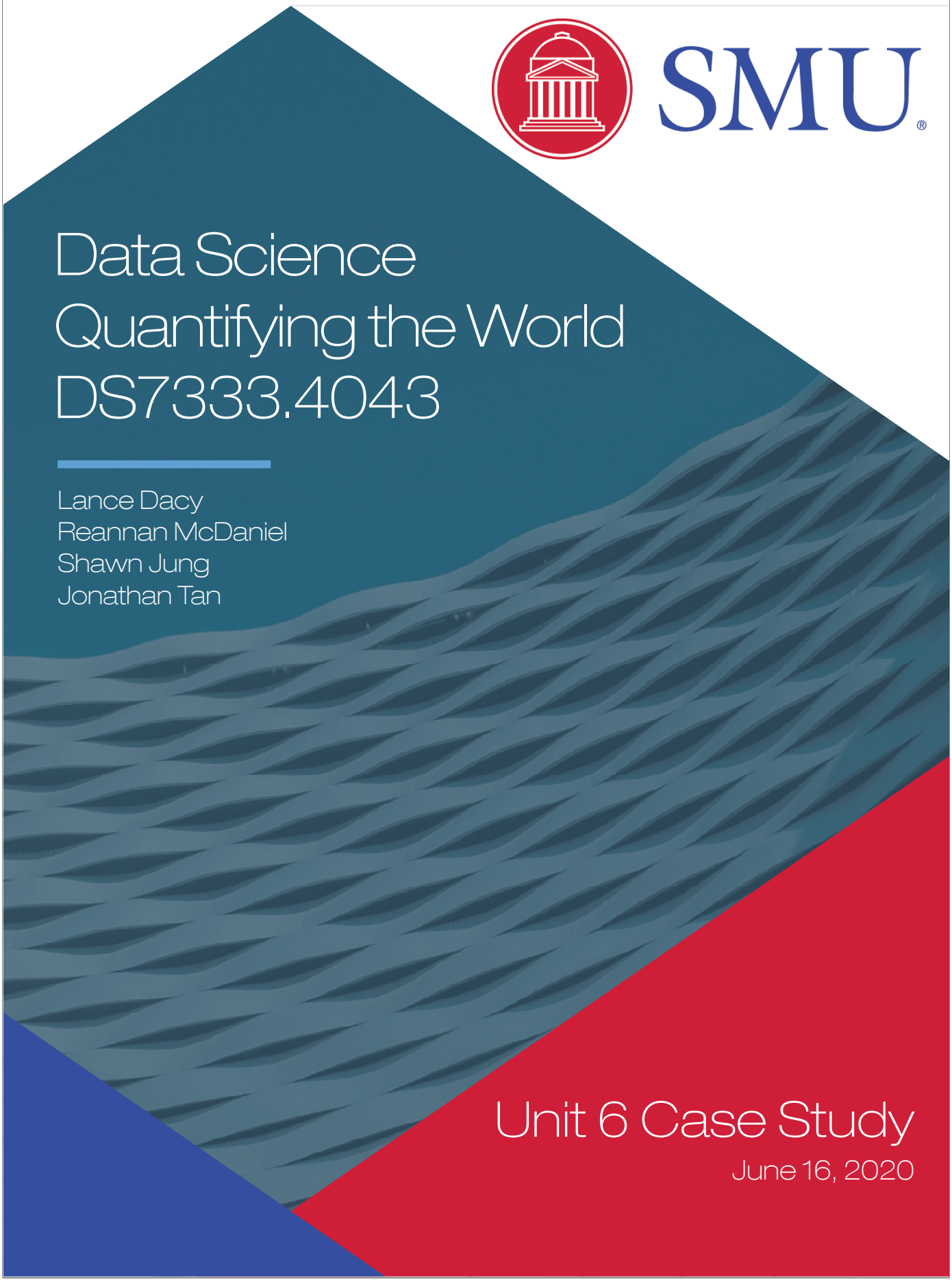


SMU

Data Science Quantifying the World DS7333.4043

Lance Dacy
Reannan McDaniel
Shawn Jung
Jonathan Tan

Unit 6 Case Study

June 16, 2020

Table of Contents

Introduction

Technology today provides near real-time information as long as you are connected to the Internet and are willing to manage the inbound information. Determining the channels and securing a repeatable method to classify and store the information is critical.

