 16 categorical and 13 numeric variables that represent characteristics of the email such as "isRe" which denotes whether an email contains the "RE" prefix in the subject line, "hour" which denotes the hour of the day in which the email was received, "bodyCharCt" which represents the number of characters in the body of the message, and "perCaps" which represents the percent of characters that where capitalized. More detail of the variables in play can be found below:

- *isRe (logical) - TRUE if Re: appears at the start of a 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 addresses 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 Attachments in the message*
- priority (logical) - TRUE if 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 int he header*
- sortedRec (logical) - True if the recipients email addresses are sorted
- *subFunc (logical) - TRUE if the words int eh subject have punctuation or numbers embedded in them*
- 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 senders address (before the @) ends in a number
- *isYelliing (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
- *isOrigMsg (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 a message
- numDlr (numeric) - Number of dollar signs in the message body.

```
## 'data.frame':    9348 obs. of  30 variables:
##  $ isSpam       : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isRe         : Factor w/ 2 levels "F","T": 2 1 1 1 2 2 1 2 1 2 ...
##  $ underscore   : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ priority     : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isInReplyTo  : Factor w/ 2 levels "F","T": 2 1 1 1 1 2 1 1 1 2 ...
##  $ sortedRec    : Factor w/ 2 levels "F","T": 2 2 2 2 2 2 2 2 2 2 ...
##  $ subPunc      : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ multipartText: Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ hasImages    : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isPGPsigned  : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ subSpamWords : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ noHost       : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ numEnd       : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isYelling    : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isOrigMsg    : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isDear       : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isWrote      : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 2 1 1 ...
##  $ numLines     : int  50 26 38 32 31 25 38 39 126 50 ...
##  $ bodyCharCt   : int  1554 873 1713 1095 1021 718 1288 1182 5989 1554 ...
##  $ subExcCt     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ subQuesCt    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ numAtt       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ numRec       : int  2 1 1 0 1 1 1 1 1 2 ...
##  $ perCaps      : num  4.45 7.49 7.44 5.09 6.12 ...
##  $ hour         : num  11 11 12 13 13 13 13 14 14 11 ...
##  $ perHTML      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ subBlanks    : num  12.5 8 8 18.9 15.2 ...
##  $ forwards     : num  0 0 0 3.12 6.45 ...
##  $ avgWordLen   : num  4.38 4.56 4.82 4.71 4.23 ...
##  $ numDlr       : int  3 0 0 0 0 0 0 0 0 3 ...
```

For better interpretation we renamed and releveled the response variable from "T" and "F" to "Spam" and "valid".

```
emailDFrp$isSpam <- emailDFrp$isSpam %>%
                      revalue(c("T"="Spam", "F"="Valid")) %>%
                        relevel("Spam")
```

A check of NA values by column indicated that there were up to 363 observations with missing values of the following seven variables:

- subSpamWords
- subQuesCt
- subExcCt
- SubBlanks
- numRec
- noHost
- isYelling

This indicates that up to 3.9% of our observations could have missing values.

```
sapply(emailDFrp, function(x) sum(is.na(x)))

##        isSpam          isRe    underscore      priority   isInReplyTo
##             0             0             0             0             0
##     sortedRec       subPunc multipartText     hasImages   isPGPsigned
##              0             0             0             0             0
##    subSpamWords        noHost        numEnd     isYelling     isOrigMsg
##               7             1             0             7             0
##          isDear       isWrote      numLines   bodyCharCt      subExcCt
##               0             0             0             0            20
##       subQuesCt        numAtt        numRec       perCaps          hour
##              20             0           282             0             0
##         perHTML     subBlanks      forwards    avgWordLen        numDlr
##               0            20             0             0             0
```

Though dropping the missing values was an option, we decided to impute the missing values based on random forest classification and regression using a parallel method.

```r
registerDoParallel(cores=4)

df <- missForest(emailDFrp,
                 maxiter=5,
                 ntree=200,
                 parallelize = c('forests'),
                 variablewise = TRUE)

##   missForest iteration 1 in progress...done!
##   missForest iteration 2 in progress...done!
##   missForest iteration 3 in progress...done!
##   missForest iteration 4 in progress...done!
##   missForest iteration 5 in progress...done!

# establish imputed set
emailDFrp <- df$ximp
```

The imputed values were merged into the original dataset to create a complete dataset with no missing values.

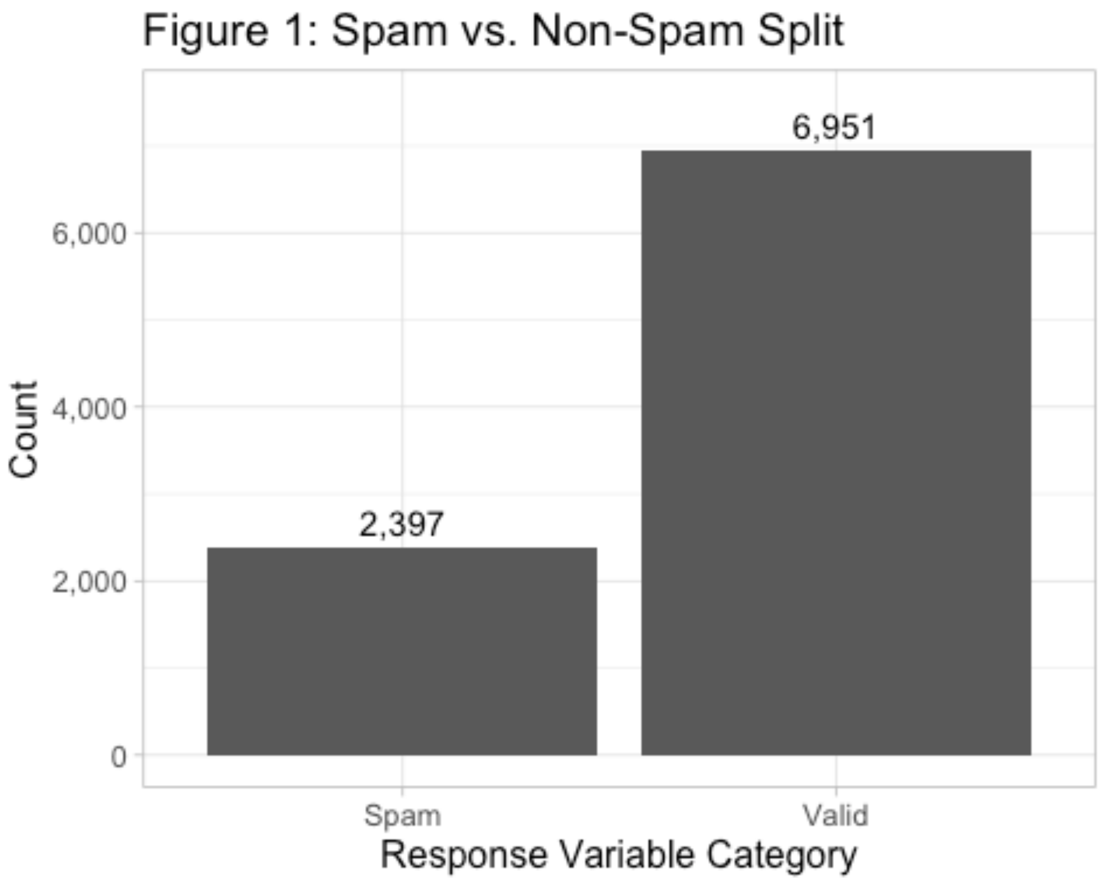
```r
sum(is.na(emailDFrp))

## [1] 0
```

## Methods

Prior to building our rpart algorithm to classify spam and valid emails, we will explore our dataset to detect some trends that we can potentially leverage for our modeling. We will explore the correlation and independence between our predictor variables. We will also look at the relationship between our predictor variables and our isSpam response variable. We will also need to account for the 30 continuous and Boolean factor variables and ensure each are being used in our model to maximize the performance. We will also leverage the rpart tuning parameters to identify the optimal features for identifying spam and valid emails as well as the optimal variable branches in our decision tree.

