Assignment 3, Predict 454

From: John Hokkanen

To: Dr. Chad Bhatti

Date: October 28, 2017

Summary

Spam email, like junk physical mail, is a scourge upon the transmission medium. The goal of spam identification algorithms is to divert the spam while simultaneously ensuring that all non-spam is admitted to the inbox. This project examines an email-related dataset to understand the basic issues confronting the prediction of spam/not-spam email and whether any particular modeling technologies are better for this task. After multiple models are created using the original and expanded dataset, the author concludes that an expanded feature set using a random forest model would be the most effective way to create a production spam classifier.

Data Quality

The dataset is a small set of 4,601 observations of fifty-seven (47) predictors with no missing values; it has one binary response variable (spam/notspam). The predictors include forty-eight word frequencies (f_), six character frequencies (cf_) and three counts related to capitalization and may be categorized as follows:

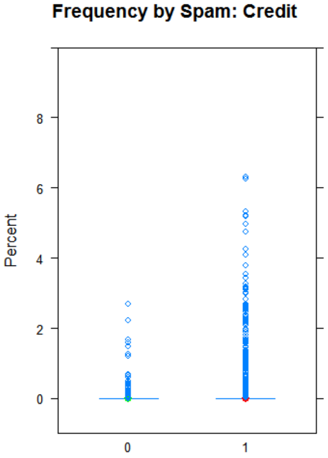
Predictor	Examples	Type	Range	Expectation
Actual words	make, address, all, remove	Frequency	0:100	Predictive value will vary by words with some words having considerably greater predictive value of spam or not spam
Numbers	650, 857, 85, 415, 000	Frequency	0:100	Low predictive value in general, though 000 might indicate business-related spam offers
Proper Names	george, hp	Frequency	0:100	Low predictive value in general. It is possible that spammers have identified names that are more effective than others, and depending on the corpus, names may refer to internal persons with a low likelihood of the message being spam.
Acronyms, abbreviations, or nonsense character strings	pm, 3d, cs, re, telnet, hpl	Frequency	0:100	Low predictive value for the examples presented, though in real datasets, there may be terms like L@@K that spammers have chosen to avoid spam detectors
Characters	exclamation, dollar sign	Frequency	0:100	Predictive value will vary by character
Caps Length	-	Count	0:1102	Higher predictive value. Capitalization is thought of as a method of expressing

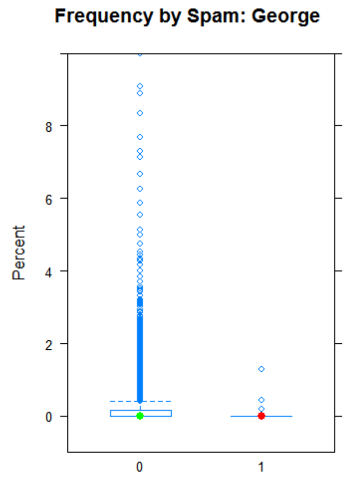
				urgency to act which is a goal of spam email.
Caps Length Average	-	Count	0:9989	Higher predictive value
Caps Length Total	-	Count	0:15841	Higher predictive value

In comparison to the forty-eight words provided, the English language has approximately 150,000 words in use, and emails contain many more unique sets of symbols that may be parsed like words (e.g., lol, ur, etc.). Though the dataset is robust enough to identify approaches that may be useful, the breadth of the dataset is probably too small to produce an effective production model where non-spam is rarely classified as spam (the “false positive” rate.

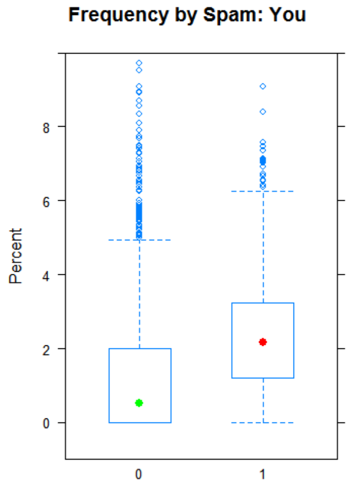
Exploratory Data Analysis

A spam/no-spam classifier is a binary classification model, and one must identify predictors, and ranges within predictors, that have differences in their data between the two classifications. In other words, one must identify predictors where the counts or frequencies for spam=True is different from the counts/frequencies for spam=False. Word and character frequencies in a corpus tend to be quite small, and the following exemplars are illustrative of the problems seen in the box plots:

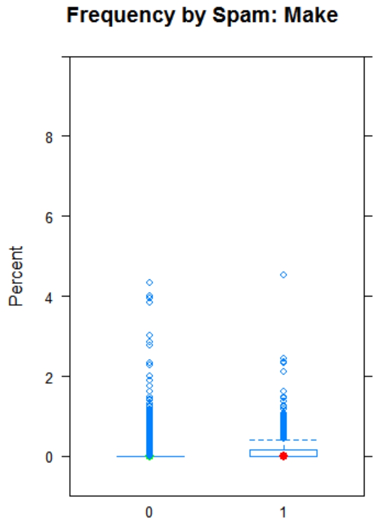
Plot	Analysis
	This plot for the word credit is typical for many words. As you can see the box and whiskers of the plot is completely compressed on zero. This is because nearly every document fails to contain the word credit. However, in a few documents the word occurs in non-spam documents, and in a larger number of spam documents. If it occurs more than one percent of the time in a document, there is a substantial disparity between the spam and no-spam documents. However, as you can see from this very first plot, the data extends well beyond 1.5 times past the interquartile range (the whiskers of the plot). This is the very nature of word frequencies within a body of text. In short, what might be an outlier in other kinds of modeling is the signal for this type of project. The pattern is repeated over and over among the predictors.



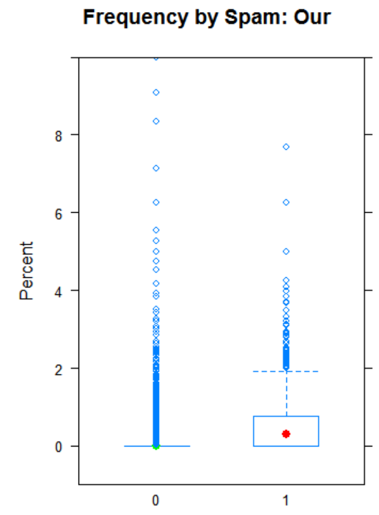
An example with the distributions flipped is seen in the proper name George. Due to the particular corpus that is being analyzed, the non-spam distribution shows the box and one whisker, and, the disparity between the two suggests that the word predicts non-spam.



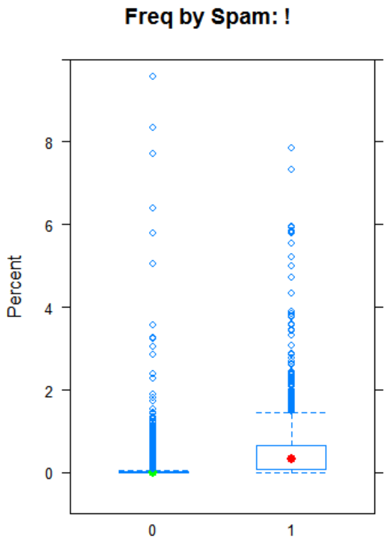
The distributions of the word you are more distinctive. The medians are shifted, with you having a more balanced distribution in the case of spam. In terms of word frequencies in the English language, you is at or near the top, and so while it may be used more often by the spammers, it is also used quite a bit by non-spammers. This means that there is a great deal of noise mixed in with this signal, and separating the noise from the signal may be difficult.



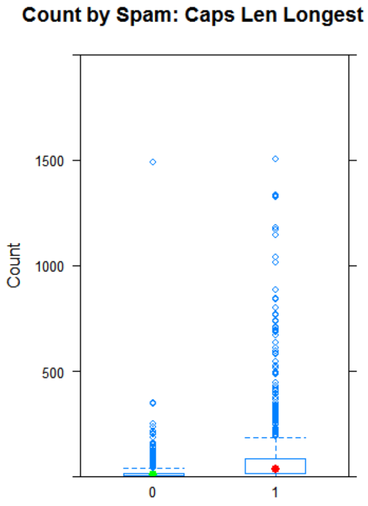
The word make may be more promising. Though the occurrence in non-spam has the same range as the spam distribution, the spam distribution is significantly more positive where non-spam distribution is collapsed around 0.



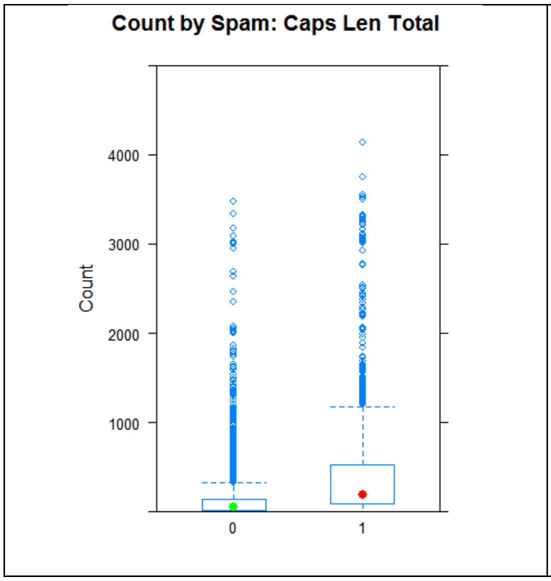
The word our has a more promising set of distributions. In the case of non-spam, the distribution is collapsed around 0, where the word has a much more balanced distribution for spam.



The exclamation mark is even better than the word our, and one can now see both whiskers in the spam distribution. Though used in non-spam, it appears that the exclamation mark is much more common in spam emails.



Two of the caps length counters shown to the left have promise as predictors for the reasons previously set forth. This suggests that counters may be more valuable than frequency percentages.



With the foregoing boxplots in mind, the data is further analyzed in the following table to identify which predictor variables stick out as unusual and in what ways in this regard:

Predictor	Max	Mean	Median	CorSpam	Cor>.1*	3Q>1	Notes
f_make	4.54	0.10	0	0.13	0.2		
f_address	14.28	0.21	0	-0.03			
f_all	5.1	0.28	0	0.20	0.2		
f_3d	42.81	0.07	0	0.06			
f_our	10	0.31	0	0.24			
f_over	5.88	0.10	0	0.23	0.2		
f_remove	7.27	0.11	0	0.33	0.2		
f_internet	11.11	0.11	0	0.21	0.2		
f_order	5.26	0.09	0	0.23	0.2		
f_mail	18.18	0.24	0	0.14	0.2		
f_receive	2.61	0.06	0	0.23	0.3		
f_will	9.67	0.54	0.1	0.01			
f_people	5.55	0.09	0	0.13	0.2		
f_report	10	0.06	0	0.06	0.2		
f_addresses	4.41	0.05	0	0.20	0.4		
f_free	20	0.25	0	0.26			
f_business	7.14	0.14	0	0.26	0.2		
f_email	9.09	0.18	0	0.20	0.3		
f_you	18.75	1.66	1.31	0.27	0.3	X	
f_credit	18.18	0.09	0	0.19	0.2		
f_your	11.11	0.81	0.22	0.38	0.3	X	
f_font	17.1	0.12	0	0.09	0.4		
f_000	5.45	0.10	0	0.33	0.4		
f_money	12.5	0.09	0	0.22	0.2		
f_hp	20.83	0.55	0	-0.26	0.5		
f_hpl	16.66	0.27	0	-0.23	0.5		
f_george	33.33	0.77	0	-0.18			
f_650	9.09	0.12	0	-0.16	0.6		
f_lab	14.28	0.10	0	-0.13	0.5		
f_labs	5.88	0.10	0	-0.17	0.7		
f_telnet	12.5	0.06	0	-0.13	0.7		
f_857	4.76	0.05	0	-0.11	1		f_415

f_data	18.18	0.10	0	-0.12			
f_415	4.76	0.05	0	-0.11	1		f_857
f_85	20	0.11	0	-0.15	0.6		
f_technology	7.69	0.10	0	-0.14	0.7		
f_1999	6.89	0.14	0	-0.18	0.3		
f_parts	8.33	0.01	0	-0.03	0.2		
f_pm	11.11	0.08	0	-0.12	0.2		
f_direct	4.76	0.06	0	-0.06	0.8		f_415
f_cs	7.14	0.04	0	-0.10	0.3		
f_meeting	14.28	0.13	0	-0.14	0.4		
f_original	3.57	0.05	0	-0.14	0.3		
f_project	20	0.08	0	-0.09			
f_re	21.42	0.30	0	-0.14			
f_edu	22.05	0.18	0	-0.15	0.3		
f_table	2.17	0.01	0	-0.04			
f_conference	10	0.03	0	-0.08			
cf_semicolon	4.385	0.04	0	-0.06	0.4		
cf_paren	9.752	0.14	0.065	-0.09	0.4		
cf_bracket	4.081	0.02	0	-0.06			
cf_exclam	32.478	0.27	0	0.24	0.2		
cf_dollar	6.003	0.08	0	0.32	0.3		
cf_pound	19.829	0.04	0	0.07	0.2		
caps_len_average	1102.5	5.19	2.276	0.11	0.5	X	
caps_len_longest	9989	52.17	15	0.22	0.5	X	
caps_len_total	15841	283.29	95	0.25	0.5	X	

