Fishing for Spam

Sean O’Sullivan

# 1. Summary/Abstract

This study explores the efficacy of advanced machine learning techniques in classifying emails as either spam or legitimate. The process begins with tokenization – converting email text into meaningful tokens. Term Frequency-Inverse Document Frequency (TF-IDF) is then applied to transform these tokens into numerical features that reflect their importance across the data set. To reduce dimensionality and capture the most significant patterns, Singular Value Decomposition (SVD) is then employed, enhancing the efficiency and performance of the subsequent models. Repeated cross-validation is used to ensure robust and reliable model evaluation as three powerful classifiers — Random Forest, Radial Support Vector Machines (SVM), and XGBoost — are trained and compared. The performance of each model is assessed using Recall and F1 scores to ensure their ability to effectively distinguish between spam and legitimate emails while erring on the side of maximizing as many true positives as possible. The results highlight the potential of combining tokenization, TF-IDF, SVD, and repeated cross-validation in building high-performance, black-box email classification systems. Conversely, they demonstrate the difficulty of maintaining human interpretability when data sets are large and dimensionality is high.

# 2. Introduction

## 2.1 Description of the Data

The data set(s) that we’re examining are a handful of annotated data sets pertaining to fraudulent (spam) emails. They come from a handful of sources, the largest of the bunch is from the Conference on Email and Anti-Spam. That data set and a pair of others can be found on [Kaggle](https://www.kaggle.com/datasets/naserabdullahalam/phishing-email-dataset/data?select=CEAS_08.csv) contain text information from the subjects and bodies of the emails as well as the senders, receivers, timestamps, and whether or not URLs are contained within.

## 2.2 Hypotheses to Be Addressed

The purpose of this project is to explore the features that are prevalent in the identification of spam emails. Utilizing text analysis, in addition to the other email attributes supplied and engineered, my end goal is to be able to model a classifier that has a high rate of efficacy in identifying spam emails for use of automated detection by some popular email providers and large internal IT departments across the corporate landscape.

We expect that a number of keywords common in spam emails will be relevant, but it’s also likely that potential sender domains, whether the sender and receiver domain are the same, and the length of the email body or subject also highly correlated with intentionally spammy behavior.

While there may be some models better suited for this than others, our intent is to attempt a handful of ensemble, rule-based classifiers and parameter hyper-tuning in order to compare and determine the best overall predictor of spam emails.

## 2.3 Data Import and Cleaning

To begin we start by downloading our data files of interest from [Kaggle](https://www.kaggle.com/datasets/naserabdullahalam/phishing-email-dataset/data?select=CEAS_08.csv). While there are a handful provided at the prior link we’re mostly interested in the 3 CSV files that contain 6 features in addition to the spam annotation labels. Next we’ll load the data into R with the read\_csv and here functions. Once each of the 3 files is individually loaded into R we append the separate dataframes to one another by way of the dplyr::bind\_rows function and start to summarize our cumulative data so that we can assess what data cleaning steps and feature generation is necessary.

Our first step in cleaning the data centers around data type conversions, namely changing the urls and label variables to be factors rather than character data types. Next, we proceeded with cleaning and extricating sender and receiver emails from their contact names where appropriate. In order to achieve this we utilized the stringr package and regular expressions to isolate the email addresses from the contact and mutate the original columns accordingly. We’ll employee a similar technique shortly to isolate just the domain names for both sender and receiver emails.