Emails have become a normal part of the technology world with nearly 3.9 billion active email users; that is half the global population. Marketers have learned capitalized on the use of these email addresses to ensure you are aware of the next deal or sells that are going on in the world.

The very first version of what would become known as email was invented in 1965 at Massachusetts Institute of Technology (MIT) as part of the university's Compatible Time-Sharing System, which allowed users to share files and messages on a central disk, logging in from remote terminals.

American computer programmer Ray Tomlinson arguably conceived the method of sending email between different computers across the forerunner to the internet, Arpanet, at the US Defense Advanced Research Projects Agency (Darpa), introducing the "@" sign to allow messages to be targeted at certain users on certain machines. The first email standard was proposed in 1973 at Darpa and finalized within Arpanet in 1977, including common things such as the to and from fields, and the ability to forward emails to others who were not initially a recipient.

Fast forward to the world today and we are literally drowning in what is known as SPAM. According to Wikipedia, SPAM was a word introduced as a result of a Monty Python sketch:

"Email spam, also referred to as junk email, is unsolicited messages sent in bulk by email (spamming). The name comes from Spam luncheon meat by way of a Monty Python sketch in which Spam is ubiquitous, unavoidable, and repetitive. Email spam has steadily grown since the early 1990s, and by 2014 was estimated to account for around 90% of total email traffic.

Since the expense of the spam is borne mostly by the recipient, it is effectively postage due advertising. This makes it an excellent example of a negative externality."

SPAM can exist in many forms, but email is typically the first reference. Tools today have been able to mitigate much of the frustrations with the use of predicting which emails are SPAM. Somewhat like a Gandalph for email "This email shall not pass". It ill be interesting to see how the next medium of SPAM is tackled; cell phone robo calls.

In this research; Machine Learning: Random Forests will be used to predict which emails would be classified as SPAM using real data.

Method

Random Forests is a supervised learning algorithm that can be used both for classification or/ and regression. While a real forest is comprised of trees; as is a machine learning forest comprised of decision trees. It is said that the more trees a forest has, the more robust a forest can perform. Random forests creates decision trees on randomly selected data samples, providing a prediction from each tree. It then selects the best solution by means of "voting".

For the purposes of this data set, the Random Forest Classifier will be used to predict whether an email is SPAM or not. The final decision will be the result of a set of decision trees from randomly selected subset of training set. It will then aggregate the votes from different decision trees to decide the final class of the test object.

The goal is for the classier to provide an interpretable or explainable model such as feature importance while also providing a solid training and testing performance standard. Considering the imbalanced nature of the target variable, the decision was made to use an F1 Score as a primary metric to balance between sensitivity and specificity while using accuracy and AUC as a secondary metric.

The F1 Score is the harmonic mean of precision and recall. When the F1 Score reaches its best value of 1; it assumed a perfect precision and recall. In addition, the following metrics will be used as secondary measures:

- **Accuracy** is the proportion of true results among the total number of cases examined. While this metric is chosen as a way to evaluate the model, it will likely be less valuable in our business case given that we could predict that a customer will not churn and likely be accurate.
- **Sensitivity** will measure the model's ability to measure the proportion of actual positivities that are correctly identifi ed. This metric will likely be more valuable to the business case; it is more valuable to predict correctly when an asteroid will hit Earth or correctly predict that a patient will have cancer than to simply say a person has cancer and be wrong. The model success in this instance will be that it can predict when a customer will churn.
- **Specificity** will measure the proportion of actual negatives that are correctly identifi ed which is a valuable metric to determine model performance, but not likely as valuable as Sensitivity for the business outcome of the model (which is to help the business know which customers will churn).
- **Area Under the Curve (AUC)** will measure the performance of the classifi cation problem at various thresholds. This metric will also be signifi cant as the model is tuned based on the business problem of classifi cation and the way that the customers have been placed in bins depending on their length of contract. The closer the model can be tuned toward the number 1, the more likely it has a good measure of separability.