*Cor>.1 is the maximum correlation with another predictor variable. It is not displayed when <.1

As the table shows, only a handful of values have median values above zero and this is the point that was made in the first boxplot. Most words are missing from most documents. Only two of the frequency variables (you, your) that have enough usage so that third quartile values are above a frequency of 1%, and these words have unusually high frequency distributions in English texts.

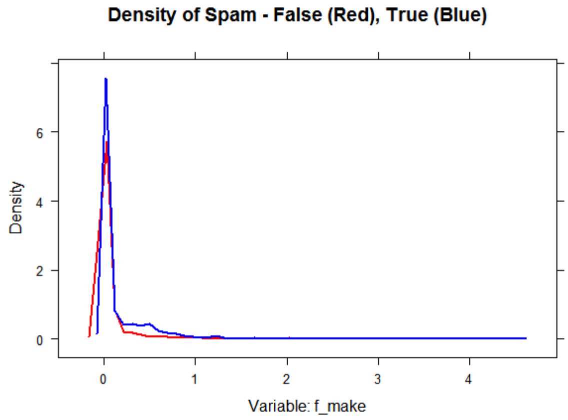
Some of the variables, notably the tech variables, have very high correlations with other tech variables, and this collinearity will likely mean that some or most of them will likely be duplicative. Two of them have perfect correlations.

Turning to the column with orange highlights, quite a few variables have correlations with the response variable that are greater than .2 or less than -.2 (marked in orange). Despite the very small ranges in some of these variables, these features have some predictive value. It should be noted, however, that the use of frequency percentages creates some very odd data for consideration. For example, what does it mean for “edu” to comprise 20% of an email unless that email is 5 words long (e.g., Visit us at abccompany.edu.). High percentages probably reflect small emails more than the reflect actual percentages, and this suggests that some sort of count variable might be of use.

The density plots of some exemplar variables confirm the challenges with these variables. Because of the highly skewed nature of the variables, the plots on the right have been made using a log(variable+1) transformation to show the differences in densities more clearly.

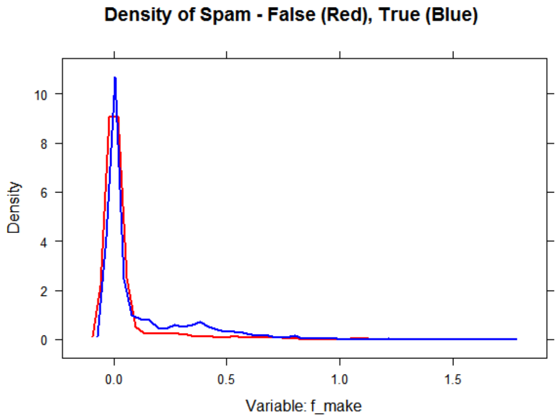
Exemplar Density Plots : No 3rd quartile value >1

Non-transformed value

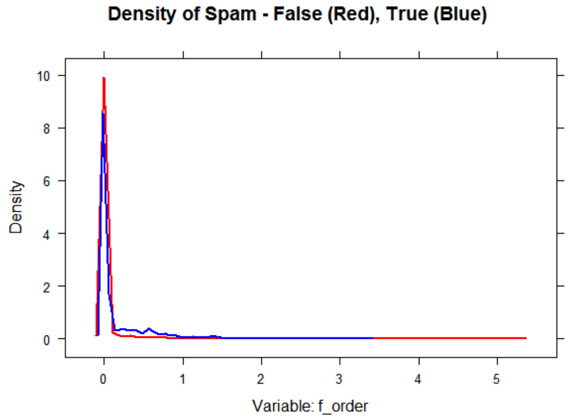


This is an example of a plot where the median value is zero, with no 3rd quarter value exceeding a value of 1. As you can see, the true and false plots are extremely similar and separation of these may be difficult with the exception of a few blue values.

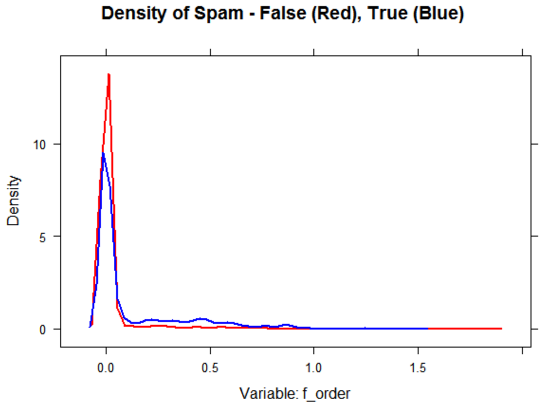
Log(Variable +1)



As you can see, transforming the variable with a log function yields a much more interpretable plot for comparison. It reveals that there is very little separable signal except to the far right.



This is an example of a variable with a correlation with the response that is nearly double the variable above. It is difficult to see why this is the case from the density plot.

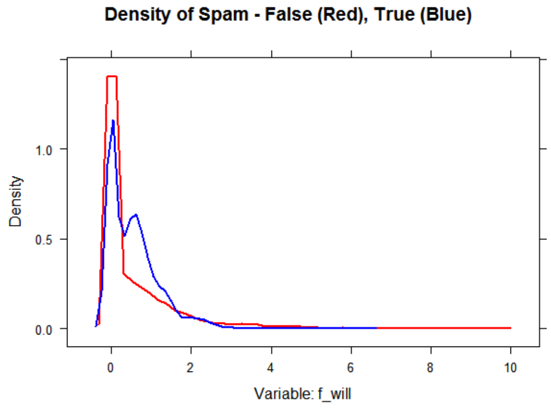


One can now see why the correlation for this variable is stronger. The distribution for true is almost bimodal, and this variable may benefit from segmentation.

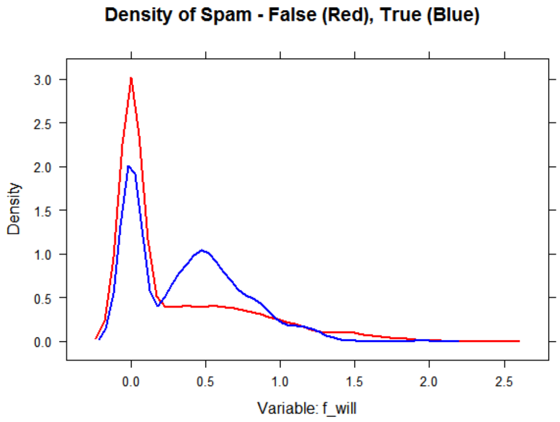
The variables that have a median values above zero all appear to have have potential predictive power.

Density Plots of Specific Variables with Median Values >=.1

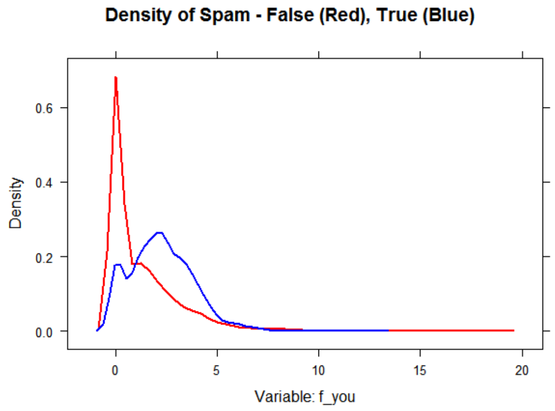
Non-transformed Variable



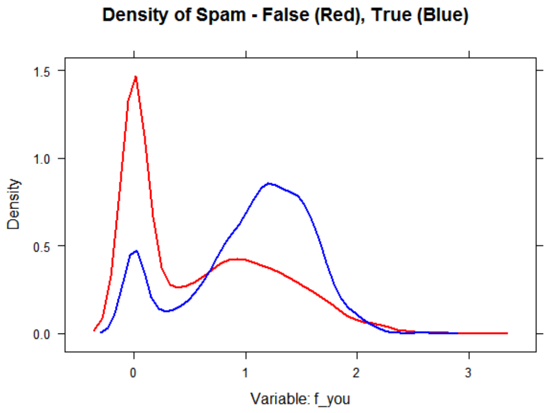
Log(Variable+1)



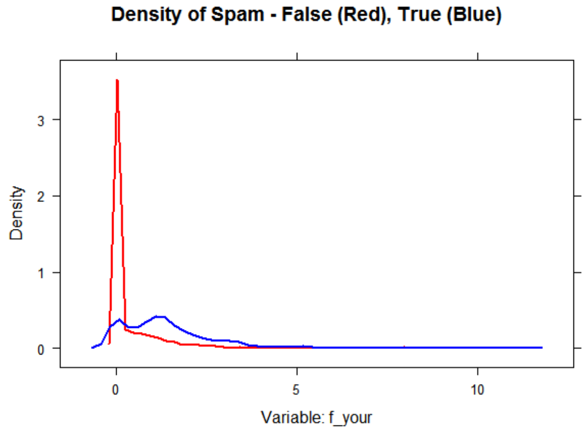
Segmentation of this variable may prove beneficial to enhance the discrimination value.



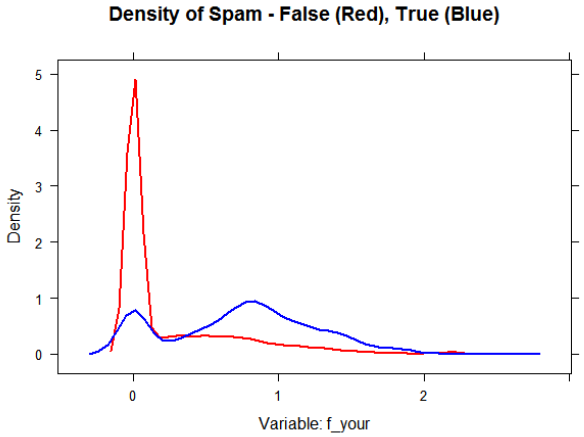
This density distribution shows the problem previously identified in the box plot for you.



Though different in distributions, the word has high frequency and the overlap of the two distributions is significant.