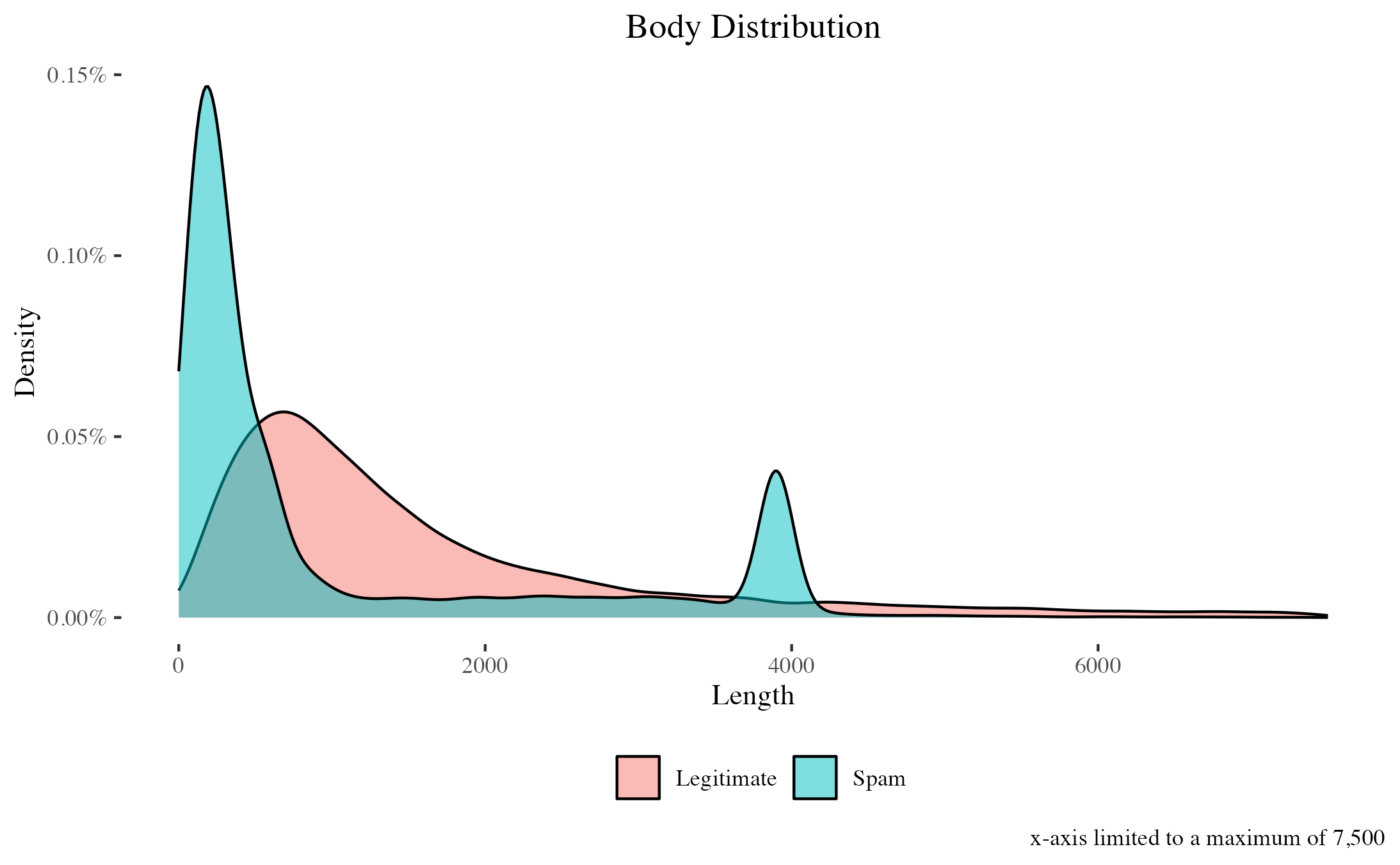
In addition to cleaning some of the existing features we also sought to generate a few simple features that we believe may be of use in our analysis and modeling stages further down the line. Specifically, we created features that measure the character length of both the subject line and email bodies as well as a boolean features that assesses whether or not the sender and receiver email were the same.