Step 1: Build and Evaluate a Tree-Based Model

Using the raw data (described in the Data section), a training and testing data set was created at an 80/20 split for cross-validation, predictions, and accuracy. The regressor selected is a binary function and has been converted into a classification variable by using a scoring classification with the building of the model. To ensure the purest level is optimized as well as achieving the best split; the following functions were utilized:

- max_depth of 10
- min_sample_leaf of 2
- GINI

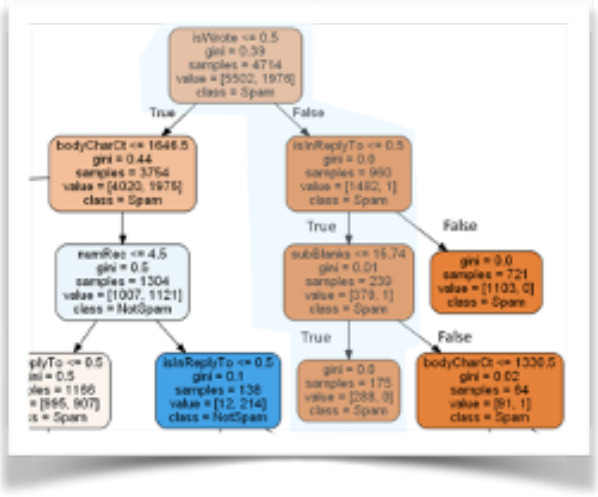
The final score using the following techniques was 89.57%. Adopting the the F1 Score for performance allowed for the balancing of reducing false negatives and false positives. The end result displayed a good F1 Score as well as a higher accuracy score.

Step 2: Plot and Analyze the Paths

Upon inspecting the trees, a path was selected that identifies SPAM. The path selected contained an email sent that was absent of any previous conversation. In addition, it includes sentences such as: 'As a price of distance country, I wrote this letter to get your help to pull my money from swiss bank'.

- Step1 – isWrote <=0.5
  - False (A message includes 'wrote')
- Step2 – isInReplyTo <= 0.5
  - True (A message header does not contains 'In-Reply-To' keyword)
- Step3 – subBlanks <= 15.74
  - True (A message subject has blanks character lower than 15.74%)

Figure 1: Tree Inspection Path



Step 3: Explain the Parameters for Tuning the Model

The following parameters were used to assist in tuning the model and were used in the grid search method:

- Criterion: GINI & Entropy (to decide which criterion provides for good discrimination on target variables)
- Max depth: from 4 to 10 (to decide optimal tree depth that provides good performance while avoiding over fitting)
- Minimal samples per leaf: 2 to 10 (to avoid overfitting)

Iterating through the options above, 5-fold cross-validation was used to evaluate the predictive qualities of the hyper-parameter combination by partitioning the original data into a training and test data set. The 'validation' dataset was set a side for independent performance evaluation.