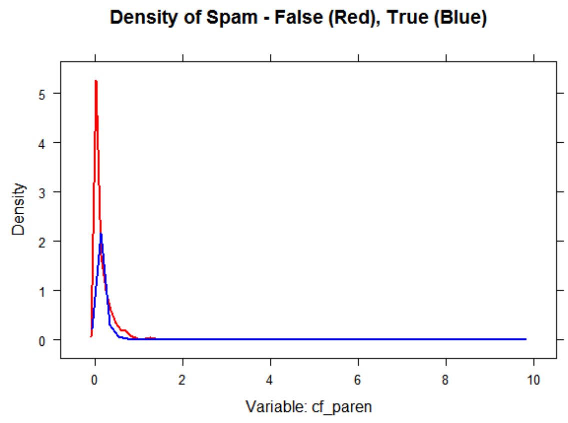


As this variable grows in magnitude, its value as a

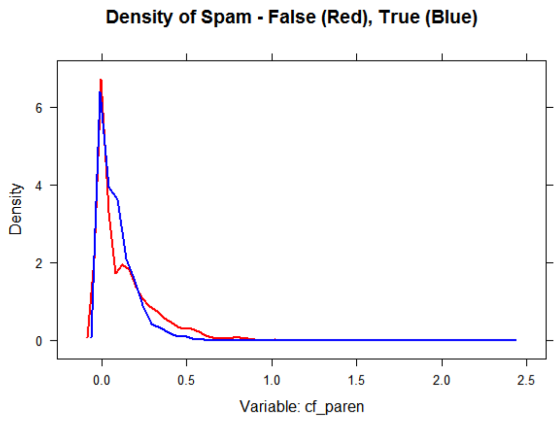


Again, the distribution was more visible with the discriminator may be good. Similar to you, but the high end is quite extended.

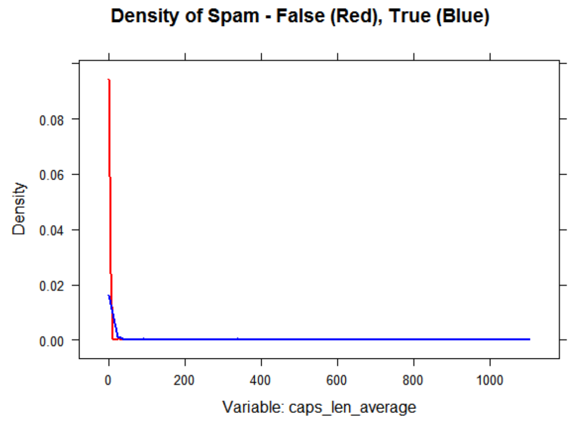
non-transformed version.



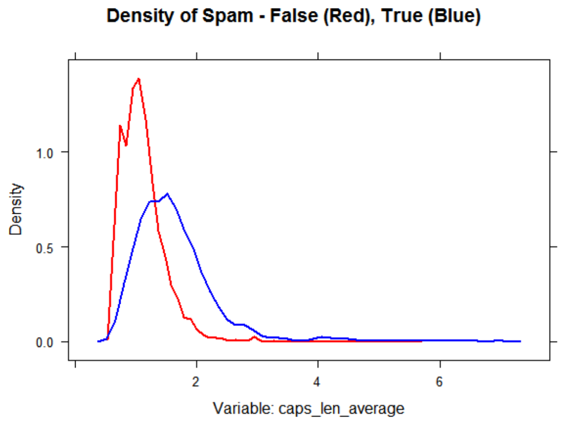
Initially, this variable looks like it might have some predictive value.



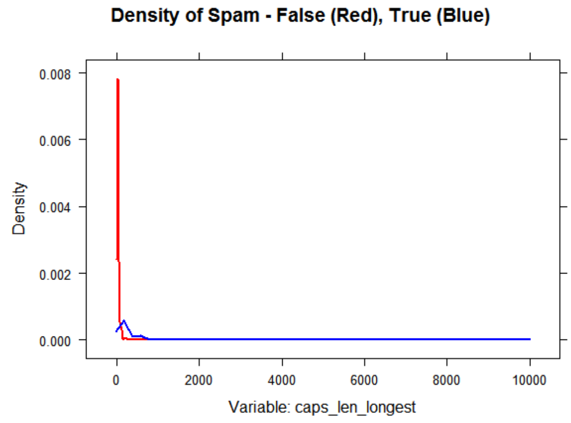
The log version of the plot shows that the variable may not be very promising.



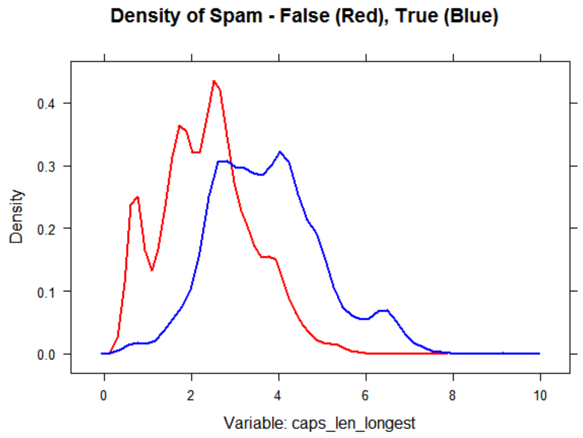
This variable warrants a closer examination



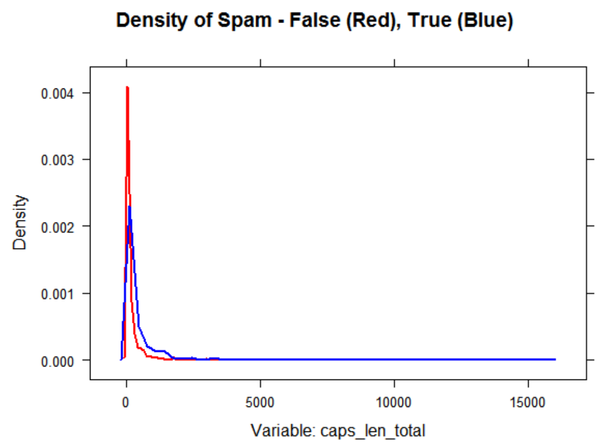
The caps length variable definitely has distinct distributions.

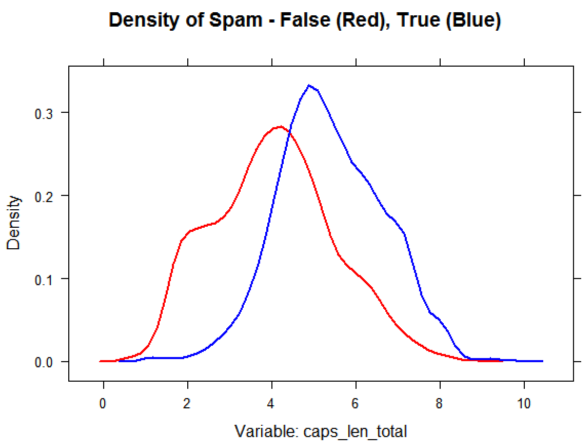


Worth a closer examination.



The caps longest variable appears to have very strong potential discrimination value.





Difficult to examine with the non-transformed plot. Nice distinct distributions.

This initial analysis reveals the central issue of this project. There are many, many weak indicators that have a signal for one or the other classification and the distributions have considerable overlap.

<u>Outliers</u>

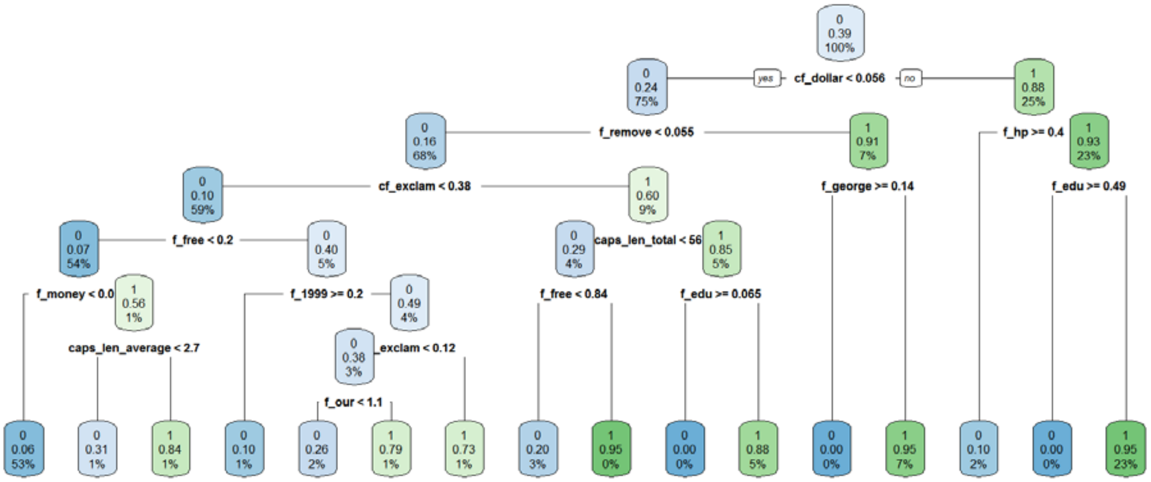
This problem inverts the usual relationships of data to predictors. For many prediction problems, the predictors have distributions that are normal or quasi-normal. For text analysis, the vast majority of the predictors have zero values, and predictive power arises from differences when these low-occurrence variables occur. As the box-plots show, variables are so prone to zero values that predictive power comes from outliers. The words that occur with regular frequency like “a” and “the” are generally considered to be noise words due to their common occurrence. Thus, for most modeling problems, extreme values are a problem, but in the world of text analysis, the most valuable terms are those that are infrequent. In other words, classifying spam/not-spam means identifying extreme values and how those extreme values relate to one another.

Because the underlying email is unavailable for inspection, it is impossible to know why the unusual frequencies are errors or outliers, and, as noted previously, the fact that a variable has an unusually high frequency may be more indicative of the email length than repeated occurrence of the word or character. Finally, the author has received emails with banners or separators made from exclamation marks or other characters; these emails are not errors but are simply emails crafted by humans. Any spam detector that would work in real-world conditions needs to consider the full range of emails regardless of how wacky they may appear statistically.

Because it is impossible to know whether these data records involve outlier variables (with the possible exception of the capital length variables) and because the best indicators may, in fact, be outliers, the use of a non-parametric model may be a less risky and more valid approach to the modeling of this problem.

Tree Model

To further the exploratory data analysis, a relatively simple tree model was trained against the data using cross validation and a tuning length of ten with the following results:



This model confirms what we have already speculated about the nature of the problem. First, in examining all of the predictions of Spam=True (where the value in the terminal node is 1), none of these predictions are at 100%. This suggests that discrimination between the two classes is complex, with many documents of both classes sharing the same words. Most of the terminal nodes reflect this complexity with preceding branches (i.e., interactions between completely differing variables) preceding the terminal node. The false positive rate (falsely flagging a non-spam document as spam) is a significant problem; one terminal node shows accurate predictions only 73% of the time. This confirms that the project involves a great deal of complexity with relatively weak predictors. It further suggests that expanding the dataset may improve the predictive power of the final model.

The tree above shows the most discriminating features at the top of the tree (e.g., dollar sign, remove, exclamation mark). The branch interactions show that some seek to avoid false positives by using non-spam words to classify non-spam, and other branches seek to remove false negatives by winnowing the buckets to higher frequencies of spam. This suggests that a shallow tree may be inadequate because all of the TRUE spam bins in this model have false positives within them based on the percentages indicated. An effective spam detector must have the lowest possible false positive rate.

By increasing the tuning length to twenty, the problems persisted. With this tuning length, 37 of the 57 predictors were used, creating almost 200 decisions with 96 terminal nodes. Yet even with this more complicated tree, the tree model was continuing to report false positives.