## Exploratory Data Analysis

After ensuring that no missing values existed in the data set we began our exploratory data analysis by examining the distribution of the response variable.



The chart above shows that we have an unbalanced dataset with nearly 2,400 spam emails compared with nearly 7,000 non-spam (valid) emails. This means that, roughly, 1 of every 4 emails in our set are considered spam. This fact highlights the possible need to account for the imbalance in our future modeling and ensure that we are tuning and training our models such that they are not being rewarded for leaning too much on assigning the valid class to new valid. That said, because the unbalance is not huge, we will not need to apply oversampling methods to address this issue.

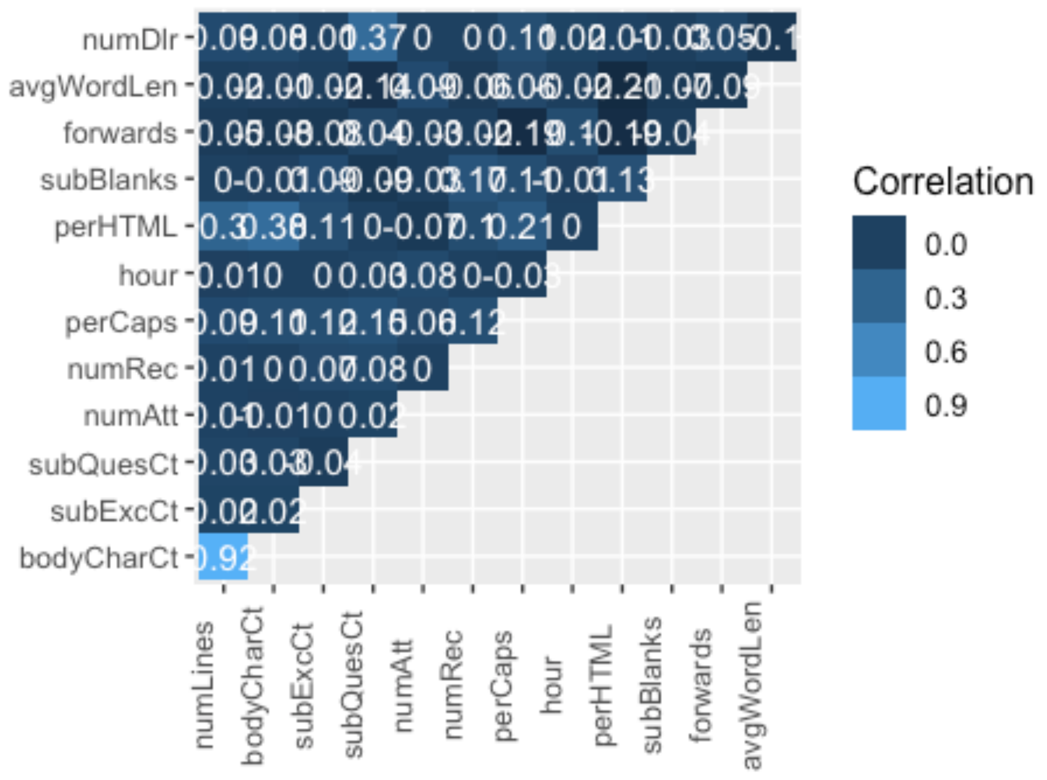
A quick review of the numeric variables indicates that there is a great degree of variation both within and between the individual variables. This indicates that normalization or standardization might be necessary.

```
##     numLines         bodyCharCt        subExcCt         subQuesCt
##  Min.   :   2.00   Min.   :     6   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.:  19.00   1st Qu.:   587   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median :  32.00   Median :  1088   Median : 0.0000   Median : 0.0000
##  Mean   :  66.91   Mean   :  2844   Mean   : 0.1315   Mean   : 0.1378
##  3rd Qu.:  59.00   3rd Qu.:  2192   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :6319.00   Max.   :188505   Max.   :42.0000   Max.   :12.0000
##     numAtt              numRec           perCaps              hour
##  Min.   : 0.00000   Min.   :  0.000   Min.   :  0.000   Min.   : 0.00
##  1st Qu.: 0.00000   1st Qu.:  1.000   1st Qu.:  4.255   1st Qu.: 8.00
##  Median : 0.00000   Median :  1.000   Median :  6.055   Median :13.00
##  Mean   : 0.06579   Mean   :  1.917   Mean   :  8.850   Mean   :12.21
##  3rd Qu.: 0.00000   3rd Qu.:  1.368   3rd Qu.:  9.059   3rd Qu.:18.00
##  Max.   :18.00000   Max.   :311.000   Max.   :100.000   Max.   :23.00
##    perHTML          subBlanks        forwards        avgWordLen
##  Min.   :  0.000   Min.   : 0.00   Min.   : 0.00   Min.   : 1.363
##  1st Qu.:  0.000   1st Qu.:10.53   1st Qu.: 0.00   1st Qu.: 4.208
##  Median :  0.000   Median :13.25   Median : 0.00   Median : 4.455
##  Mean   :  6.517   Mean   :13.87   Mean   :10.45   Mean   : 4.487
##  3rd Qu.:  0.000   3rd Qu.:15.69   3rd Qu.:15.38   3rd Qu.: 4.729
##  Max.   :100.000   Max.   :86.42   Max.   :99.06   Max.   :26.000
##     numDlr
##  Min.   :   0.000
##  1st Qu.:   0.000
##  Median :   0.000
##  Mean   :   1.782
##  3rd Qu.:   0.000
##  Max.   :1977.000
```

### Explanatory Variable Relationships

The correlation matrix for our imputed dataset shows that slight positive and negative correlations between our numeric predictors exist,with the only strong (0.92) multicollinear relationship being between “bodyCharCt” and “numLines”.

Figure 2: Correlation Between Numeric Pred



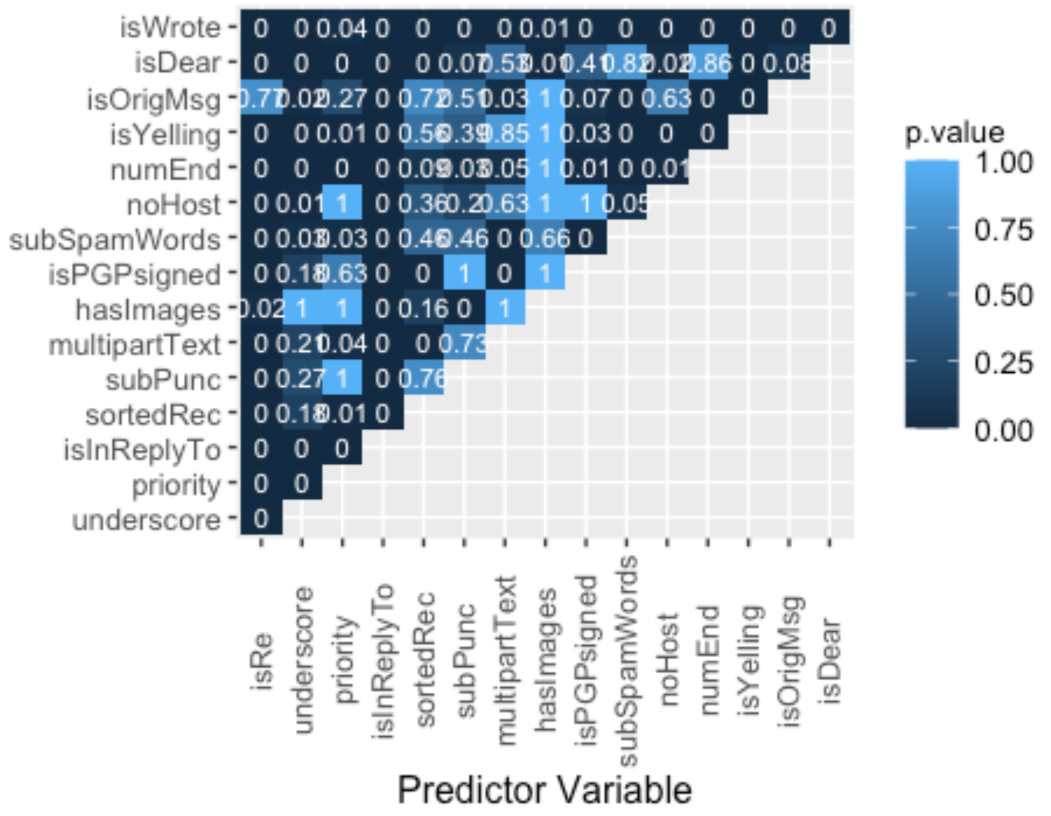
Three additional positive relationships were identified.