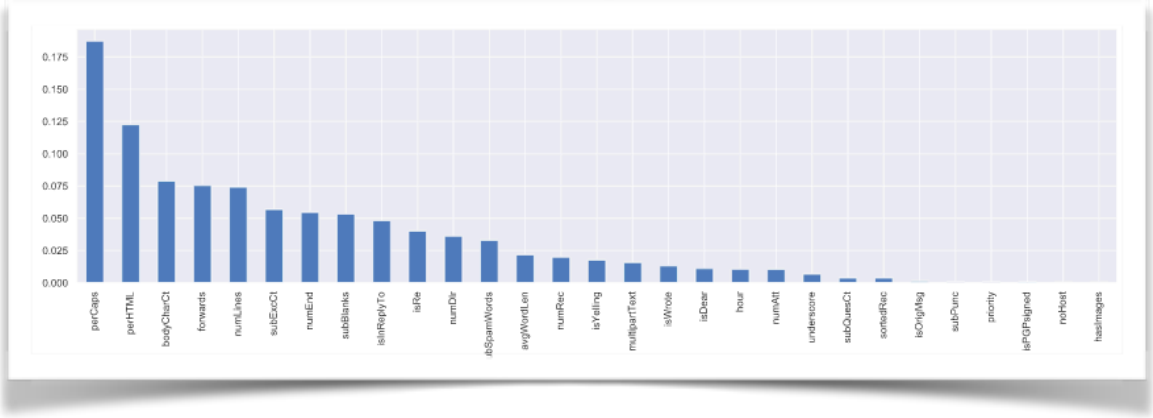
Step 4: Deciding Which Variables are Most Important

Upon inspecting the trees and determining the variables that are most important, the following were chosen to represent the final model:

- perCaps (percentage of capital characters)
- perHTML (percentage of characters in HTML tags)
- numLines (number of lines in a message)

The conclusion is supported by manually inspecting some of the data assigned to this path; "CHEAP RAYBAN SUNGLASSES" emails with HTML-Laden message bodies signifies SPAM from our own experience with receiving SPAM.

Figure 2: Feature Importance Plot



Step 5: Evaluating the Performance of the Model

In comparing the grid search results with F1-Score, it was discovered that a GINI criterion with maximum tree depth 10 and minimal sample leaf 2 showed the best F1-Score of 0.90. Applying these parameters to a split validation data, balanced result was achieved.

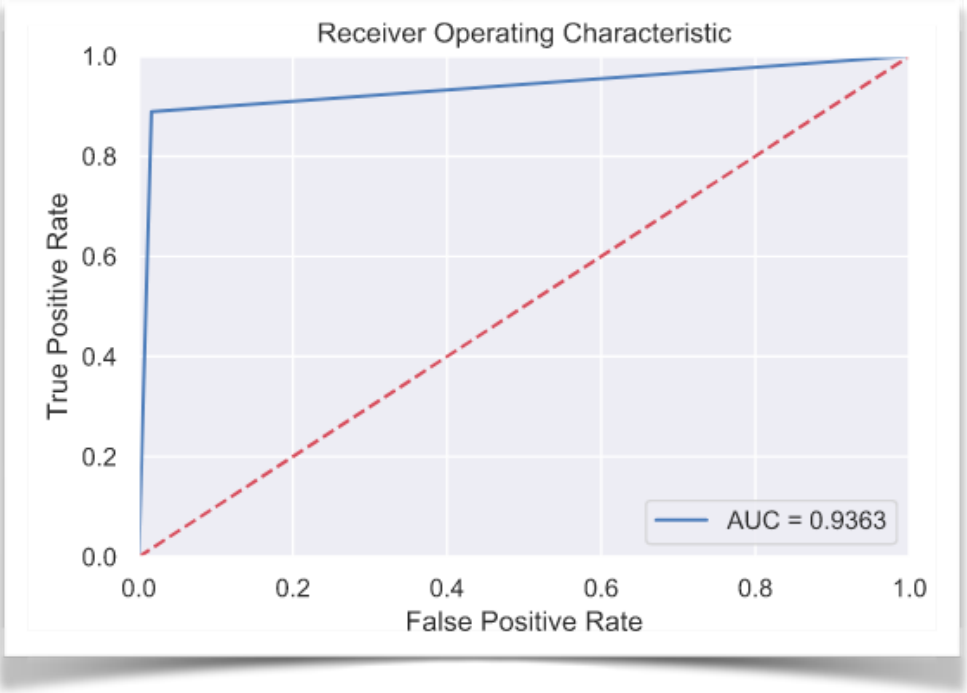
Figure 3: Tuning Metrics Summary

	Precision	Recall	F1-Score
Not-Spam (0)	0.97	0.98	0.97
Spam (1)	0.95	0.90	0.92
Accuracy (Weighted Avg)	0.96	0.96	0.96

- Specificity: .98
- Sensitivity: .90
- ROC AUC: .94

The final interpretation of the performance was resting on the metrics above given their operating values achieving a range that was above satisfactory for prediction.