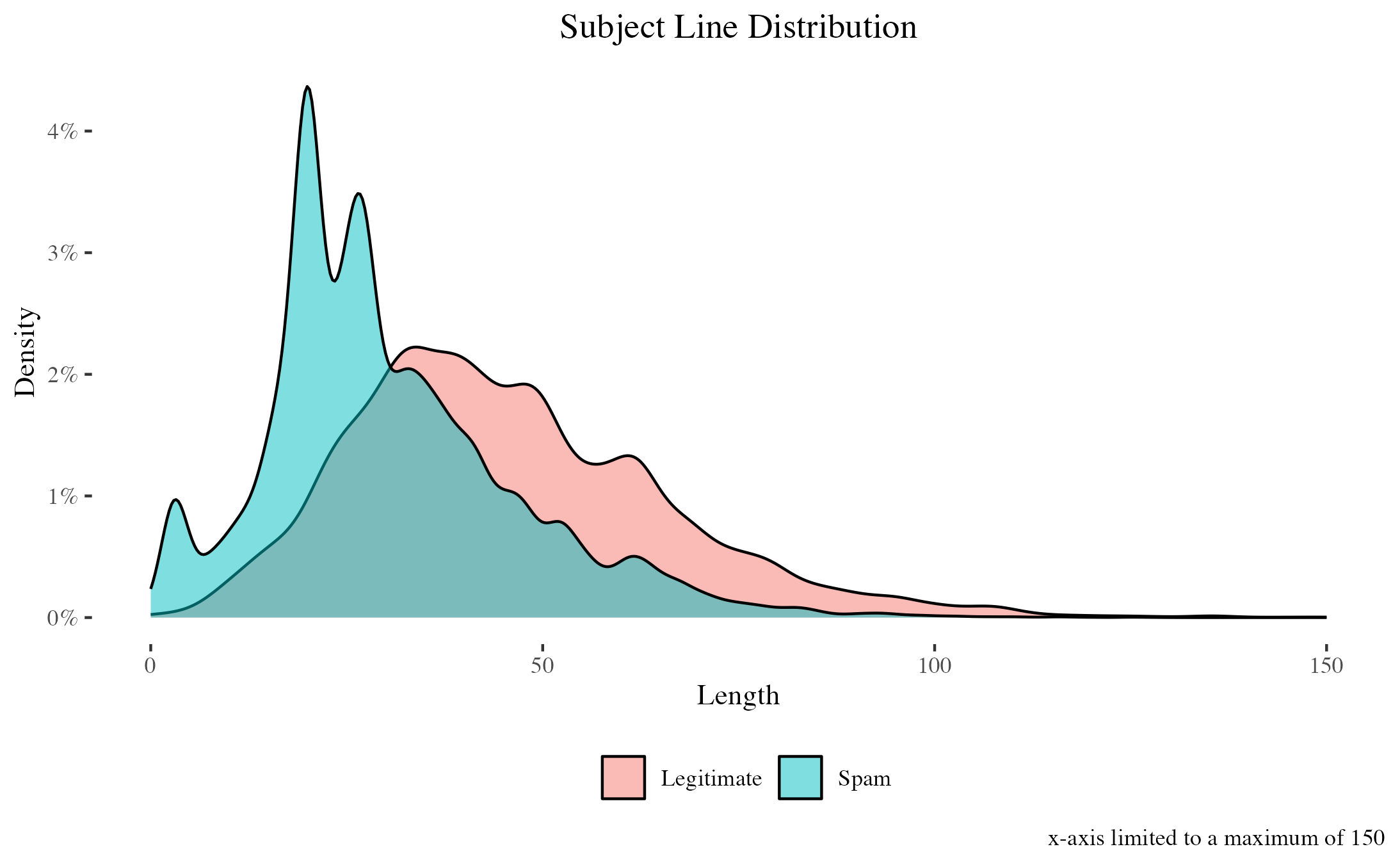
By the end of our data loading and processing we arrived at a dataset that contains 7 character columns, 2 numeric columns, 2 factor columns, and a single boolean column – taking care to ensure that our steps prevented generating any missing values where there hadn’t previously been any. Ultimately this makes for a dataset with 48,295 rows across 