- perHTML and numLines (.30)
- perHTML and bodyCahrCt (.38)
- numDir and subQuesCt (.37)

The correlation of these predictors can potentially be accounted for in our modeling procedures, because they might be dependent. For example, numLines and bodyCharct are both functions of the length of the email in question, so the metrics are similar. However, if we use recursive partitioning in our modeling, the collinearity between these pairs of variables will be accounted for, by selecting the most important variable if similar variables are found.

Looking at the non-numerical values in our dataset, there are 16 Boolean values that we can factor into our model. To do so, we will employ the Fisher's exact p-test to show the resulting p-values for our dichotomous variables, which we will use as a numerical comparison similar to the way in which we used correlation on our numerical variables.

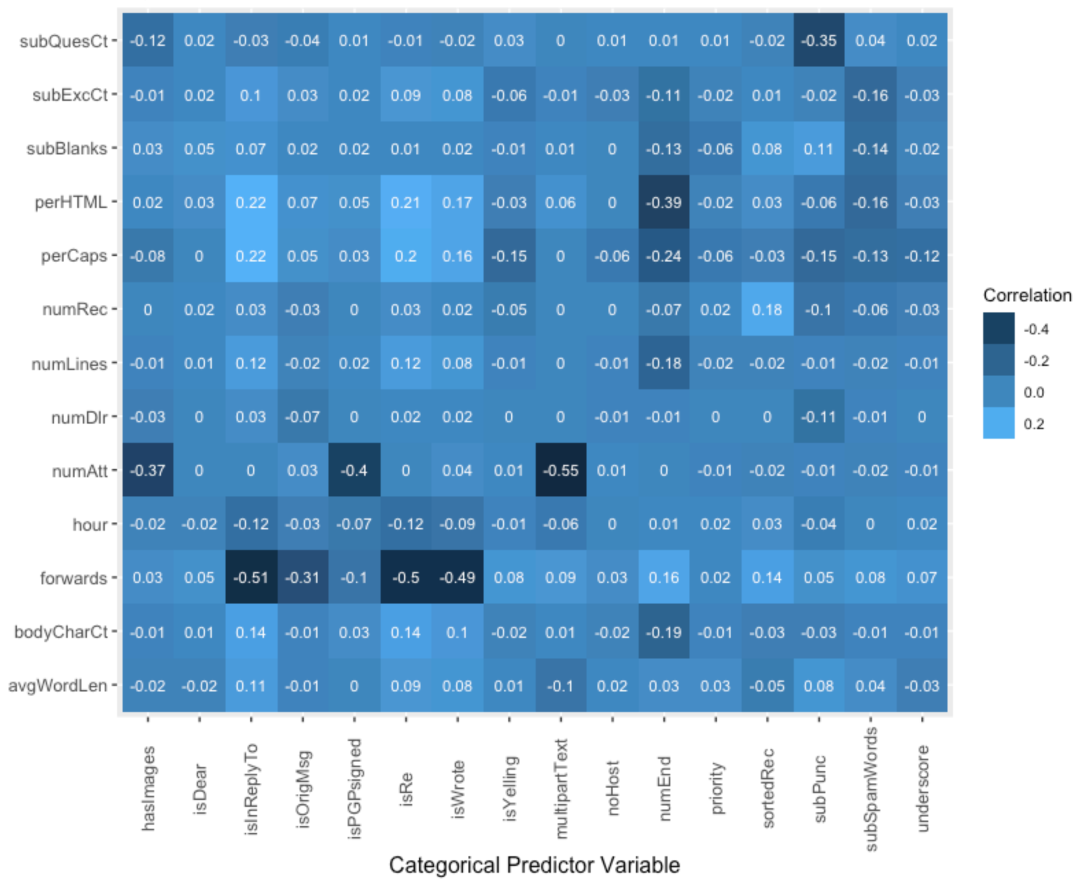
Figure 3: Fisher's Exact Test for Boolean I



The lower p values indicate that we reject the null of a randomized association between dichotomous variables. Here we see that there are some large non-random dependencies for variables such as "isWrote" which indicate whether an email is electronically scribed. Because this is apparent in almost all instances, we can likely remove. However, there are some instances for variables such as "priority" and "noHost" which may be interesting for identifying spam. This would make sense as the lack of host name and sender are variables written by the email sender.

We can also visually review the correlation between factors and continuous variables using a biserial correlation (as we have dichotomous factors for all our non-continuous variables). After review, we see some relationships emerge.

Figure 4: Biserial Correlation - Factor and Continuous Predictors



Reviewing the plot further, we see that the number of attachments (numAtt) has a negative correlation with the Boolean value of "multipartText". Multipart text messages typically do not typically have attachments. In addition, we are seeing that the variable for number of forwards (forwards) has a negative correlation with "isInReplyTo", which suggests that the replies do not typically have a ton of forwards.

Overall, we will look into ways that continuous and categorical variables can be used to predict the response variable "isSpam"" while also estimating variable importance in the rpart package.

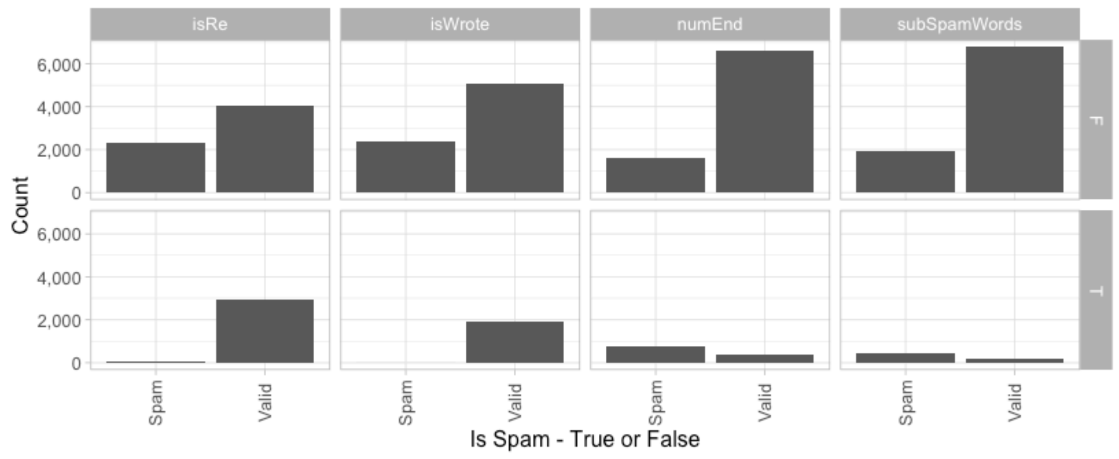
## Response Variable Relationships

As mentioned before, most instances in our dataset were not considered spam. That said, we can visualize the relationships between spam and valid emails for the categorical and continuous predictor variables using different plotting techniques.