Feature Expansion

One of the difficulties with the dataset provided is the fact that most of the features are simply frequency percentages of the word within the document. In search analytics these frequencies are used heavily, but in conjunction with frequencies within the document collection as a whole (the “corpus”). For example, a word that is barely observed in the English language that appears frequently in a document tells one much more about the document especially when one is trying to identify the most relevant search terms.

In this case, we are provided only the frequency within the document. As an experiment, the author created a relative frequency value by dividing the frequency value for word by the mean frequency of the word for all records, but these new variables had as much noise as they had signal and did not prove useful. Additional examination would be required to determine how to effectively transfer the insight from search algorithms into this classification context.

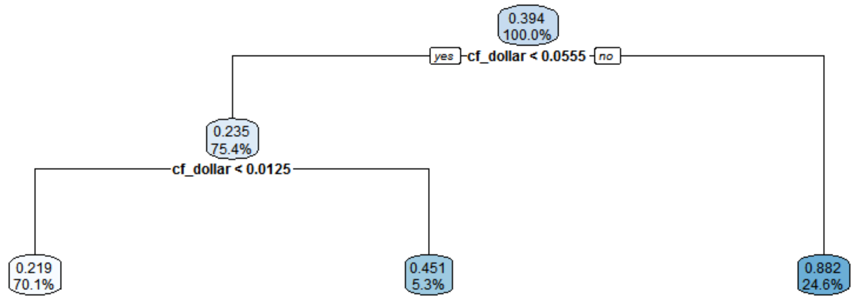
A number of other potentially relevant additional features also came to mind based on past experience with text processing and this particular collection and were created:

Feature	Reason
PercentAccounted	This is simply the sums of the percentages so that there is some awareness of what portion of the document has been reflected in the features that are presented
PositiveCount	Using the positive correlations above, the number of features from that set of correlations that reported a frequency>0 were summed. Thus, if 5 features had reported values, then the count is 5. This is sort of a momentum feature.
NegativeCount	This is the same as positive count, but for the negatively correlated features. This is an momentum feature for defining non-spam.
CapsCount	Total caps length / average length. The goal is to get an idea of how many capitalized sequences existed in the email.
CapsPercentage	Caps longest / Caps total length. The goal is the same as the prior feature, but calculated differently.

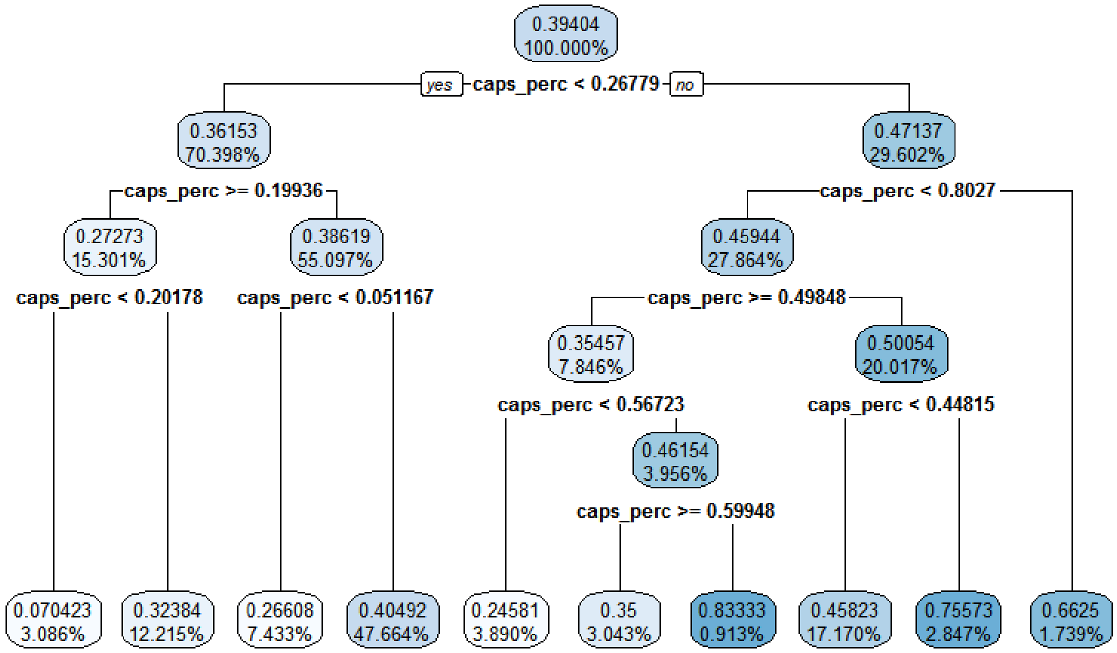
Non-linearities

To identify non-linearities for feature expansion, a simple tree was constructed to identify the most important variables. With the list of most important variables, combined with the new variables, tree models using these single variables as predictors to predict the response variable were created. The branches of these trees were then examined to identify non-linear segmentation that might have significant value for prediction. An example tree for the word dollar to identify nonlinearities is as follows:

Tree Plot for Dollar



Because of the importance of this variable, dummy variables for all of these non-linearities were created, and this procedure was continued for the other variables. In the case of some of the variables, not all non-linearities were modeled; the goal was to identify the segments that had sufficiently robust discriminatory value between spam and non-spam. For example, the perc_count tree is as follows:



If this case only a few of the most potent segments were used (e.g., .45<caps_perc<.80, .80<caps_perc).

As a result of the creation of wholly new variables and nonlinearity segments, approximately 35 additional variables were constructed.

Models

Four predictive model types were applied to the problem: 1) logistic regression; 2) a tree model; 3) a support vector machine; and 4) a random forest tree model. All are discussed below.

Logistic Model

Within the logistic model framework, two different approaches were taken to create logistic models. The first was intensely manual and labor intensive, but was conducted not only for performance comparison but also to glean additional knowledge about the problem domain. It involved no standardization, but instead used glm package to manually iterate (with manual inspection of the model summary) first through dozens of models to identify a model with no singularities so that the glm function would not fault. Then, a stepwise logistic regression was used to reduce the predictor set, and then additional manual intervention added a few variables back into the model.

The alternative, second approach was to automate the process using the glmnet package. First, the dataset with all available variables was standardized and placed into a matrix. A tuned lasso model was run iteratively to identify variables for removal, and once those variables were removed, cross-validation was used to tune the model coefficients to create a final model.

A second pair of models were created by using only the variables from the original data set. The goal was to assess whether the expanded feature set was providing any gain to the final results.

For the hand-crafted model, of the 97 available predictors, 48 variables were used, 22 of which are from the original data set. Thus, more than half of the variables have been modified in some form to improve the predictive power of the model. The full list of variables for the model is set forth in Appendix A. In the case of the hand-crafted model using the original variables, 35 of the original 57 variables were included.

The automated procedure using standardized data and the lasso deleted only four of the original variables and only four of the new variables to produce a model with 89 variables. The original-variable, standardized model retained all of the original variables.

The following table provides relevant training/test performance results as well as comparison values to the stepwise regression generated with only the original variables:

	AUC	**F1 Measure**	**FalseNegRate**	**FalsePosRate**
	Train/Test	**Train/Test**	**Train/Test**	**Train/Test**
Manual - Original Data	.951/.952	.870/.870	.079/.073	.106/.102
Manual – w/ Expanded data	.989/.984	.942/.935	.055/.060	.039/.046
Automated – Original Data	.932/.919	.918/.904	.070/.085	.058/.070
Automated – Expanded Data	.953/.951	.944/.943	.051/.049	.038/.043

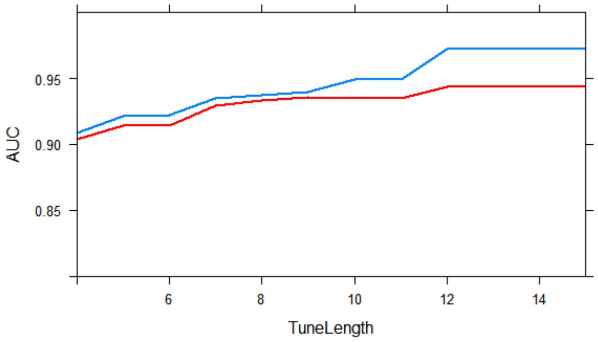
The performance metrics to compare models involve four different metrics. The AUC is the area under the ROC curve which assesses the total discriminatory value of the model. The F1 measure calculates a metric of accuracy that balances both precision and recall. The false negative rate shows the percentage of emails deemed to be non-spam emails that are in fact spam, and the false positive rate shows the percentage of emails that would have been sent to the junk folder as spam that are in fact non-spam emails. Though overall accuracy and total predictive power are generally important and the latter two measures depend upon them, the last measure, the false positive rate, is particularly important for this problem.

As the table shows, the logistic regression using the basic set of variables simply could not compete with the expanded variable set. It is interesting that the manually-tuned model performed better on the area under the curve, but the automated process using the enhanced data set simply outperformed all of the other models on the other measures. This is likely due to the fact that it included nearly every variable and was able to tune all of the separation power from the weak variables. To some extent, the author is pleased that his manually-tuned model came as close as it did. While he learned a lot from doing this about the problem domain, it was a profoundly inefficient expenditure of time given the speed of the automated process; this was an important lesson to learn.

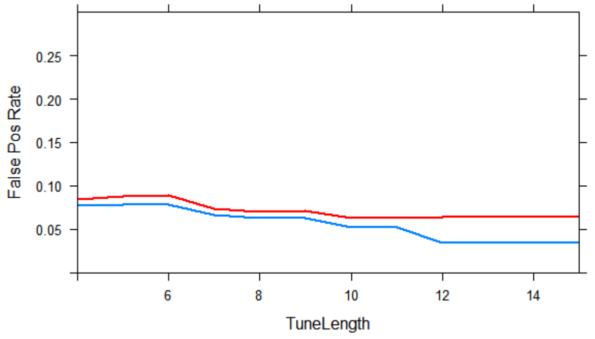
<u>Tree Model</u>

To create the tree model, the enhanced data set was run through a ten-fold cross validation tuning procedure at various tuning lengths. The performance results are plotted below:

Tree Model Performance for Train (Blue) and Test (Red)

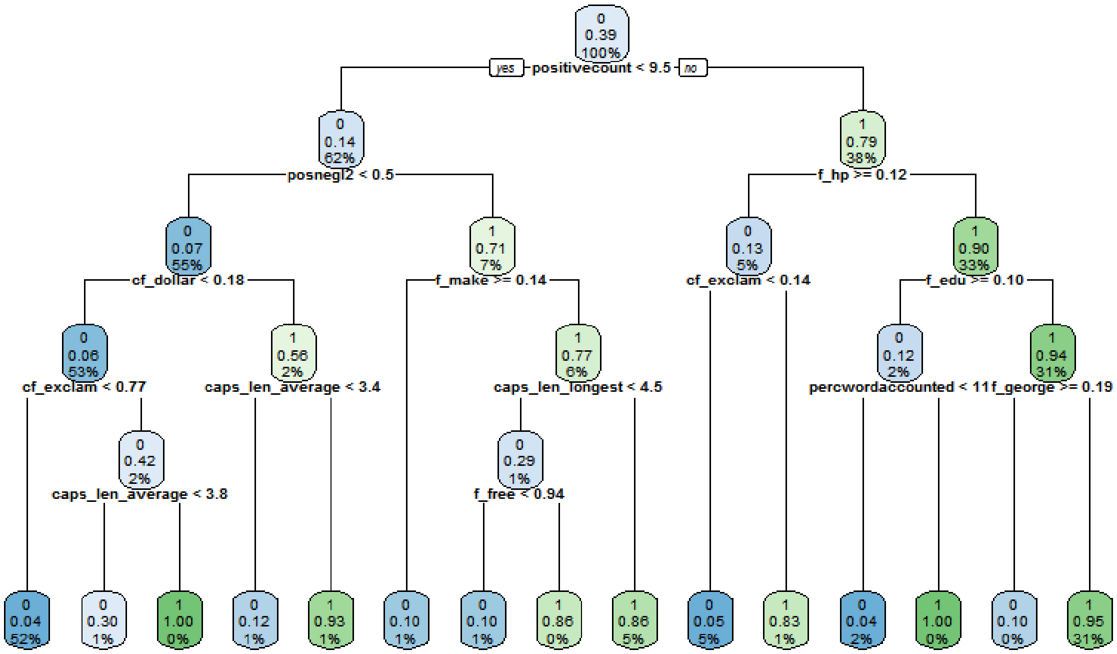


Tree Model False Pos. Performance: Train (Blue) and Test (Red)



As you can see from the upper graph, the area under the curve performance for the test data (red) begins to flatten at a tuning length of 10, and the false positive rate on the test also flattens at the same time for the test data. The results for the training data (blue) continues to rise for AUC and fall for false positive rate, respectively, but that is to be expected as the model is being overfitted to the training data. A similar approach was taken with tuning a tree for the original unmodified dataset and those results are reported in Appendix B.