11 features and our variable of interest. Complete steps and code can be located in the .../code/processing/processingcode.R file found within the project.

## 2.4 Exploratory Analysis

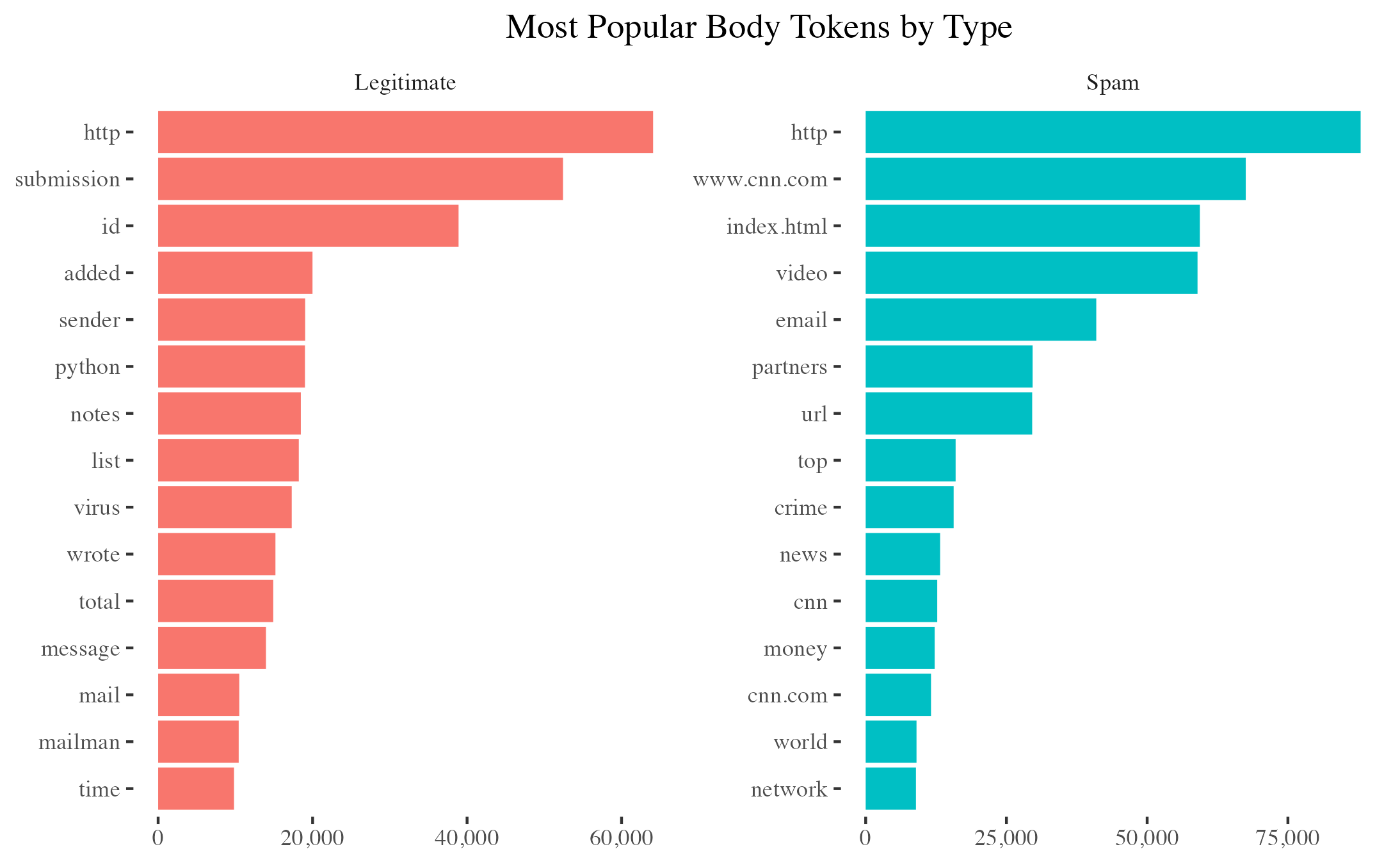
Once the data had been loaded and cleaned we began to examine some of the relationships between anticipated predictor variables and our variable of interest.