First, we will look further into "isRe", "numEnd", "subSpamWords" and "isWrote". The variable "numEnd" indicates whether or not the 'from' email prefix ends with a number, such as 'clark.daniel424@gmail.com.'

Figure 5: Boolean Predictor Variables and Spam Outcomes

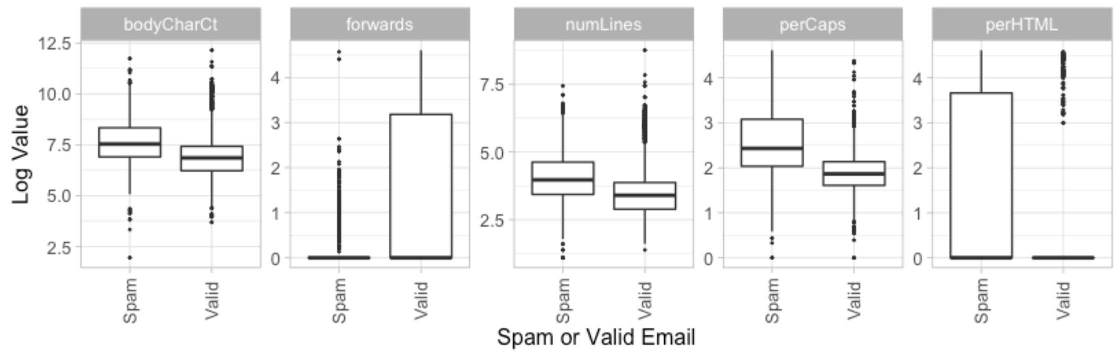
Y Axis Faceting Shows Spam or Valid Email for Each Predictor



In the chart above, “subSpamWords” is a Boolean that indicates when a known spam word is indicated in the subject of an email, such as viagra which would trigger a value for “subSpamwords”. Many of our spam cases happen when factors are set to false. Additionally, the mostly valid emails are related to instances when “isRe” and “isWrote” are set to true. This will prove to be useful when seeing how a decision tree can be split between our categorical variables in the decision function to determine if an email is spam or not spam.

We will also investigate the separation of classes for numeric variables by looking at the log values for our numeric placements in a box plot. We will review some of the interesting numerics in the figure below.

Figure 6: Continous Predictor Variables and Spam Outcomes



The “forwards” predictor variable shows a great concentration of value distribution for the third quartile of messages that are labeled as valid. The “perCaps” predictor variable shows a larger predictor interval for spam than valid emails. We can also see that the median value for spam is higher than the valid messages. On “perHTML”, the mean and median value for valid emails is nearly zero while the range is much greater for the spam messages.

Examination of these predictor variables indicates that we may have some leads to uncovering a predictor and important features related to detecting spam.

## Modeling

### Base Model Naive Bayes

A Naive Bayes classifier is a probabilistic machine learning model that uses Bayes theorem to calculate the probability of a condition being true, given that another condition has occurred. Naive Bayes classifiers work on the assumption that the predictors variables are independent.

First, we execute and 80/20 split stratified on the “isSpam” response variable to account for the unbalanced data. Like our full dataset the distribution of spam emails in both the training and testing sets is close to 25%

```
table(EDFtrain$isSpam) %>% prop.table()

##
##      Spam     Valid
## 0.2564514 0.7435486
```

The training data is then modeled using the Naive Bayes method with 10-fold cross validation as the train control. No other parameter tuning was applied. The resulting model has a 99.3% accuracy score on the training data and is applied to the test data set.

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction Spam Valid
##      Spam  25.1   0.2
##      Valid  0.5  74.2
##
##  Accuracy (average) : 0.9926
```

When applied to the test set the Naive Bayes model accurately classified 466 out of 467 spam emails giving it a .9979 Precision score and a .0021 false positive rate. The model accurately classified 1389 of our 1401 valid email giving it a .9729 Recall score and a .0271 false positive rate. The model performed very well, but the stratified clean data may have resulted in a certain degree of data leakage and may not generalize well.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Spam Valid
##      Spam   473     1
##      Valid    6  1389
##
##                Accuracy : 0.9963
##                  95% CI : (0.9923, 0.9985)
##     No Information Rate : 0.7437
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9901
##
##  Mcnemar's Test P-Value : 0.1306
##
##               Precision : 0.9979
##                  Recall : 0.9875
##                      F1 : 0.9927
##              Prevalence : 0.2563
##          Detection Rate : 0.2531
##    Detection Prevalence : 0.2536
##       Balanced Accuracy : 0.9934
##
##        'Positive' Class : Spam
##
```

With a baseline established we began the creation of a decision tree-based model by starting with variable selection.

## Variable Selection and Model Comparison Setup

In hopes to avoid the complexity of variable selection algorithms to run an algorithm against our email data set, we will set up a fit of a rpart model using the training data an all 29 features and default model parameters. For the default parameters for rpart, we will use a minsplit of 20 along with a complexity parameter of 0.01 with a max depth of 30. The splitting criteria will use the default Gini Index.

for our setup, we will use an 80/20 split between the training and testing set. Since we have an unbalanced relationship between spam emails and valid emails, we can use a stratified sampling of the observations in our training and testing data sets to ensure we maintain the original balance. We have 9,348 total observations, so this would mean that our testing set will have roughly 1,800 observations.

As we will run a series of models in our experiment, we will maintain this distribution of training and testing data for each model. As our rpart is trained, it will provide a listing of variables that are playing the greatest role in deciding upon valid or spam email. The chart below provides the most important features using the rpart base model.

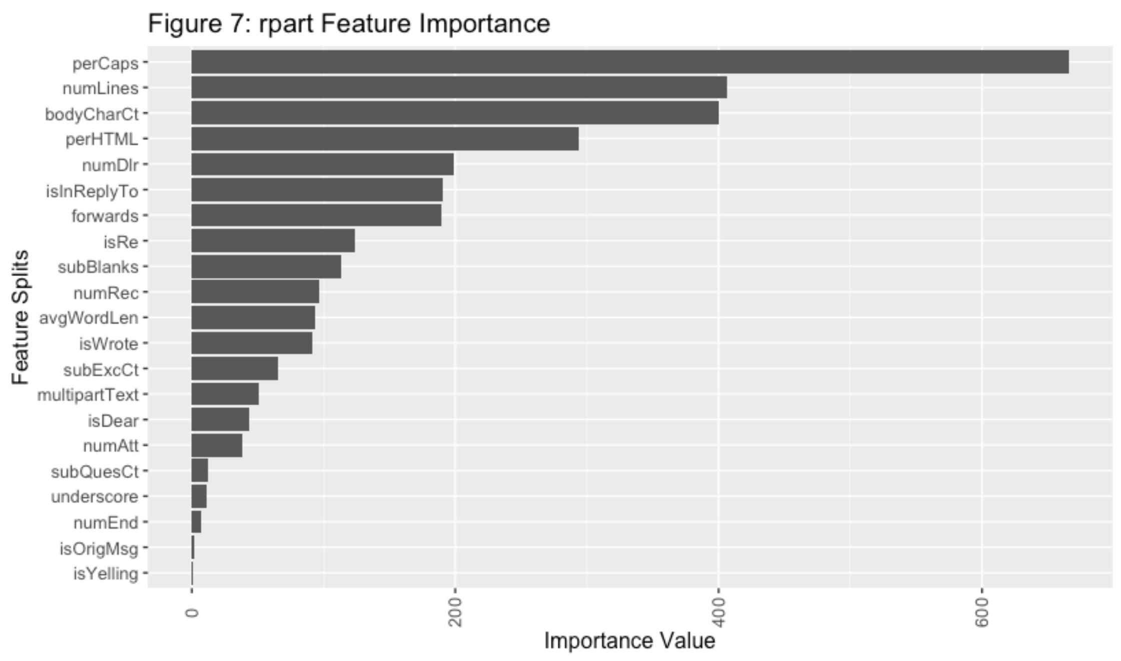


Figure 7 indicates that “perCaps” is the most important feature in defining the spam messages using the rpart base model. “perCaps” is the percentage of capital alpha characters in the body of the email. For this model, the importance is weighted based on the sum of the impurity for each variable split. Secondly and thirdly, we can see that “numLines” and “bodyCharCt” hold the second and third-most important variables in our model.