Using the tuning length of ten, the tree model that was generated is as follows; it was remarkably shallow:



Yet despite its shallowness, its results are surprisingly effective. Those results are compared with the best-tuned tree for the original data which was a tuning length of 16. The tree for the original data is significantly deeper but that is necessary because the original variables may be less predictive. The performance results for the trees are as follows:

	AUC	**F1**	**FNR**	**FPR**
	Train/Test	**Train/Test**	**Train/Test**	**Train/Test**
Tree w/				
Original Data	.973/.948	.941/.905	.047/.084	.044/.069
Expanded Data	.949/.935	.927/.913	.063/.079	.053/.063

The expanded dataset allowed the tree model to perform better, though there was less of a difference between them than for the logistic model. This is to be expected because the tree model can inherently handle variable interactions and nonlinearities. However, the expanded data items like positive and negative count go beyond mere interactions and nonlinearities, and as the table of the top twelve features shows below, both positive and negative count featured prominently and this helps explain the enhanced performance:

Top 12 Most Important Features for Tree Model

Feature	Imp	Feature	Imp	Feature	Imp
positivecount	650	f_all	322	posnegl2	152
posl2	427	f_our	310	Negativecount	97
cf_dollar	393	f_hp	173	f_hpl	93
caps_len_total	339	f_hpl1	173	f_edu	74

Support Vector Machine Model

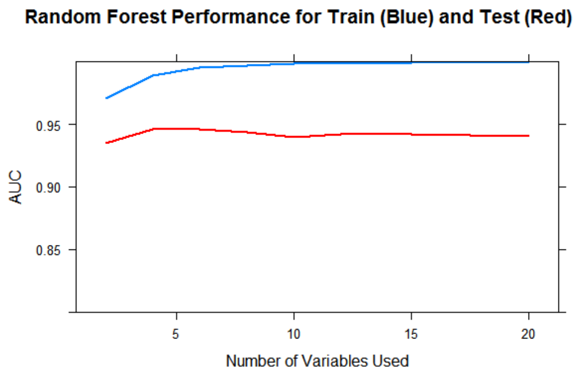
The support vector machine model ("SVM") has renowned classification capabilities. Nevertheless, it should be noted that it might not lend itself to a production implementation involving training on millions of training records as the model tends not to scale as well as model types like the random forest. For this project, the SVM was provided the entire enhanced data set with one variable (f_857) removed because of its collinearity. First, the SVM was tuned with ten-fold cross-validation training using the train function. The same process was followed with an SVM model using the original data set. For both models, the radial control was used rather than the linear model since it is more comprehensive and the goal was to see what optimal performance might be achieved. The results for both models were strong as expected as shown in the following table:

	AUC	**F1**	**FNR**	**FPR**
	Train/Test	**Train/Test**	**Train/Test**	**Train/Test**
SVM w/ Original	.949/.921	.940/.909	.042/.061	.047/.076
Final SVM	.963/.949	.956/.941	.035/.042	.033/.049

For an out-of-the-box model, it was remarkably easy to configure and run, and the SVM package even handled the required standardization. The model had impressive performance and nearly tied the two logistic regression models (hand-tailored and automated).

Random Forest Model

The random forest model is also a fairly out-of-the-box model with benefits that include being non-parametric and handling outliers, scalable, and effective. There are two primary tuning parameters: variable selection count and number of trees. The rule-of-thumb is to use the square root of the variable count, but instead a check was run by iterating through the number of variables at 1,000 trees, and the results confirmed the basic rule for this variable set of one hundred variables:



As you can see, the performance has little, if any, improvement for more than ten variables. A check by iterating over the number of trees (plot not shown) also confirmed the use of 1,000 trees. Using those tuning parameters, the random forest models were run for both variable sets with the following results:

	AUC	F1	FNR	FPR
	Train/Test	**Train/Test**	**Train/Test**	**Train/Test**
RF w/ Original	.999/.941	.998/.931	.001/.059	.002/.052
RF w/ Expanded	.999/.952	.999/.945	.001/.043	.001/.044

As one can see from the blue line in the plot above, the random forest model was able to obtain almost 100% predictive capability on the training set. As was to be expected based on the previous three model types, the use of the expanded data set enhanced the performance of the random forest model.

Final Model Comparison

As with many projects, two issues determine the highest level of predictive performance in the deployed model: 1) what modeling technology (assuming they are all equally tuned) is deployed; and 2) what predictive features are available. It is important to note that for purposes of these computations, the default threshold value of >.5 is spam was used. It should be noted that additional tuning may obtain different performance measures if one tunes the threshold to decrease false positives.

To address the first question, the five models using the original data set were reviewed as shown below:

	AUC	F1	FNR	FPR
	Train/Test	**Train/Test**	**Train/Test**	**Train/Test**
Manual Log w/ Original	.951/.952	.870/.870	.079/.073	.106/.102
Automated Log w/Original	.932/.919	.918/.904	.070/.085	.058/.070
Tree w/ Original	.973/.948	.941/.905	.047/.084	.044/.069
SVM w/ Original	.949/.921	.940/.909	.042/.061	.047/.076
RF w/ Original	.999/.941	.998/.931	.001/.059	.002/.052

For this data, the easy winner was the random forest model. On the unseen test data, it was over 2% better on the F1 measure which provides a balanced accuracy measure. That success is seen in the False Negative (FNR) and False Positive (FRP) rates which were the lowest of the models. The training error in latter two rates of the random forest models (.001 and .002, respectively) shows that there was little more that the random forest model could extract from the training data.

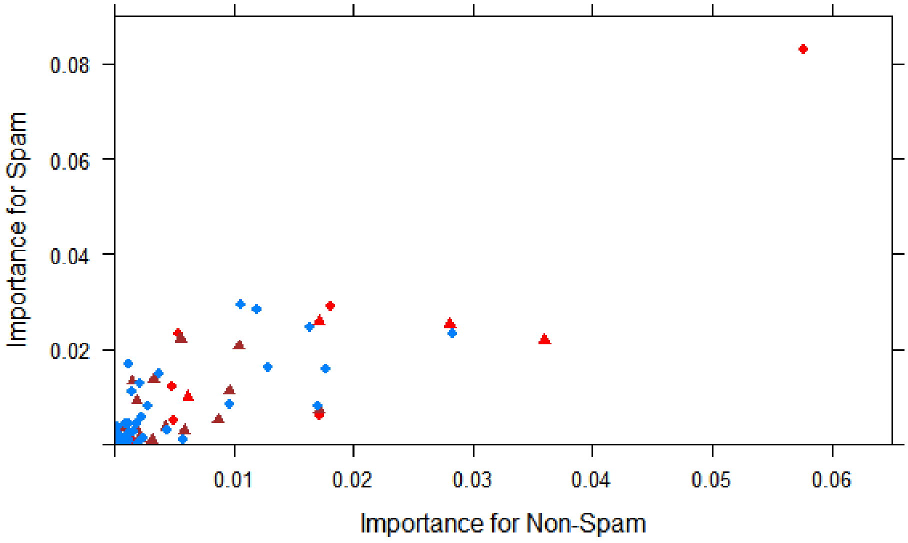
There are many benefits that accrue in selecting a random forest model as well. Many issues related to the other models disappear: 1) no scaling or standardization; 2) no analysis regarding nonlinearities or interactions; 3) no issues with respect to outliers; and 4) no concern about whether parametric model considerations are met. In less technical terms, less work, more output, and less chance that the model will break under stress.

The second question, whether the modeling would benefit from an expanded feature set, is an interesting and important question. Model selection and performance is, in part, driven by the data that is available to the modeler. In the case of just a few frequencies, the random forest model was able to make the most of it. However, when the feature set was expanded, that non-controversial conclusion became somewhat cloudier as shown in the following table:

	AUC	**F1**	**FNR**	**FPR**
	Train/Test	**Train/Test**	**Train/Test**	**Train/Test**
Manual Log w/ Expanded	.989/.984	.942/.935	.055/.060	.039/.046
Automated Log w/ Expanded	.953/.951	.944/.943	.051/.049	.038/.043
Tree w/ Expanded	.949/.935	.927/.913	.063/.079	.053/.063
SVM w/ Expanded	.963/.949	.956/.941	.035/.042	.033/.049
RF w/ Expanded	.999/.952	.999/.945	.001/.043	.001/.044

As the table below shows, all of the models with the expanded feature set performed much better. In particular, the logistic model with standardized data and automated tuning achieved the lowest false positive rate on the test data of any model, and the support vector machine model achieved the lowest rate on the false negative rate. However, the random forest again had the best balanced results, and that is shown again by its having the highest F1 measure and area and total predictive accuracy (AUC).

These additional results support the initial judgment that the random forest model will likely be the best model of the four model types for deployment on this kind of classification problem. Using the random forest model results, the question of the expanded variable set can also be considered. To begin that analysis, the plot of variable importance for both spam and non-spam provides compelling results:



Notes: The blue dots are the original variables, and the brown triangles are nonlinear dummy variables associated with the original variables. The red dots are the novel variables, and the red triangles are nonlinear dummy variables associated with segments of the novel variables.

As you can see, one novel variable (red dot) in particular stands out in the northeast corner of the plot: positivecount. This variable reflects the count sum of positive correlated features that were nonzero, and, as a predictor, it is the most important variable of all in identifying both spam and non-spam emails. Negative count is the next highest novel variable in both directions, and there are numerous red triangles relating to non-linear aspects of these two variables.

The random forest model had already accomplished all that it could with the original feature set, and with additional predictive features, the model reduced the false positive rate by nearly a full percentage point. As the author has noted, the features provided to solve this problem are extremely limited. In the judgment of the author, a false positive error rate of 4% (where one out of every 25 good emails is deemed to be spam) is simply too high for a deployed production system. The author uses a commercial system which he has tuned, and it achieves a false positive rate less than .00001, and a false negative rate of .0003. (If the false negative rate increases where the author receives more than 1 or 2 spam emails in a week, the system quickly gets some tuning adjustment to return it to an acceptable rate!)

A few minutes of brainstorming resulted in a list of other features that could greatly assist in creation of a production model:

- Word counts as well as word frequencies
- Increase number of feature words/symbols
- Include words from subject line as separately identified words
- Total length of email
- Misspelling counter
- Misspellings related to key words (e.g., de@l, fvck, etc.)
- Similarity to honey-pot emails

- Meta-data concerning the email
    - Is there a graphic embedded in the email
    - Header information
        - To, From, CC, BCC
        - Domain Country
        - Originating Mail Server Country Location
        - Character Set Information
        - Rich text or plain text
        - Count of recipients

In addition, a production anti-spam model would probably benefit from some clustering analysis to identify common sub-populations of emails. A multi-class classifier would first classify the email into one or another subpopulation. Categories might include corporate emails, social media, or even subject-matter category like finance or humor. These categories could then be tuned to identify not just a single spam category but multiple spam categories including prosperity opportunities, economic scam (e.g., African prince), mortgage refinance, porn, erectile dysfunction medicine, alternative medicines, chain letters, phishing, and virus infection. The thresholds for false positives would likely vary considerably across these different groups allowing on to tune the total suite of models to minimize the spam while admitting the good email.