First examining the distribution of email body length between spam and legitimate emails we observe a tendency for spam emails to be shorter in length. However, interestingly the distribution of spam emails is bi-modal – with a second peak at around the 4,000 character count – which may indicate that spam vendors are aware of the identification potential of email length and have begun targeting a longer length to avoid notice.

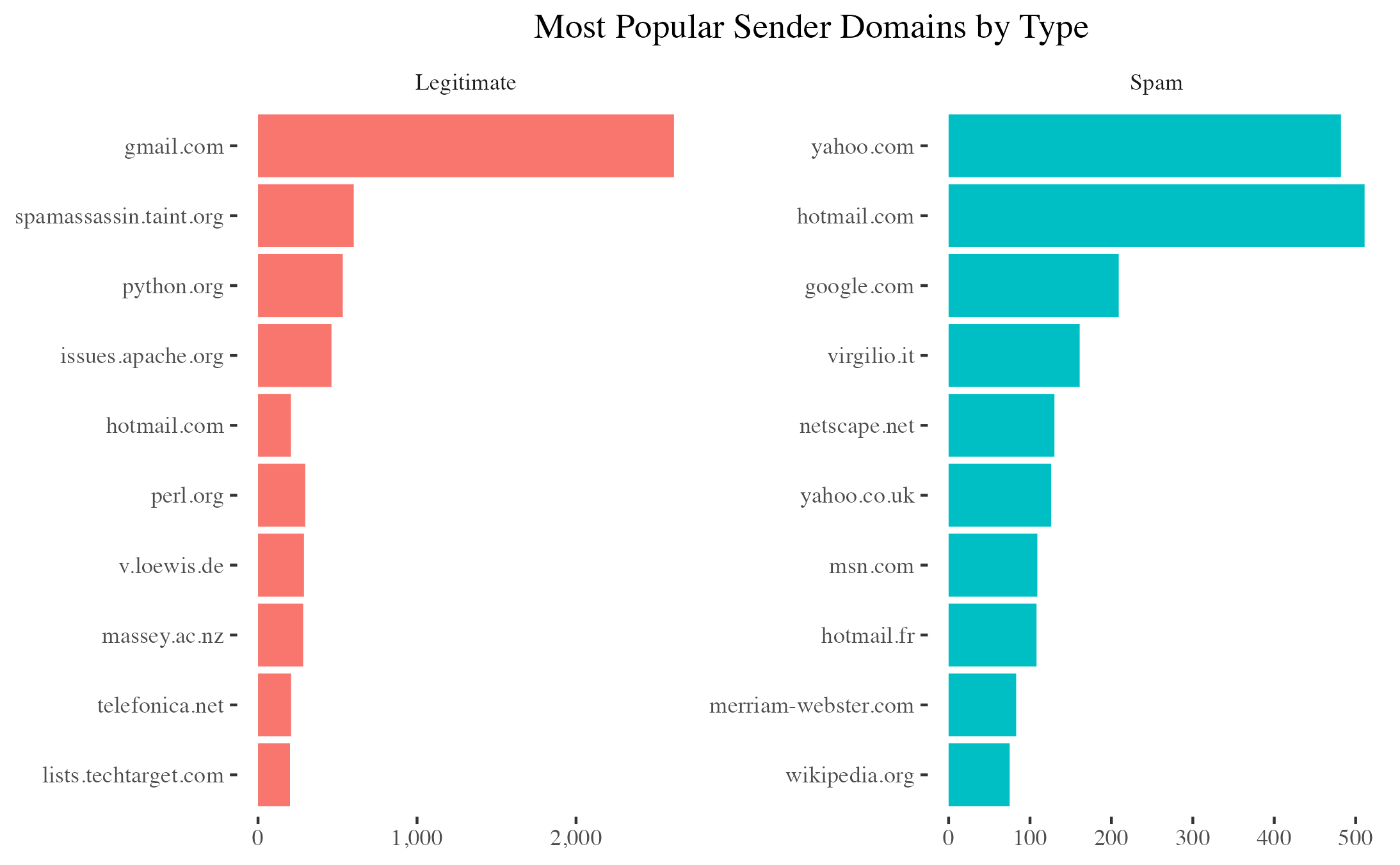


Similarly, when we examine subject line length we observe the same tendency for spam emails to trend shorter. Subject line does not share the same distinct bi-modal shape for spam emails that the bodies do, but subject line length distribution between both spam and legitimate emails overlap in large part which may reduce it’s efficacy for classification modeling. 

When plotting body tokens we can see that ‘http’, which precedes most links within each email, is the most common token for each label. Considering this token should approximate not just the presence of a URL, but also the quantity of URLs within each individual email, it will be interesting to see if including the supplied URLs boolean will be of value or not in our end model. Given the below, we suspect that the overlap will result in unnecessary noise and dimensionality to our final model. Additionally, considering the above bar chart suggests link presence and quantity is very similarly distributed across both labels, it may wind up improving model performance altogether to exclude ‘http’ or URL features from the final model.



Finally, we examined the most popular sender domain names for vendors of both spam and legitimate emails. In doing so we see some clear indications of which domains are favorable to spam vendors and which are not. Notably, Gmail fails to make the top 25 domains for spam vendors despite being the far-and-away most common choice for legitimate emails. Similarly, Yahoo and Hotmail lead the pack for spam and are not among common domains for legitimate emails. This bodes well for the classification power of sender domain when we enter the modeling phase and begin to incorporate bag-of-words and text processing.



A complete set of exploratory visualization steps utilizing ggplot2 can be located in the .../code/eda-code/eda.qmd file found within the project.

## 2.5 Statistical Analysis

### 2.5.1 Preparing the Data

For the purposes of this project our statistical analysis has been completed on a subset of the overall data to allow for reasonable processing times on consumer grade hardware. This extends into some of the choices that were made to exclude especially complex permutations of models whose parameter tuning would take days to complete on such hardware. For comments and notes about the specific choices made throughout the analysis process you can reference the statistical-analysis.qmd in its entirety. The following will be an overview of the data manipulation, feature engineering, dimensionality reduction, and model selection employed within:

We began our analysis by establishing consistent, reusable tuning grids and custom functions for transforming the feature space of our training and testing data. In addition, we also established single-line, custom model training functions to improve legibility as we work through various permutations of both features and model types. We then proceeded to split our data into training and testing splits of 70% and 30% respectively with the latter withheld for model evaluation and to assess any overfitting issues.

Once we had determined our training and testing splits we began to transform our unstructured text data into a structured feature space. This was begun by utilizing the quanteda package to tokenize each email body and subject, forcing all characters of each token to be lowercase, removing tokens for common English stopwords, and stemming each token to it’s English, semantic root so that they could be aggregated on their shared meanings.

With our unstructured text data now structured as simplified tokens we needed to identify which tokens within the overall text corpus are over represented and therefore provided little analytic value. In order to do so, we converted our data to a document-frequency matrix and calculated both the individual term frequencies of each token as well as their inverse document frequencies. The end goal of which was to created a Term Frequency - Inverse Document Frequency (TF-IDF) value for each token that penalizes and discounts those tokens that are over represented in our email corpus.