Figure 4: Receiver Operating Graph (AUC)



Data

Data Source

The data is generated into a raw text file in which RdaReader in Python was used to read in the dataset called "emailDFrp." The actual data is made up of a corpus of emails from SpamAssassin.org. In total, there are 9,348 unique emails which contain 29 predictor variables; including one response variable named isSpam.

Of the 30 total variables, 17 are boolean factor variables and the remaining 13 variables are numeric variables. Each email has been previously classified as spam or valid.

Figure 5: Raw Data Frame Information

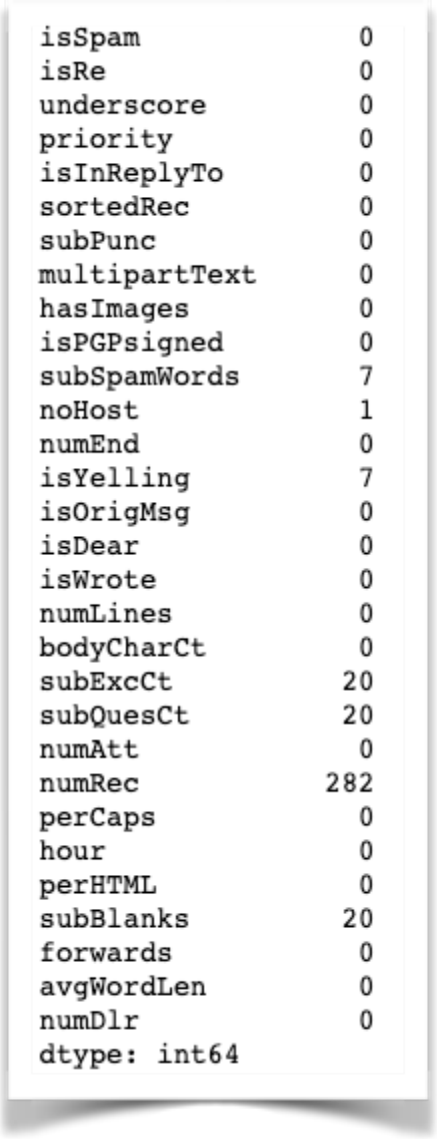
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9348 entries, 0 to 9347
Data columns (total 30 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   isSpam          9348 non-null   category
 1   isRe            9348 non-null   category
 2   underscore      9348 non-null   category
 3   priority        9348 non-null   category
 4   isInReplyTo     9348 non-null   category
 5   sortedRec       9348 non-null   category
 6   subPunc         9348 non-null   category
 7   multipartText   9348 non-null   category
 8   hasImages       9348 non-null   category
 9   isPGPsigned     9348 non-null   category
 10  subSpamWords    9341 non-null   category
 11  noHost          9347 non-null   category
 12  numEnd          9348 non-null   category
 13  isYelling       9341 non-null   category
 14  isOrigMsg       9348 non-null   category
 15  isDear          9348 non-null   category
 16  isWrote         9348 non-null   category
 17  numLines        9348 non-null   int32
 18  bodyCharCt      9348 non-null   int32
 19  subExcCt        9328 non-null   object
 20  subQuesCt       9328 non-null   object
 21  numAtt          9348 non-null   float64
 22  numRec          9066 non-null   object
 23  perCaps         9348 non-null   float64
 24  hour            9348 non-null   float64
 25  perHTML         9348 non-null   float64
 26  subBlanks       9328 non-null   float64
 27  forwards        9348 non-null   float64
 28  avgWordLen      9348 non-null   float64
 29  numDlr          9348 non-null   int32
dtypes: category(17), float64(7), int32(3), object(3)
memory usage: 996.8+ KB
```

Exploratory Data Analysis

Other types of variables in the data set are comprised of metadata or detail for each email. For example, average word length or number of lines in the body of the email. If an attribute contains a "NA" value, it is considered "True" anything else is "False." Within the Spam data set there are 7 attributes that contain "NA." These "NA values" were set to "0."

There are systemic null values with the original data. It was decided to treat them as 'False' or 'Zero'. For instance, there is a feature called 'isYelling' to indicate if an email writer used uppercase letters with email subject. For some of emails, the Subject element itself is missing, so this data will be classified as null; in those cases it was changed to 'False (Not Yelling)'.

Figure 6: Data Quality Check / Replacement