We will also investigate the rpart control parameters and their ability to classify spam emails based on fitting a separate, optimized rpart model. To do this, we will investigate 4 different parameters for tuning. Below, we outline the description of these.

- Complexity parameter (cp) - A scaled complexity penalty that ranges from 0 to 1. The cp is compared against the error rate related to the previous split. Any split that does not decrease the error rate is not considered.
- Minsplit - the minimum number of observations that need to exist in a node for the split to be attempted.
- Maxdepth - The maximum depth of any node of the final tree, with the root node counted at depth 0.
- Splitting criteria - or the gini or information. It utilizes the gini index to optimize split points and entropy and information gain.

We will use decision trees to avoid overfitting.

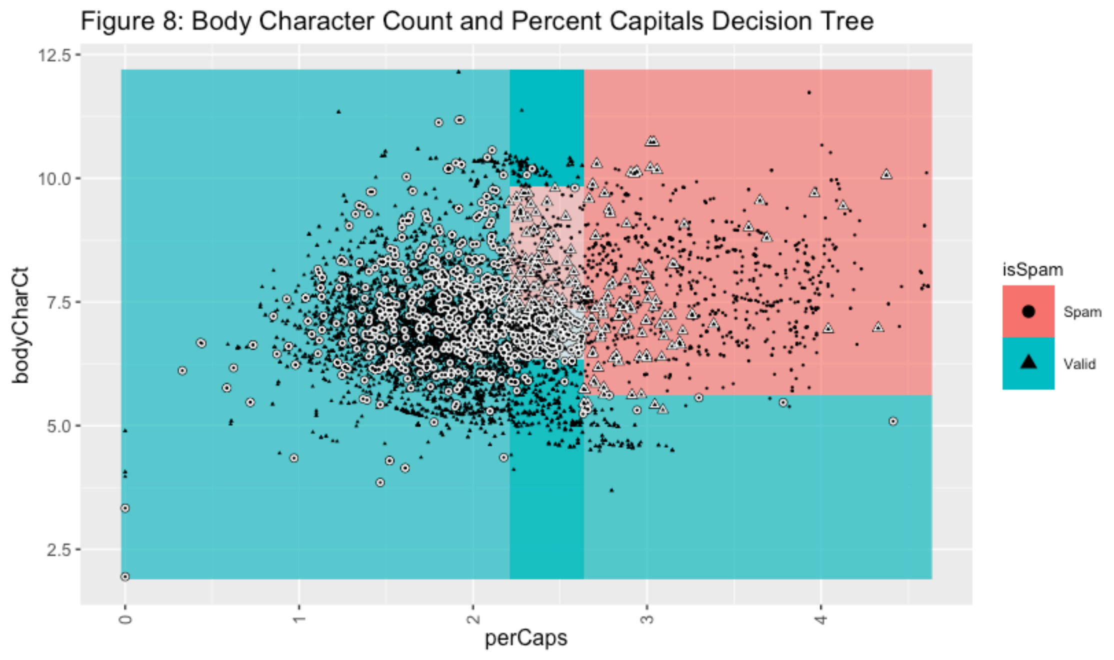
## Hyperparameter Optimization

We will be exploring a discrete list of the 4 parameters of interest to help ensure we are running the models as quickly and efficiently as possible. A grid search procedure will be used in conjunction with a 10-fold cross-validation.

We are going to be using our 4-panel procedure for evaluating classification performance and maximizing true positive classification where "spam" is the positive class. We will measure this using a ROC (AUC) curve and determining which model provides us the greatest area under the curve. A false positive would mean that a valid email will be marked as spam. A false negative would mean that a spam message is misidentified as a legitimate email. The latter 2 would mean that we have a model error.

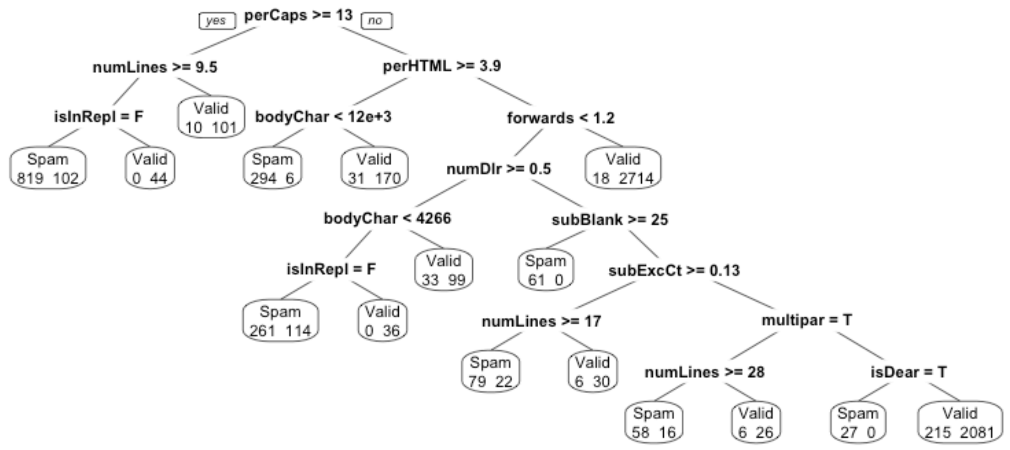
## Base Decision Tree Model Results

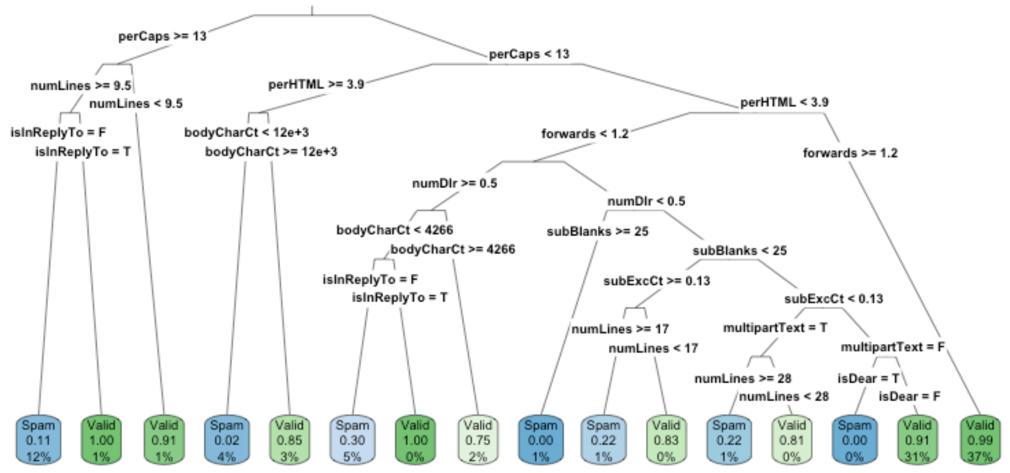
Our base model identified "perCaps" as our most important feature followed by "BodyCharCt". Given this, we will look at an rpart model that uses only "perCaps"" and "BodyCharCt".