Having standardized the values for each token by utilizing the TF-IDF, we were left with a structured matrix of tokens with incredibly high dimensionality (>49K tokens) and incredibly high sparsity (>99.9%). While these are common problems when analytically modeling text data, and many machine learning models are robust to data sets where the number of features far exceeds the number of observations, the computational burden for modeling with such high dimensionality is far greater than the typical predictive gains. Additionally, with such high dimensionality and sparsity we run the risk of added predictive noise depending on the models employed.

To address these concerns our next step involved utilizing singular value decomposition to condense our features set to only those 300 singular values most capable of explaining the variation within our data. This process allowed us to both reduce the dimensionality and sparsity of the data while also simultaneously reducing the amount of noise presented by less significant tokens. However, it’s worth noting that this comes at the cost of interpretability.

Having completed this process for our training data we proceeded to project our test data into the same, final feature space so that the models can ultimately be used to predict with the new data.

### 2.5.2 Modeling

With our final structured data prepared we proceeded with tuning 3 different types of classification models with 6 different combinations of features.

The selected models were:

* Random Forest
* Radial Support Vector Machines
* and XGBoost

These models were selected due to their capacity to handle high dimensional feature sets, their robustness to outlier values, and their ability to make accurate predictions about non-linearly related data.

The selected combinations of features utilized with each model were:

* Email Body Tokens Only
* Email Body Tokens with Email Body Length
* Email Subject Tokens Only
* Email Subject Tokens with Email Body Length
* All Email Tokens Only
* and All Email Tokens with Email Body Length

Additional code is also provided that includes permutations with sender email domains. However, due to the high cardinality of that feature and the limited processing power available on consumer grade hardware, the results were not considered in the scope of this project.

# 3. Results

Having completed each model we are now able to assess our model performance and ascertain which model is the most well suited for identification of spam emails. Utilizing the withheld test data we created predictions for the test data with each of our tuned models and generate confusion matrices to evaluate the overall performance of each.

We’ll use this information to help inform which models generalized best when classifying new data and will assess which types of mistakes the models tend to err on the side of – for our purposes allowing the occasional spam message through is likely better than overzealously blocking legitimate emails from reaching the user’s inbox.

In order to determine the model least likely to overzealously block legitimate emails we evaluated the performance of each model on the basis of Recall and F1 score. The former ensures we’re classifying as many true legitimate emails correctly as possible, while the latter assesses the overall accuracy of our classification while factoring in the weights of any class imbalance.

The following table contains the results of each model:

Model Sensitivity Specificity Pos Pred Value Neg Pred Value  
1 rf.body.svd 0.9579439 0.9665428 0.9579439 0.9665428  
2 rf.body.svd.length 0.9595016 0.9702602 0.9625000 0.9678616  
3 rf.sub.svd 0.9034268 0.9541512 0.9400324 0.9254808  
4 rf.sub.svd.length 0.9003115 0.9615861 0.9490969 0.9238095  
5 rf.all.svd 0.9610592 0.9677819 0.9595645 0.9689826  
6 rf.all.svd.length 0.9672897 0.9690211 0.9613003 0.9738481  
7 svm.body.svd 0.9797508 0.9479554 0.9374069 0.9832905  
8 svm.body.svd.length 0.9781931 0.9504337 0.9401198 0.9820743  
9 svm.sub.svd 0.8722741 0.8488228 0.8211144 0.8930900  
10 svm.sub.svd.length 0.8691589 0.9429988 0.9238411 0.9005917  
11 svm.all.svd 0.8785047 0.9653036 0.9527027 0.9089848  
12 svm.all.svd.length 0.8785047 0.9640644 0.9510961 0.9088785  
13 xg.body.svd 0.9719626 0.9677819 0.9600000 0.9774718  
14 xg.body.svd.length 0.9672897 0.9690211 0.9613003 0.9738481  
15 xg.sub.svd 0.9018692 0.9467162 0.9308682 0.9238210  
16 xg.sub.svd.length 0.9158879 0.9553903 0.9423077 0.9345455  
17 xg.all.svd 0.9704050 0.9653036 0.9569892 0.9761905  
18 xg.all.svd.length 0.9672897 0.9702602 0.9627907 0.9738806  
 Precision Recall F1 Prevalence Detection Rate Detection Prevalence  
1 0.9579439 0.9579439 0.9579439 0.4430642 0.4244306 0.4430642  
2 0.9625000 0.9595016 0.9609984 0.4430642 0.4251208 0.4416839  
3 0.9400324 0.9034268 0.9213662 0.4430642 0.4002761 0.4258109  
4 0.9490969 0.9003115 0.9240608 0.4430642 0.3988958 0.4202899  
5 0.9595645 0.9610592 0.9603113 0.4430642 0.4258109 0.4437543  
6 0.9613003 0.9672897 0.9642857 0.4430642 0.4285714 0.4458247  
7 0.9374069 0.9797508 0.9581112 0.4430642 0.4340925 0.4630780  
8 0.9401198 0.9781931 0.9587786 0.4430642 0.4334023 0.4610076  
9 0.8211144 0.8722741 0.8459215 0.4430642 0.3864734 0.4706694  
10 0.9238411 0.8691589 0.8956661 0.4430642 0.3850932 0.4168392  
11 0.9527027 0.8785047 0.9141005 0.4430642 0.3892340 0.4085576  
12 0.9510961 0.8785047 0.9133603 0.4430642 0.3892340 0.4092478  
13 0.9600000 0.9719626 0.9659443 0.4430642 0.4306418 0.4485852  
14 0.9613003 0.9672897 0.9642857 0.4430642 0.4285714 0.4458247  
15 0.9308682 0.9018692 0.9161392 0.4430642 0.3995859 0.4292616  
16 0.9423077 0.9158879 0.9289100 0.4430642 0.4057971 0.4306418  
17 0.9569892 0.9704050 0.9636504 0.4430642 0.4299517 0.4492754  
18 0.9627907 0.9672897 0.9650350 0.4430642 0.4285714 0.4451346  
 Balanced Accuracy  
1 0.9622433  
2 0.9648809  
3 0.9287890  
4 0.9309488  
5 0.9644205  
6 0.9681554  
7 0.9638531  
8 0.9643134  
9 0.8605485  
10 0.9060788  
11 0.9219041  
12 0.9212846  
13 0.9698723  
14 0.9681554  
15 0.9242927  
16 0.9356391  
17 0.9678543  
18 0.9687750