```
isSpam           0
isRe             0
underscore       0
priority         0
isInReplyTo      0
sortedRec        0
subPunc          0
multipartText    0
hasImages        0
isPGPsigned      0
subSpamWords     7
noHost           1
numEnd           0
isYelling        7
isOrigMsg        0
isDear           0
isWrote          0
numLines         0
bodyCharCt       0
subExcCt        20
subQuesCt       20
numAtt           0
numRec         282
perCaps          0
hour             0
perHTML          0
subBlanks       20
forwards         0
avgWordLen       0
numDlr           0
dtype: int64
```

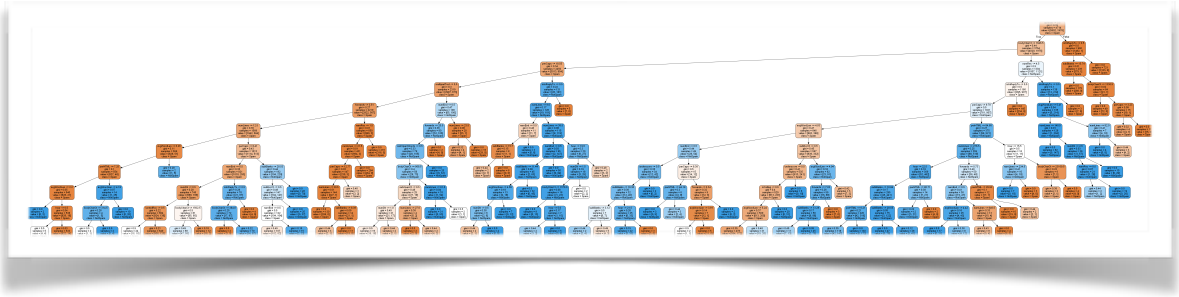
Conclusion

A 2012 article in the Journal of Economic Perspectives cites that up to 3% of the 50 billion pieces of spam e-mail sent each today are successful in reaching the recipient. The authors of the paper develop a cost of $20 billion annually to American consumers from SPAM. As technology increases to determine more effective techniques to reach consumers that don't want to be reached; technology is increasing to combat these techniques in preventing SPAM from the inbox.

While there will likely never be a 100% effective solution to predicting SPAM emails; the technology to assist data scientists is ever increasing. These prediction algorithms can be combined with techniques to allow for categorizing and reviewing from the receivers; significantly reducing the cost of the SPAM.

The graphic below illustrates how much more effective a computer can be at deciding if something is SPAM compared to a human reading through their emails. If the computer can perform at a 90% accuracy; the humans can trust that the SPAM they are having to review is of some consequence and likely be tolerable of the technology short-comings.

Figure 7: Graphical Representation of Random Forest Trees



Appendix

As with any technology; the system required to generate meaningful data, cleaning of the data, as well as the models that were implemented require various software packages. The raw "code" used to explore the methods detailed above can be found in the following libraries:

- GitHub Repository