Conclusion

Provided an expanded feature set is available, a very good, usable spam classifier could be developed using a random forest modeling technology. With a suite of classifiers, one could probably achieve very high performance levels.

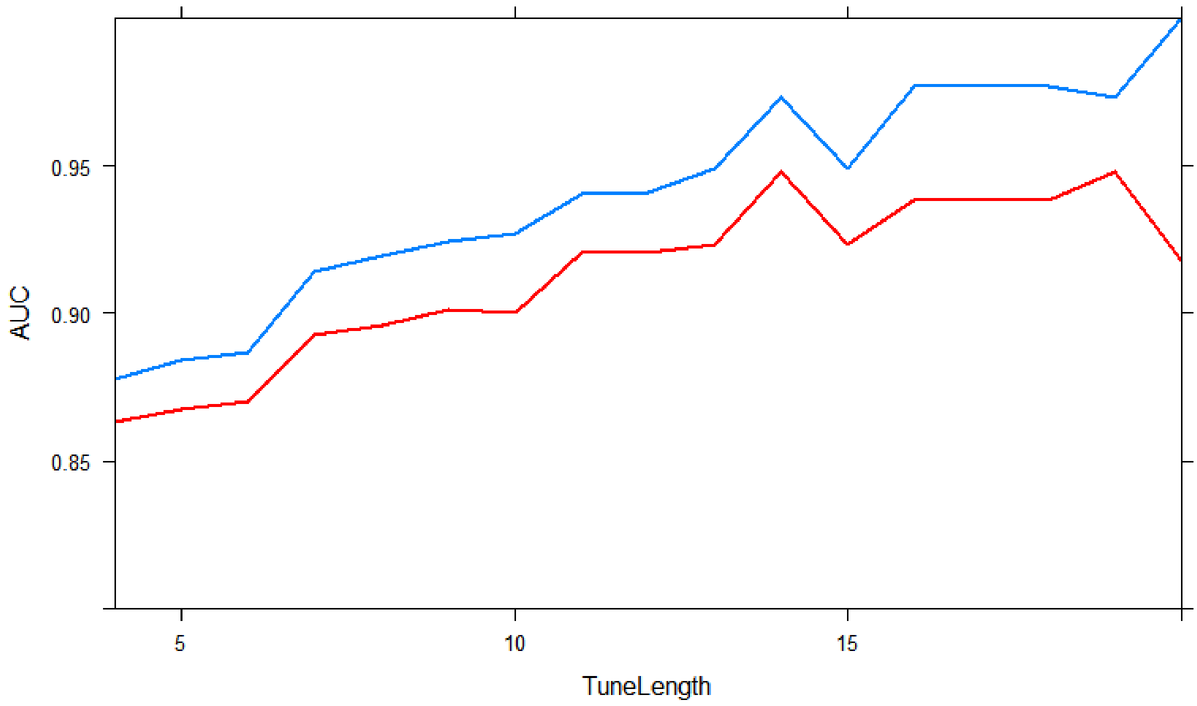
Appendix A

Non-standardized, Manually-constructed Model

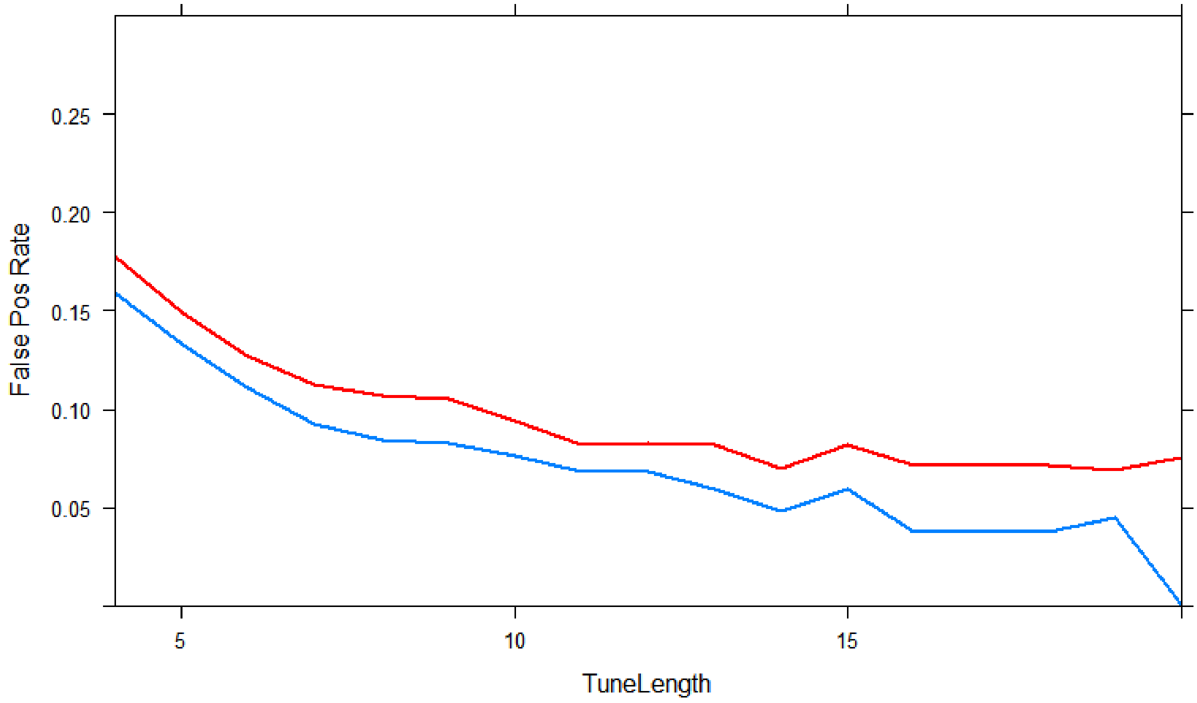
Feature	Estimate	Pr(>|z|)	Signif.	Source
(Intercept)	0.12515	0.9085		
f_make	-0.73906	0.0095	**	as-is
f_all	-0.67242	0.0007	***	as-is
f_our	0.350745	0.0046	**	as-is
f_people	-1.24865	0.0004	***	as-is
f_free	0.411594	0.0000	***	as-is
f_email	-0.31464	0.0413	*	as-is
f_credit	0.360459	0.0613	.	as-is
f_000	0.788718	0.0906	.	as-is
f_money	0.472218	0.0029	**	as-is
f_650	0.874515	0.0009	***	as-is
f_technology	1.38864	0.0011	**	as-is
f_1999	1.093684	0.0007	***	as-is
f_re	-0.58278	0.0174	*	as-is
f_table	-6.87179	0.2065		as-is
cf_semicolon	-3.71872	0.0000	***	as-is
cf_paren	-3.23885	0.0001	***	as-is
cf_bracket	-2.54689	0.0119	*	as-is
cf_exclam	-2.69287	0.0000	***	as-is
cf_pound	-2.74223	0.0050	**	as-is
caps_len_total	0.002039	0.0172	*	as-is
caps_count	-0.0051	0.0152	*	expanded
caps_perc	-2.4033	0.0067	**	expanded
negativecount	-0.70635	0.0000	***	expanded
positivecount	0.560362	<2E-16	***	expanded
percwordaccounted	-0.09109	0.0006	***	expanded
perccharaccounted	2.898968	0.0000	***	expanded
cf_exclamI1	-1.74169	0.0000	***	nonlinear
cf_exclamI2	1.171879	0.0025	**	nonlinear
cf_exclamI3	-0.88053	0.0028	**	nonlinear
f_removeI1	-1.01757	0.0398	*	nonlinear
f_1999I1	1.665962	0.0004	***	nonlinear
f_eduI1	-2.06969	0.0000	***	nonlinear
f_eduI2	-4.03714	0.2344		nonlinear
f_georgeI1	-3.27316	0.0000	***	nonlinear
f_hpI1	-2.77565	0.0000	***	nonlinear
capslenI3	0.557416	0.0247	*	nonlinear
capsavgI1	-2.29368	0.0000	***	nonlinear
capsavgI2	-1.87378	0.0000	***	nonlinear
capsavgI3	-0.67107	0.0230	*	nonlinear
posnegI2	0.498864	0.0862	.	nonlinear
posI1	-1.11211	0.0009	***	nonlinear
negI1	-1.56515	0.0006	***	nonlinear
capspercI1	0.582367	0.0436	*	nonlinear
capslenI4	-0.52086	0.1560		nonlinear
f_order	-0.34929	0.2805		post-step
f_receive	-0.65839	0.1011		post-step
f_remove	0.313434	0.4062		post-step
f_will	-0.07331	0.5255		post-step

APPENDIX B
TREE TRAINING WITH ORIGINAL DATASET

Tree Model Performance for Train (Blue) and Test (Red)



Tree Model False Pos. Performance: Train (Blue) and Test (Red)