The plot above provides a log scale visualization of the classification regions. The observations in white are misclassifications while the color boundaries represents the outcomes for the spam email. The lighter shades of blue and pink represent the lower probabilities for the perCaps and bodyCharCt classes. As shown in Fig. 8, using only 2 features in our model results in a high frequency of false positives as indicated by the number of circles in the blue section. To improve upon this, we will need to include more variables to increase performance. However, these 2 seem to be a good starting point.

From here, we used rpart to build a decision tree with 14 splits as represented below. We will start with the default decision tree as discussed in the previous section and fit it with the training data set.





We can see from the decision tree above that we are using “bodyCharCt” multiple times in the splitting process for the training data. Additionally, we can see there is a bit of misclassification particularly as you go down the tree. Particularly in the sense of when we incorrectly classify spam emails as valid.

We will now leverage the test data to produce a confusion matrix to determine the model performance in the table below. As mentioned previously, we will be reviewing false positive rate, false negative rate, MMCE (model misclassification error) and ROC AUC.

```
##          predicted
## true     Spam Valid -err.-
##   Spam    388    91     91
##   Valid    75  1315     75
##   -err.-   75    91    166

##         auc       mmce        fpr        fnr
## 0.93449257 0.08881755 0.05395683 0.18997912
```

To estimate general error and correction rates, we will be using MMCE and AUC as our KPIs on performance, as they generally indicate how well we are correctly classifying spam and valid emails. As a diagnostic measure, we will be looking at FPR and FNR to keep our models unbiased Looking at our confusion matrix above, we see that the performance is generally good on AUC, the model struggles with false negative rates. This means that we misclassified 91 spam observations of the 479 observations in our training set resulting in a false negative rate of nearly 19%. MMCE is also not particularly high at 9%. Our aim moving forward will be to improve using rpart's hyperparameters.

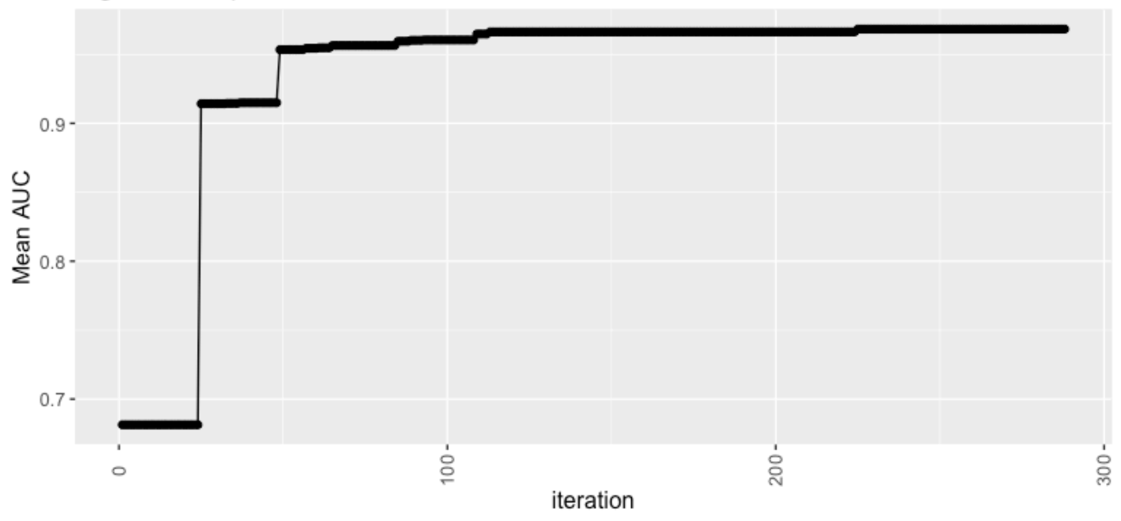
## Hyperparameter Tuning

To help improve our MMCE and FNR, we will adjust our complexity parameter, minimum split, maximum depth and the splitting criterion. To avoid running the model repeatedly and to lighten the code usage, we will be using a grid search procedure to test the rpart on various tuning combinations of our parameters. Below is a list of parameters we will be exploring.

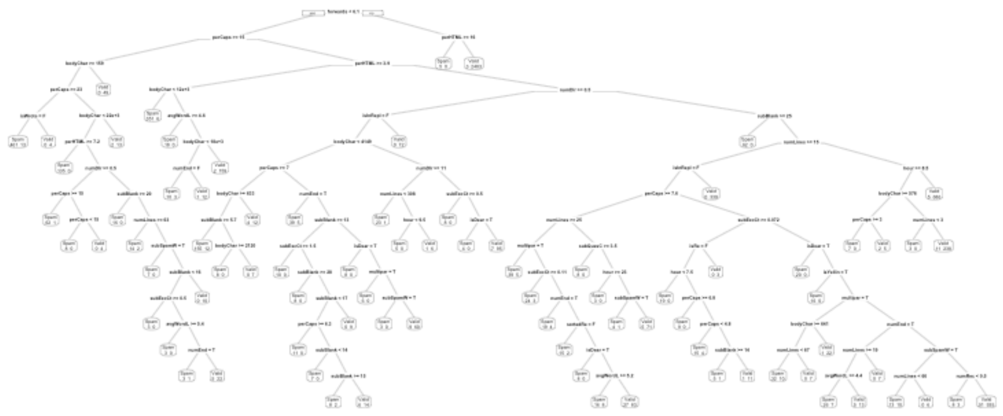
Parameter Search Criteria *complexity parameter (cp) -0.001, 0.01, 0.1, 0.2, 0.5,* minsplit - 1, 5, 10, 15, 20, 30 *maxdepth - 1, 5, 10, 15, 20, 30* splitting criterion - gini, information

On each parameter we ran, we performed a 10-fold cross validation procedure using the training data with AUC as our key performance indicator.