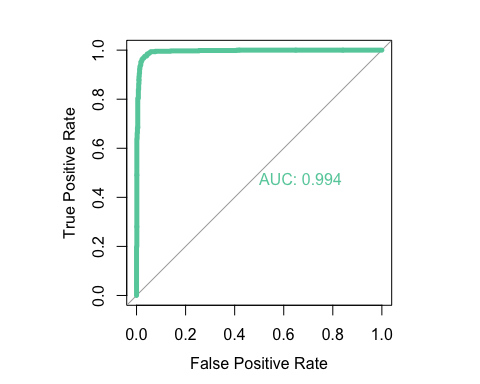
Here we can see that the XGBoost model with only email body tokens seems to achieve the best balance of both F1 score and Recall. The optimal tuning parameters for this model are listed below:

Parameter Value  
1 nrounds 200.0  
2 max\_depth 3.0  
3 eta 0.3  
4 gamma 0.0  
5 colsample\_bytree 0.8  
6 min\_child\_weight 3.0  
7 subsample 0.8

We can see that when examining the confusion matrix for this particular model permutation only 18 false negatives occurred and equally impressively only 26 false positives occurred. This is the exact side of classification error that we would hope to land on with more emails being allowed through than necessarily ought to be, but with few legitimate emails being caught in the model.