APPENDIX C
R CODE

```
# functions
r1.0 = function(myvar){
  returnValue(ifelse(myvar>0,1,0))
}
r.5 = function(myvar){
  returnValue(ifelse(myvar>.5,1,0))
}


scoreAUC <- function(predcol,outcol) {
  perf <- performance(prediction(predcol,outcol==1),'auc')
  as.numeric(perf@y.values)
}

myAUC = function(x,truth){
  returnValue(rcorr.cens(as.numeric(x),as.numeric(truth))[1])
}
myAUC2 = function(x,truth){
  returnValue(rcorr.cens(x,truth)[1])
}

mystandardize = function(dftostandardize,referencedf){
  meanvals <- apply(referencedf, 2, mean)
  stddevs <- apply(referencedf, 2, sd)
  t((t(dftostandardize)-meanvals)/stddevs)
}

FMeasure = function(beta,predictions,truths){
  tbldata = table(predictions,truths)
  TN = tbldata[1,1]
  TP = tbldata[2,2]
  FN = tbldata[2,1]
  FP = tbldata[1,2]
  precision=TP/(TP+FP)
  recall=TP/(TP+FN)
  Fval = (beta*beta + 1)*precision*recall / (beta*beta*precision + recall)
  returnValue(Fval)
}
FPR = function(predictions,truths){
  tbldata = table(predictions,truths)
  TN = tbldata[1,1]
  TP = tbldata[2,2]
  FN = tbldata[2,1]
  FP = tbldata[1,2]
  falseposrate=FP/(FP+TN)
  returnValue(as.numeric(falseposrate))
}
FDR = function(predictions,truths){
  tbldata = table(predictions,truths)
  TN = tbldata[1,1]
  TP = tbldata[2,2]
  FN = tbldata[2,1]
  FP = tbldata[1,2]
  falsediscrate=FP/(FP+TP)
  returnValue(falsediscrate)
}
FNR = function(predictions,truths){
  tbldata = table(predictions,truths)
  TN = tbldata[1,1]
  TP = tbldata[2,2]
  FN = tbldata[2,1]
  FP = tbldata[1,2]
  falsenegrate=FN/(FN+TP)
  returnValue(falsenegrate)
}

####################################################
#################################################### critical added variables
# count up those positive and negative factors
df$negativecount = r1.0(
df$f_hp)+r1.0(df$f_hpl)+r1.0(df$f_george)+r1.0(df$f_1999)+r1.0(df$f_labs)+r1.0(df$f_6
50)+r1.0(df$f_85)+r1.0(df$f_edu)+r1.0(df$f_re)+r1.0(df$f_meeting)+r1.0(df$f_technolog
y)+r1.0(df$f_original)+r1.0(df$f_lab)+r1.0(df$f_telnet)+r1.0(df$f_pm)+r1.0(df$f_data)
+r1.0(df$f_857)+r1.0(df$f_415)+r1.0(df$f_cs)+r1.0(df$f_project)+r1.0(df$cf_paren)+r1.
0(df$f_conference)+r1.0(df$f_direct)+r1.0(df$cf_bracket)+r1.0(df$cf_semicolon)+r1.0(d
f$f_table)
df$positivecount = r1.0(df$f_3d)+r1.0(df$f_report)+r1.0(df$cf_pound)+r1.0(df$f_font
)+r1.0(df$f_make)+r1.0(df$resp_spam)+r1.0(df$f_people)+r1.0(df$f_mail)+r1.0(df$f_cred
it)+r1.0(df$f_addresses)+r1.0(df$f_all)+r1.0(df$f_email)+r1.0(df$f_internet)+r1.0(df$
caps_len_longest)+r1.0(df$f_money)+r1.0(df$f_order)+r1.0(df$f_over)+r1.0(df$f_receive
)+r1.0(df$cf_exclam)+r1.0(df$f_our)+r1.0(df$caps_len_total)+r1.0(df$f_business)+r1.0(
df$f_free)+r1.0(df$f_you)+r1.0(df$cf_dollar)+r1.0(df$f_remove)+r1.0(df$f_000)+r1.0(df
$f_your)+r1.0(df$caps_len_average)
####################################################
####################################################  Box  and density plots
bwplot(f_make~fctr_spam, df, horizontal =FALSE, ylab="Percent", main="Frequency by
Spam: Make",ylim=c(-1, 10),lwd=1,col=c("green","red"))
bwplot(f_our~fctr_spam, df, horizontal =FALSE, ylab="Percent", main="Frequency by
Spam: Our",ylim=c(-1, 10),lwd=1,col=c("green","red"))
bwplot(f_you~fctr_spam, df, horizontal =FALSE, ylab="Percent", main="Frequency by
Spam: You",ylim=c(-1, 10),lwd=1,col=c("green","red"))
bwplot(f_george~fctr_spam, df, horizontal =FALSE, ylab="Percent", main="Frequency by
Spam: George",ylim=c(-1, 10),lwd=1,col=c("green","red"))
bwplot(f_credit~fctr_spam, df, horizontal =FALSE, ylab="Percent", main="Frequency by
Spam: Credit",ylim=c(-1, 10),lwd=1,col=c("green","red"))
bwplot(cf_exclam~fctr_spam, df, horizontal =FALSE, ylab="Percent", main="Freq by
Spam: !",ylim=c(-1, 10),lwd=1,col=c("green","red"))
bwplot(cf_dollar~fctr_spam, df, horizontal =FALSE, ylab="Percent", main="Freq by
Spam: $",ylim=c(-1, 10),lwd=1,col=c("green","red"))
bwplot(caps_len_average~fctr_spam, df, horizontal =FALSE, ylab="Count", main="Count
by Spam: Caps Len Avg",ylim=c(-1, 400),lwd=1,col=c("green","red"))
bwplot(caps_len_longest~fctr_spam, df, horizontal =FALSE, ylab="Count", main="Count
by Spam: Caps Len Longest",lwd=1,ylim=c(-1, 2000),col=c("green","red"))
bwplot(caps_len_total~fctr_spam, df, horizontal =FALSE, ylab="Count", main="Count by
Spam: Caps Len Total",lwd=1,ylim=c(-1, 5000),col=c("green","red"))
```

```
Plotlist =
c("f_make","f_order","f_will","f_you","f_your","cf_paren","caps_len_average","caps_le
n_longest","caps_len_total")
for (i in Plotlist){
  pformula = paste("~",i,sep="")
  LabelText = paste("Variable: ",i,sep="")
  print(densityplot(as.formula(pformula), df, main="Density of Spam - False (Red),
True (Blue)",xlab = LabelText,plot.points=FALSE,groups=resp_spam,
lwd=2,col=c("red","blue")))
}

Plotlist =
c("f_make","f_order","f_will","f_you","f_your","cf_paren","caps_len_average","caps_le
n_longest","caps_len_total")
for (i in Plotlist){
  pformula = paste("~log(",i,")",sep="")
  LabelText = paste("Log(",i,")",sep="")
  print(densityplot(as.formula(pformula), df, main="Density of Spam - False (Red),
True (Blue)",xlab = LabelText,plot.points=FALSE,groups=resp_spam,
lwd=2,col=c("red","blue")))
}
####################################################
####################################################  EDA Tree models

tmodel <- train(fctr_spam~.-resp_spam, data = df.orig, method = "rpart",
                 control=rpart.control(minsplit=2),
                 trControl = trainControl(method = "cv", number = 10),
                 tuneLength=10)
#names(tmodel)
rpart.plot(tmodel$finalModel,cex=.75)
tmodel$results
tmodel$finalModel

tmodel <- train(fctr_spam~.-resp_spam, data = df.orig, method = "rpart",
                control=rpart.control(minsplit=2),
                trControl = trainControl(method = "cv", number = 10),
                tuneLength=20)
#names(tmodel)
rpart.plot(tmodel$finalModel,cex=.5)
tmodel$results
tmodel$finalModel
tmodel$coefnames

####################################################
####################################################  tree models for nonlinearities
#list of vars from tree model
ListOfTreeVars =
c("f_will","f_you","f_your","cf_paren","caps_len_average","cf_dollar","f_remove","cf_
exclam","f_free","f_money","caps_len_average","f_1999","f_our","caps_len_total","f_ed
u","f_remove","f_george","f_hp","positivecount","negativecount","positivecount+negati
vecount","caps_count","caps_perc")
for (i in ListOfTreeVars){
  myformula = paste("resp_spam","~",i,sep="")
model.tree=rpart(as.formula(myformula),df,method="anova",control=rpart.control(minspl
it = 10, cp=.005))
  print(model.tree)
  rpart.plot(model.tree,digits=3,cex=.75)
}
####################################################
####################################################  Models Models Models
#############################################
########LOGISTIC Hand-tuned model
#############################################
df3=df[df$part=="train",]
trainresp= df$resp_spam[df$part=="train"]
df3$part = NULL
df3$resp_spam = NULL
#names(df3)
##### First, step by step: remove all the singularites  shown here.
logmdl = glm(fctr_spam ~ f_make + f_address + f_all
#              + f_george
                +f_cs
                + caps_len_longest
                + f_3d
                + f_hp + f_hpl + f_lab
                + f_telnet + f_415 + f_85
                + f_meeting
                + f_project
                + f_conference
                + cf_dollar
               + caps_len_total
               + caps_len_average
               + f_our + f_over
               + f_remove + f_internet + f_order + f_mail + f_receive + f_will
               + f_people + f_report + f_addresses + f_free + f_business
               + f_email + f_you + f_credit + f_your + f_font + f_000 + f_money
               + f_650 + f_labs + f_data
               + f_technology + f_1999 + f_parts + f_pm + f_direct
               + f_original
               + f_re
               + f_edu + f_table
               + cf_semicolon
               + cf_paren + cf_bracket + cf_exclam
               + cf_pound
               # + caps_count + caps_perc
               # + cf_dollarI1 + cf_exclamI1 + cf_exclamI2 + cf_exclamI3 + f_removeI1
               # + f_ffreeI1 + f_moneyI1 + f_1999I1 + f_ourI1 + f_ourI2 + f_eduI1
               # + f_eduI2 + f_georgeI1 + f_hpI1 + capslenI1 + capslenI2 + capslenI3
               # + capslenI4 + capsavgI1 + capsavgI2 + capsavgI3 + capsavgI4
               # + parenI1 + yourI1 + willI1 + negativecount + positivecount
               # + posnegI1 + posnegI2 + posI1 + posI2 + negI1 + capspercI1 +
capspercI2
               # + percwordaccounted
        #          + capslenavgI1 + capslenavgI2
        #          + perccharaccounted
             ,data=df3,family="binomial")
summary(logmdl)
```

```
#step(logmdl,direction="both",trace=T)

#Resulting Model - Stepwise
logmdl1=glm(formula = fctr_spam ~ f_make + f_all + f_our + 
              f_people + f_free + f_email + f_credit + f_000 + f_money + 
              f_650 + f_technology + f_1999 + f_re + f_table + cf_semicolon + 
              cf_paren + cf_bracket + cf_exclam + cf_pound + caps_count + 
              caps_perc + cf_exclamI1 + cf_exclamI2 + cf_exclamI3 + f_removeI1 + 
              f_1999I1 + f_eduI1 + f_eduI2 + f_georgeI1 + f_hpI1 + capslenI3 + 
              capsavgI1 + capsavgI2 + capsavgI3 +  negativecount +  # yourI1 + 
              positivecount + posnegI2 + posI1 + negI1 + capspercI1 + 
percwordaccounted + 
              perccharaccounted + capslenI4 +caps_len_total
            , family = "binomial", data = df3)
summary(logmdl1)
####################################################
####################################################
#############  Logistic hand-tuned with original variables.
## Original Variables
logmdlorig=glm(formula = fctr_spam~.-resp_spam
               -f_make
               -f_address
               -f_all
               -f_george
               -f_857
               -f_cs
               -caps_len_longest
               -f_3d
               -f_hp
               -f_hpl
               -f_lab
               -f_telnet
               -f_415
               -f_85
               -f_meeting
               -f_project
               -f_conference
               -cf_dollar
               , family = "binomial", data = df.orig.trn)
summary(logmdlorig)
step(logmdlorig,direction="both",trace=T)

logmdl4 = glm(formula = fctr_spam ~ f_our + f_over + f_remove + f_internet + 
                f_order + f_mail + f_will + f_people + f_addresses + f_free + 
                f_business + f_email + f_you + f_credit + f_your + f_font + 
                f_000 + f_money + f_labs + f_data + f_technology + f_1999 + 
                f_parts + f_pm + f_direct + f_original + f_re + f_edu + f_table + 