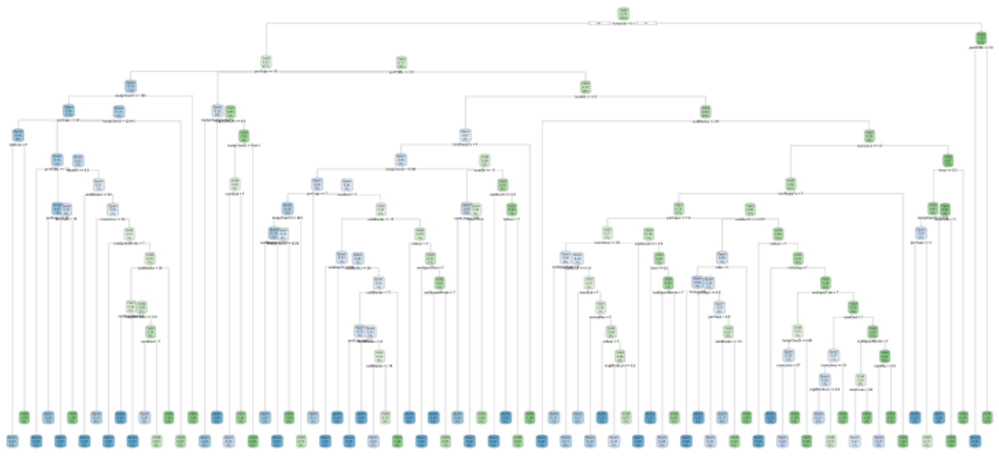
Figure 10: Optimization Path



The optimization procedure starts to maximize at about 0.97 of an AUC using our training data, and our optimized tree is much larger than our base model. With our plot above, we can see that we are using 50 to 100 total splits in our optimized tree. This is may be because our complexity penalty is lower than our optimized grid search threshold at (0.01). This may be causing an overfit to our test set.



```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

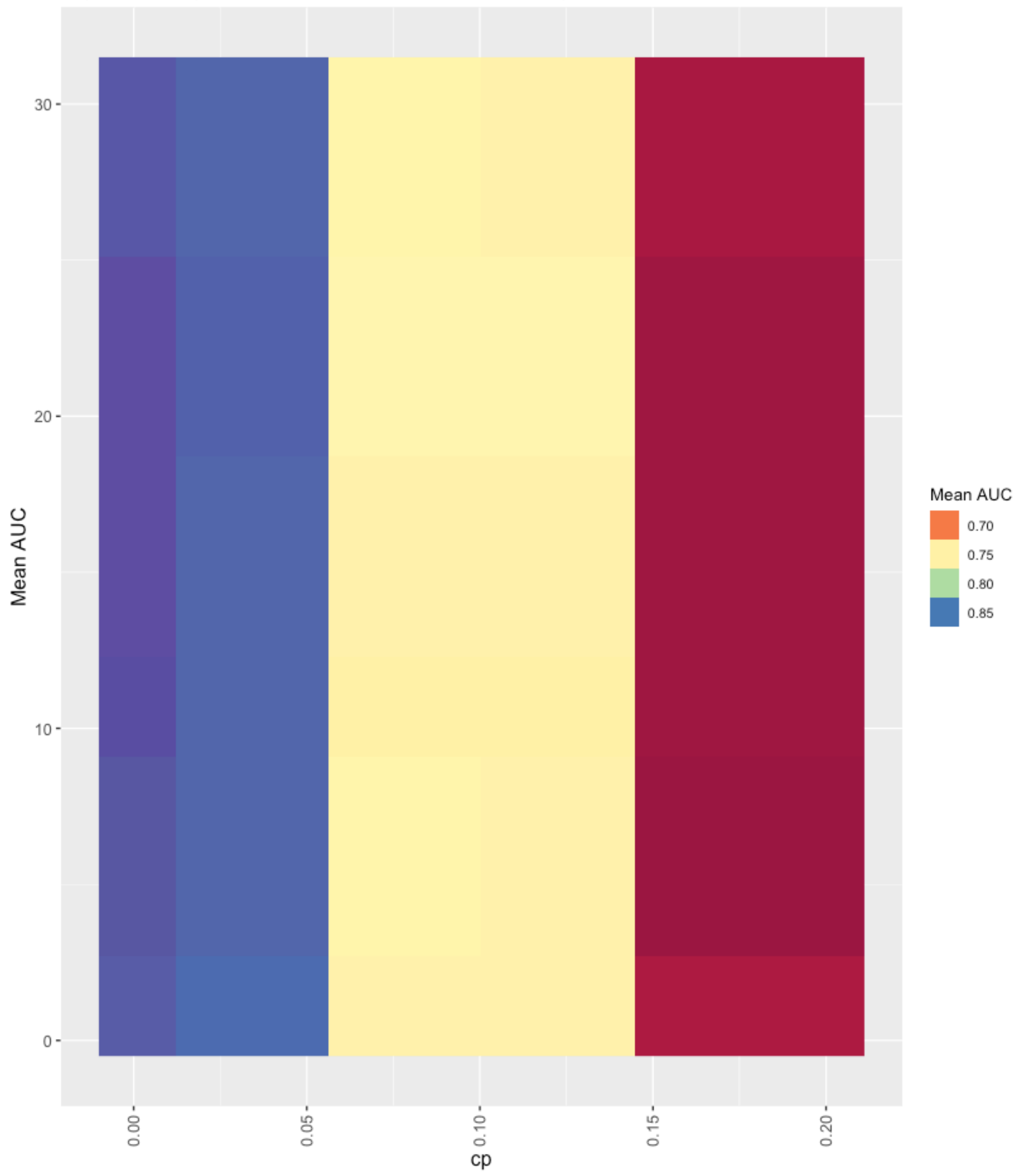


Our resulting model generated an AUC score of >0.96 with a complexity of 0.001. Additionally, the maxdepth for the decision tree node is always >10. About half our models used the information splitting criterion and most of the models used a minsplit of 10 or more. Overall, our optimizations lean toward greater splitting and a medium node depth.

The dominance of the complexity parameter is illustrated in the figure below. The higher AUC is related to a lower CP (or complexity parameter).