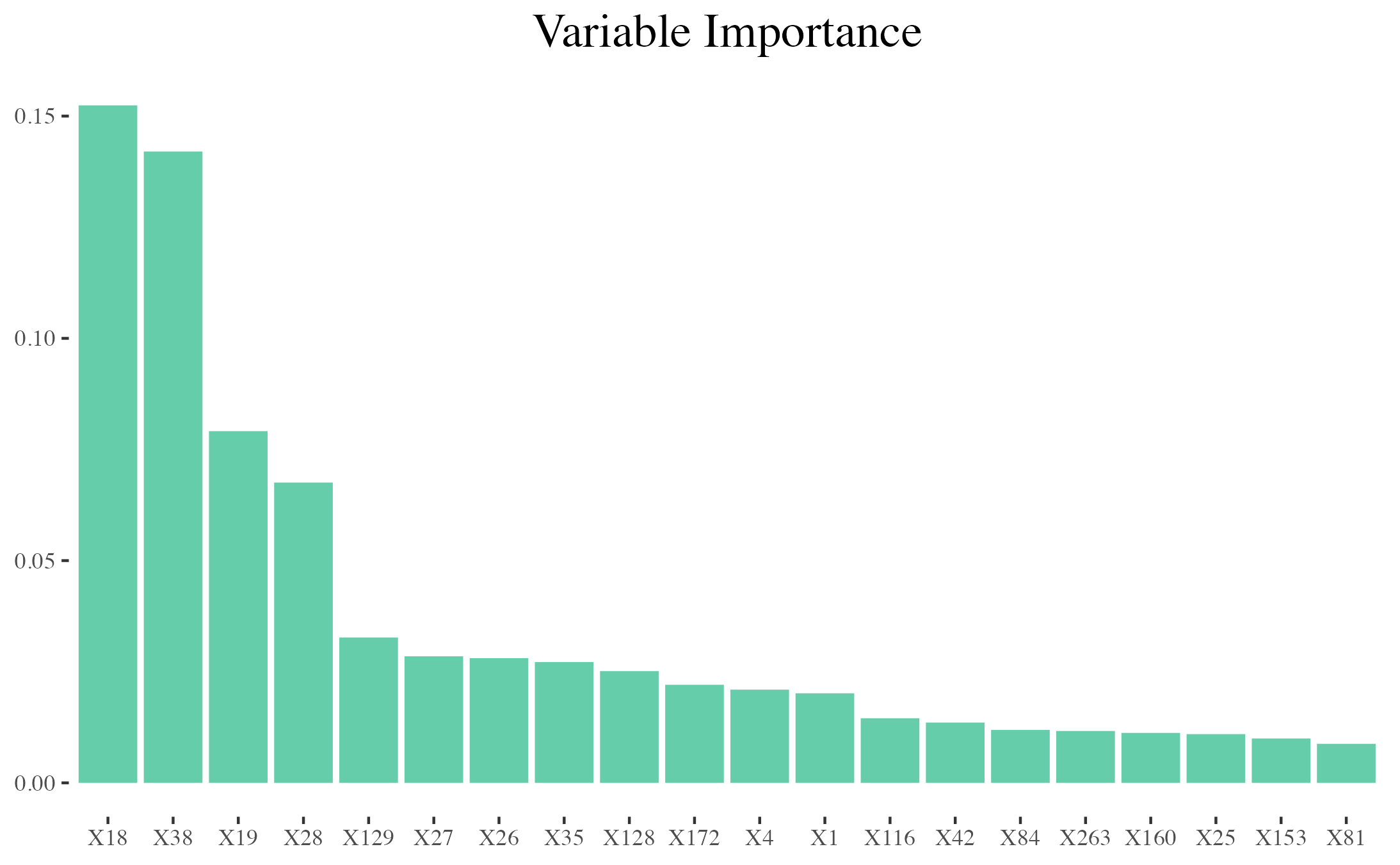
Reference  
Prediction Legitimate Spam  
 Legitimate 624 26  
 Spam 18 781

This high degree of accuracy can also be witnessed when plotting the ROC curve and deriving it’s associated AUC value. With an exceptionally tight fit with the top-left most corner of the graph, the ROC curve for our best model all but maximizes the AUC value with a value of 99.4%.



Lastly we can begin to peer into the workings of the model by examining the variable importance plot. However, we begin to see the pitfalls of SVD on model interpretability. While we there are sharp declines in importance after the first two singular values and the second two singular values we’re unable to ascertain what actual word choices in the body of the email these singular values are comprised of. However, considering the swift decline in importance it may be worth attempting to remodel using this same data without SVD on more powerful hardware so that better intrepretability might be possible.

We do note that none of the highest performing models utilized either email subject tokens or email body length to achieve their scores. It’s possible that the former only worked to introduce noise to the classification models and the latter was partially encompassed in the analysis of the tokens that comprise the body’s length.



# 4. Conclusions

Many of the limitations of this project were those that are inherent in text analytics and highly dimensional predictive modeling. With a very feature rich structured data frame the computational load is quite high. In total the modeling sessions utilizing the repeated cross-validation methodology we employed took a little over 3 hours to run through the permutations that contained the above mentioned features. There are additional features to explore in future iterations should the necessarily powerful hardware be available such as the inclusion of sender domain in addition to timestamp features like time of day, day of week, and month of year factors.

Additionally, the use of singular value decomposition allowed for fairly complex modeling to take place on modest hardware, but it significantly hurt the interpretability of the model. While this may work as a black-box solution if rolled out into production, understanding what tokens are most important could be useful to better tune the model in the future by revising and customizing some elements as stopword lists.

In spite of these limitations, once the models were allowed to run we were able to produce quite sophisticated classification results with few legitimate emails being labeled as spam by our best model within our test set.

# 5. References

Langer, D. (2017). *Introduction to Text Analytics with R*. Retrieved from <https://www.youtube.com/watch?v=4vuw0AsHeGw&list=PL8eNk_zTBST8olxIRFoo0YeXxEOkYdoxi>

Silge, J., & Robinson, D. (2017). *Text mining with R: A tidy approach* (First edition). Beijing ; Boston: O’Reilly.