                cf_semicolon + cf_bracket + cf_exclam + cf_pound + caps_len_average + 
                caps_len_total, family = "binomial", data = df.orig.trn)
summary(logmdl4)

####################################################
####################################################  Examples of metrics
print(scoreAUC(glm.trn.pred01,y[df.std$part=="train"]))
print(scoreAUC(glm.tst.pred01,y[df.std$part=="test"]))
print(FMeasure(1, glm.trn.pred01,y[df.std$part=="train"]))
print(FMeasure(1, glm.tst.pred01,y[df.std$part=="test"]))

print(FPR(glm.trn.pred01,y[df.std$part=="train"]))
print(FPR(glm.tst.pred01,y[df.std$part=="test"]))

print(FNR(glm.trn.pred01,y[df.std$part=="train"]))
print(FNR(glm.tst.pred01,y[df.std$part=="test"]))

#####################################################
#####################################################  Automated Logistic using glmnet
# first use lasso to delete undesirable variables.  In this case, the variables were
deleted iteratively (at the top of the listing, and this code shows no more to
delete.
df.glmnet = df.std.orig
# df.glmnet$f_email =NULL
# df.glmnet$f_lab =NULL
# df.glmnet$f_857 =NULL
# df.glmnet$capslenI2 =NULL
# df.glmnet$capsavgI3 =NULL
# df.glmnet$capslenavgI1  =NULL
# df.glmnet$f_direct  =NULL
# df.glmnet$caps_count  =NULL

df.glmnet$part = NULL
df.glmnet$resp_spam = NULL
df.glmnet$fctr_spam = NULL

x = as.matrix(df.glmnet)
y = as.double(as.matrix(df.std$resp_spam))

grid =10^ seq (10,-10, length =100)
lasso.mod1=glmnet(x[df.std$part=="train",],y[df.std$part=="train"],
family="binomial",alpha=1,lambda=grid)   #lasso plot
plot(lasso.mod1)
lasso.mod=cv.glmnet(x[df.std$part=="train",],y[df.std$part=="train"],family="binomial
", alpha=1,lambda=grid)  #cross validation
plot(lasso.mod)
bestlam =lasso.mod$lambda.min  #.01

lasso.pred=predict(lasso.mod,s=bestlam,newx=x[df.std$part=="train",])
mean((lasso.pred-y[df.std$part=="train"])^2)
out=glmnet(x,y,alpha =1,lambda =grid)
lasso.coef=predict(out,type ="coefficients",s=bestlam )[1:80,]
lasso.coef
lasso.coef[lasso.coef!=0]
lasso.coef[lasso.coef==0]
####
#### Insert dropped variables above as Nulls.  Should now be no more variables to be
dropped.
glm.trn.pred01 =
r.5(predict(lasso.mod,s=bestlam,type="response",newx=x[df.std$part=="train",]))
glm.tst.pred01 =
r.5(predict(lasso.mod,s=bestlam,type="response",newx=x[df.std$part=="test",]))
```

```
table(glm.trn.pred01,y[df.std$part=="train"])
table(glm.tst.pred01,y[df.std$part=="test"])

print(scoreAUC(glm.trn.pred01,y[df.std$part=="train"]))
print(scoreAUC(glm.tst.pred01,y[df.std$part=="test"]))

print(FMeasure(1, glm.trn.pred01,y[df.std$part=="train"]))
print(FMeasure(1, glm.tst.pred01,y[df.std$part=="test"]))

print(FPR(glm.trn.pred01,y[df.std$part=="train"]))
print(FPR(glm.tst.pred01,y[df.std$part=="test"]))

print(FNR(glm.trn.pred01,y[df.std$part=="train"]))
print(FNR(glm.tst.pred01,y[df.std$part=="test"]))
####################################################
####################################################  Tree Tuning for tree model
ctrlist = NULL;
AUCtrnlist = NULL;
AUCtstlist = NULL;
FPRtrnlist = NULL;
FPRtstlist = NULL;
for (i in 4:15){
  tmodel <- train(fctr_spam~.-resp_spam, data=treetrain, method = "rpart",
                  control=rpart.control(minsplit=2),
                  trControl = trainControl(method = "cv", number = 10),
                  tuneLength=i)
  ctrlist = c(ctrlist,i)
  AUCtrnlist =
c(AUCtrnlist,scoreAUC(predict(tmodel,newdata=treetrain,type="prob")[,2],dftrain$resp_
spam))
  AUCtstlist =
c(AUCtstlist,scoreAUC(predict(tmodel,newdata=dftest,type="prob")[,2],dftest$resp_spam
))
  FPRtrnlist =
c(FPRtrnlist,FPR(r.5(predict(tmodel,newdata=treetrain,type="prob")[,2]),dftrain$resp_
spam))
  FPRtstlist =
c(FPRtstlist,FPR(r.5(predict(tmodel,newdata=dftest,type="prob")[,2]),dftest$resp_spam
))
}
Perfdata = data.frame(ctrlist,AUCtrnlist,AUCtstlist,FPRtrnlist,FPRtstlist)

origplot = xyplot(Perfdata$AUCtrnlist~Perfdata$ctrlist,type='a',lwd=2,ylim=c(.60, 1),
                  xlab = "Variable Count",
                  ylab = "RSQ")
newplot = xyplot(Perfdata$AUCtstlist~Perfdata$ctrlist,lwd=2, type='a',
col=c("red"),ylim=c(.8, 1), xlim=c(4, 15),
                 xlab = "TuneLength",
                 ylab = "AUC",main="Tree Model Performance for Train (Blue) and Test
(Red)")

newplot + as.layer(origplot)

origplot = xyplot(Perfdata$FPRtrnlist~Perfdata$ctrlist,type='a',lwd=2,ylim=c(.60, 1),
                  xlab = "Variable Count",
                  ylab = "False Pos Rate")
newplot = xyplot(Perfdata$FPRtstlist~Perfdata$ctrlist,lwd=2, type='a',
col=c("red"),ylim=c(0, .3), xlim=c(4, 15),
                 xlab = "TuneLength",
                 ylab = "False Pos Rate",main="Tree Model False Pos. Performance:
Train (Blue) and Test (Red)")

newplot + as.layer(origplot)

####################################################
####################################################  tree model & metrics
set.seed(1)
tmodel <- train(fctr_spam~.-resp_spam, data=treetrain, method = "rpart",
                control=rpart.control(minsplit=2),
                trControl = trainControl(method = "cv", number = 10),
                tuneLength=10)

tmodel$finalModel
rpart.plot(tmodel$finalModel,cex=.6)
tmodel$bestTune
# cp
# 1 0.003177125


tfit <- rpart(fctr_spam~.-resp_spam, data=treetrain, method="class",
              control = rpart.control(minsplit=2,cp=0.003177125))
rpart.plot(tfit,cex=.6)
names(tfit)
tfit$variable.importance


table(trn_pred=r.5(predict(tmodel,newdata=dftrain,type="prob")[,2]),truth=dftrain$res
p_spam)
table(tst_pred=r.5(predict(tmodel,newdata=dftest,type="prob")[,2]),truth=dftest$resp_
spam)
#AUC
print(scoreAUC(predict(tmodel,newdata=dftrain,type="prob")[,2],dftrain$resp_spam))
print(scoreAUC(predict(tmodel,newdata=dftest,type="prob")[,2],dftest$resp_spam))

print(FMeasure(1,r.5(predict(tmodel,newdata=dftrain,type="prob")[,2]),dftrain$resp_sp
am))
print(FMeasure(1,r.5(predict(tmodel,newdata=dftest,type="prob")[,2]),dftest$resp_spam
))

print(FNR(r.5(predict(tmodel,newdata=dftrain,type="prob")[,2]),dftrain$resp_spam))
print(FNR(r.5(predict(tmodel,newdata=dftest,type="prob")[,2]),dftest$resp_spam))

print(FPR(r.5(predict(tmodel,newdata=dftrain,type="prob")[,2]),dftrain$resp_spam))
print(FPR(r.5(predict(tmodel,newdata=dftest,type="prob")[,2]),dftest$resp_spam))
####################################################
####################################################  Tree model – original data
set.seed(1)
treetrain = df.orig.trn
treetrain$part = NULL
```

```
ctrlist = NULL;
AUCtrnlist = NULL;
AUCtstlist = NULL;
FPRtrnlist = NULL;
FPRtstlist = NULL;
for (i in 4:20){
  tmodel <- train(fctr_spam~.-resp_spam, data=treetrain, method = "rpart",
                  control=rpart.control(minsplit=2),
                  trControl = trainControl(method = "cv", number = 10),
                  tuneLength=i)
  ctrlist = c(ctrlist,i)
  AUCtrnlist =
c(AUCtrnlist,scoreAUC(predict(tmodel,newdata=treetrain,type="prob")[,2],dftrain$resp_
spam))
  AUCtstlist =
c(AUCtstlist,scoreAUC(predict(tmodel,newdata=dftest,type="prob")[,2],dftest$resp_spam
))
  FPRtrnlist =
c(FPRtrnlist,FPR(r.5(predict(tmodel,newdata=treetrain,type="prob")[,2]),dftrain$resp_
spam))
  FPRtstlist =
c(FPRtstlist,FPR(r.5(predict(tmodel,newdata=dftest,type="prob")[,2]),dftest$resp_spam
))
}
Perfdata = data.frame(ctrlist,AUCtrnlist,AUCtstlist,FPRtrnlist,FPRtstlist)

origplot = xyplot(Perfdata$AUCtrnlist~Perfdata$ctrlist,type='a',lwd=2,ylim=c(.60, 1),
                  xlab = "Variable Count",
                  ylab = "RSQ")
newplot = xyplot(Perfdata$AUCtstlist~Perfdata$ctrlist,lwd=2, type='a',
col=c("red"),ylim=c(.8, 1), xlim=c(4, 20),
                 xlab = "TuneLength",
                 ylab = "AUC",main="Tree Model Performance for Train (Blue) and Test
(Red)")

newplot + as.layer(origplot)

origplot = xyplot(Perfdata$FPRtrnlist~Perfdata$ctrlist,type='a',lwd=2,ylim=c(.60, 1),
                  xlab = "Variable Count",
                  ylab = "False Pos Rate")
newplot = xyplot(Perfdata$FPRtstlist~Perfdata$ctrlist,lwd=2, type='a',
col=c("red"),ylim=c(0, .3), xlim=c(4, 20),
                 xlab = "TuneLength",
                 ylab = "False Pos Rate",main="Tree Model False Pos. Performance:
Train (Blue) and Test (Red)")

newplot + as.layer(origplot)

#############
### Final Tree Model for original data
#############
set.seed(1)
tmodel <- train(fctr_spam~.-resp_spam, data=treetrain, method = "rpart",
                control=rpart.control(minsplit=2),
                trControl = trainControl(method = "cv", number = 10),
                tuneLength=16)
tmodel$finalModel
# > tmodel$bestTune
# cp
# 2 0.001853323
tfit <- rpart(fctr_spam~.-resp_spam, data=treetrain, method="class",
              control = rpart.control(minsplit=2,cp =0.001853323))
rpart.plot(fit3,cex=.6)
names(tfit)
tfit$variable.importance
####################################################
####################################################  SVM with metrics
ctrl <- trainControl(method = "cv", number=10)
svmfit <- train(fctr_spam~., data=dftrain, method = "svmRadial", trControl = ctrl)

summary(svmfit$finalModel)
names(svmfit)
svmfit$bestTune
# > svmfit$bestTune
# sigma C
# 3 0.007942672 1
svmfit$coefnames

pred_SVM_trn=as.numeric(predict(svmfit,dftrain))
pred_SVM_tst=as.numeric(predict(svmfit,dftest))

table(predict=pred_SVM_trn, truth=dftrain$fctr_spam)
table(predict=pred_SVM_tst, truth=dftest$fctr_spam)

print(scoreAUC((pred_SVM_trn), dftrain$fctr_spam))
print(scoreAUC(pred_SVM_tst, dftest$fctr_spam))

print(FMeasure(1,pred_SVM_trn, dftrain$fctr_spam))
print(FMeasure(1,pred_SVM_tst, dftest$fctr_spam))

print(FNR(pred_SVM_trn, dftrain$fctr_spam))
print(FNR(pred_SVM_tst, dftest$fctr_spam))

print(FPR(pred_SVM_trn, dftrain$fctr_spam))
print(FPR(pred_SVM_tst, dftest$fctr_spam))

####################################################
####################################################  random forest with metrics
dfrf = dftrain
dfrf$part = NULL
dfrf$resp_spam = NULL

rfmdl=randomForest(fctr_spam~.,data=dfrf,mtry=10,importance=TRUE,ntree=1000)
predict.trn <- as.numeric((predict(rfmdl, newdata=dftrain, se.fit = TRUE)))
predict.tst <- as.numeric(predict(rfmdl, newdata=dftest, se.fit = TRUE))
table(trn_pred=predict.trn,truth=dftrain$resp_spam)
table(tst_pred=predict.tst,truth=dftest$resp_spam)

print(scoreAUC(predict.trn,dftrain$resp_spam))
print(scoreAUC(predict.tst,dftest$resp_spam))
print(FMeasure(1,predict.trn,dftrain$resp_spam))
print(FMeasure(1,predict.tst,dftest$resp_spam))

print(FNR(predict.trn,dftrain$resp_spam))
print(FNR(predict.tst,dftest$resp_spam))

print(FPR(predict.trn,dftrain$resp_spam))
print(FPR(predict.tst,dftest$resp_spam))
```