```
## Warning: Raster pixels are placed at uneven vertical intervals and will be
## shifted. Consider using geom_tile() instead.
```

Figure 11: Hyperparameter Effects - Complexity and Minsplit



After exploring the rpart tuning parameters during grid search, we significantly increased the AUC score on our optimized model from 0.934 to 0.966, as well as decreased our MMCE from 0.089 to 0.054. IN our base model, we had issues with a large true negative rate (classifying spam emails as legitimate), which was reduced most significantly of all, ranging from 0.190 to 0.129.

| | optimized | base |
|---|---|---|
| auc | 0.966 | 0.934 |
| mmce | 0.054 | 0.089 |

| | | |
|---|---|---|
| fpr | 0.027 | 0.054 |
| fnr | 0.129 | 0.190 |

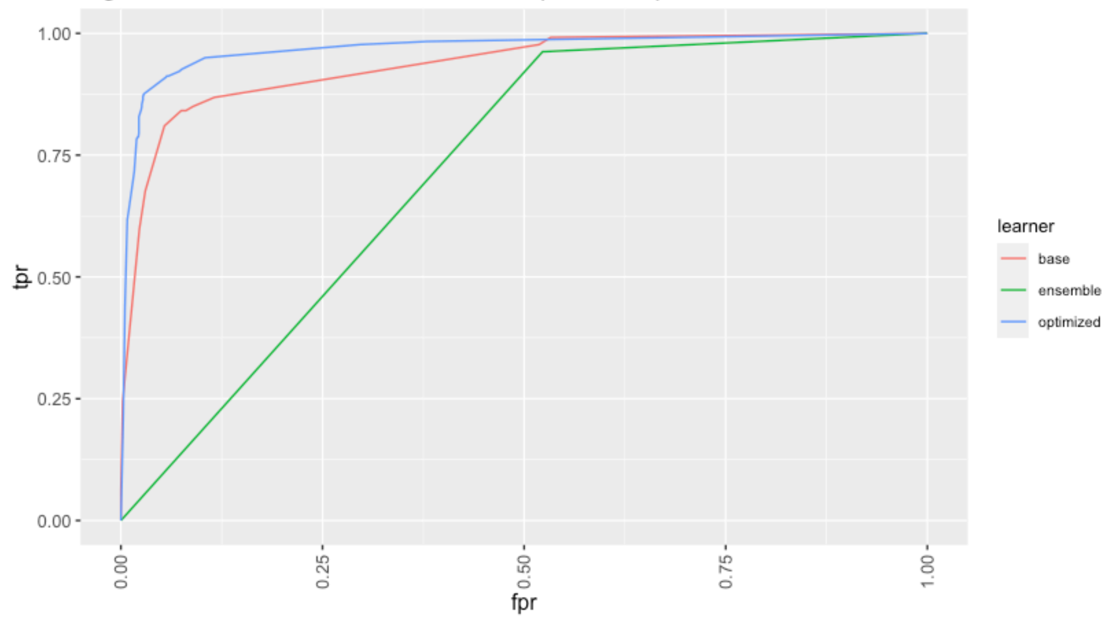
## Ensemble Hyperparameter Tuning

In a parallel package (https://github.com/dhyanshah/MS7333_QTW/blob/master/Case5/Case5-Spam.ipynb) we explored a series of models and estimator tuning procedures to test which model and tuning packages would generate the highest AUC score.

| | Ensemble_Optimized | optimized | base |
|---|---|---|---|
| auc | 0.720 | 0.966 | 0.934 |
| mmce | 0.399 | 0.054 | 0.089 |
| fpr | 0.523 | 0.027 | 0.054 |
| fnr | 0.038 | 0.129 | 0.190 |

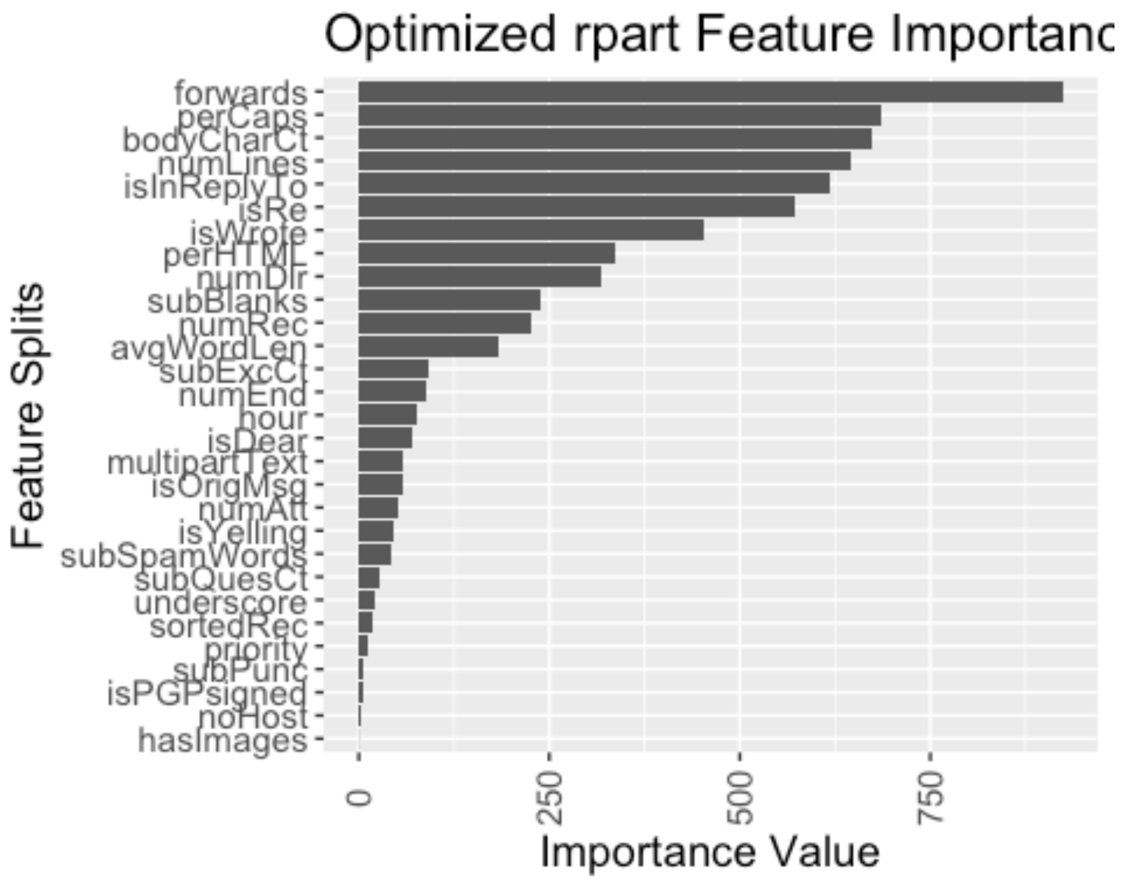
While our grid-searched optimized model exceeded our base model on each metric, we visualized the optimizations using an ROC curve below.

Figure 12: ROC Curves for Base and Optimized rpart Models



In Figure 12, it is apparent that our optimized rpart model outperformed the base rpart (with default parameters) with the greater AUC (blue line compared with the red line). Because the ROC AUC plot is the false positive rate versus the false negative rate, it is evident that the our optimized model most significantly exceeds our base model is by reducing true negatives that were not apparent in the base model.

```
# which variables are most important?
dat <- data.frame(vars=names(splits$variable.importance),
                  importance=splits$variable.importance)
# plot the feature importances
ggplot(dat, aes(reorder(vars, importance, sum), importance))+
    coord_flip()+
    geom_col()+
    theme(legend.position = "bottom",
            legend.text=element_text(size=8),
        legend.title = element_text(size=10),
        axis.text.x = element_text(angle=90, vjust=0.5),
        text=element_text(size=14))+
    ggtitle("Optimized rpart Feature Importance")+
    xlab("Feature Splits")+
    ylab("Importance Value")
```



Along with the “perCaps”" and the “bodyCharCt” in the optimized rpart feature importance chart above, one of the emerging leaders with feature importance is the number of forwards in an email in determining whether it is valid or spam. Reviewing the decision tree plot above, we see that if the number of forwards exceeds 6, then an email has been valid in previous instances.

Even though our optimized decision tree is complex, we still see some of the key features that define the classification of spam and valid emails (to a success rate of 96%). The number of forwards in the email was the biggest indicator in classifying spam vs valid. Reviewing it in our decision tree, we can see that when an email gets forwarded >7 times, it is related to 66% of spam instances. The next most important features, bodyCharCt, numLines and perCaps were the strongest indicators of spam vs valid. Additionally, if PerCaps (which measure the percent of capital letters in an email message), is >15%, it has a greater likelihood of being spam.

## Conclusion

The Naive Bayes model was easy to implement and performed well on both the training and test sets. However, as an out of the box model it tells us little about what variables are of interest and due to our "kitchen sink" approach to variable selection it may fit the data set it came from very well but not generalize well as new data is introduced.

Through a detailed exploratory data analysis, rpart modeling, testing of various models and hyperparameter tuning, we were able to leverage rpart's baseline tuning to achieve a high AUC and a low misclassification rate, false positive rate, and true negative rate. We were then able to tune our parameters to achieve nearly a 96% accuracy in partitioning between spam and valid email messages. However, when we applied the same hyperparameter tunes that we found in our python modeling ensemble, we were not able to achieve the same results as our optimized grid search. We determined that the complexity penalty that we applied to our rpart plays a strong role in our optimized AUC score given our analysis. Using our optimized model, our partitioning was made up into a relatively complex (and difficult to read) rpart decision tree.

If we had a dataset that was more balanced between spam and valid emails, while also being large enough to parse between the number of decision tree combinations to effectively classify between spam and valid emails. Looking further into the decision thresholds and learning curves should help with optimizing the classification criteria and the training data to fit an optimal rpart model.

## Deployment

The boundaries of what is considered "spam" is blurred based on different tolerance rates among users. Therefore, a balance needs to be struck between being too lenient and just classifying commercial emails as spam, and too severe so that we are sending important emails to the junk folder. As the spam mailers are getting more savvy with dodging these algorithms used to detect spam while also being able to send messages on masse, this would mean that our algorithm would need to be constantly updated with new data to ensure that it's detecting and learning from the latest trends and tactics.

These findings don't just have value for the email service providers, but also to email writers as well. Knowing the algorithms that email servers use to parse between spam and valid emails, will help marketers and users better craft their emails in ways that will get through the spam filters and reach their desired audiences. This would mean focusing on elements such as avoiding over capitalization and writing a long body email (using a lot of characters) that will look more valid to email filters.

## References

Nolan, D., Temple Lang, D. DATA SCIENCE IN R: a Case Studies Approach to Computational Reasoning and Problem Solving. CRC PRESS, 2017.