MSDS 7333 Spring 2021: Case Study 03

Email Spam Detection

Sachin Chavan,Tazeb Abera,Gautam Kapila,Sandesh Ojha

2021 February 19

Introduction

Email spams are unsolicited emails that also referred as Junk emails sent in bulk. There is no law at least in United States that prevents anybody from sending unsolicited emails whether in single email or in bulk. Research indicate that email spams are accounted for over 80% of total email traffic.[2][3][4]. First commercial spam incident was reported in 1994 [5]. That email message was sent to 393 recipients advertising new model of VMS-DEC computer. There are different forms of email spams or unsolicited emails that may affect either individuals or organizations. Following are the few of them :

- Commercial Advertisements
- Hoax emails
- Emails Spoofs or phishing
- Lottery winining notification is very common
- Money Scams
- Virus/Malware
- Fills inbox
- Potentially steals private information
- Privacy concerns.

All these can adversely affect individual as well as organizations. Such emails comes from unknown sources that we never interacted with or sometimes spammers use spoofing techniques to mislead recipients to appear as valid source. Organizations also takes advantage to send bulk emails to potential customers to advertise their products.Such bulk emails occupies network traffic all the time. As far as personal email box goes human can easily categorize such emails and trash them manually. But nowadays there so many spams that it becomes hard to maintain email box without junk. That's where automated spam filters turns out to be very useful. Such a spam filters automatically detects spams and moves them to separate folder.Google for example and many other mailbox providers do run their own spam filters.

Now a days different machine learning algorithm are used for detecting and daily inspecting for incoming emails by different email provider companies like google, yahoo and the like. Among these machine learning algorithms decision tree methods are one of the best list.

Business Understanding

The two common approaches used for filtering spam mails are knowledge engineering and machine learning. Emails are classified as either spam or ham using a set of rules in knowledge engineering [4]. Machine learning approach have proved to be more efficient than knowledge engineering approach. No rule is required to be specified, rather a set of training samples which are pre-classified email messages are provided. A particular machine learning algorithm is then used to learn the classification rules from these email messages.

Machine learning algorithms use statistical models to classify data. In the case of spam detection, a trained machine learning model must be able to determine whether the sequence of words found in an email are closer to those found in spam emails or safe ones.

Several studies have been carried out on machine learning techniques and many of these algorithms are being applied in the field of email spam filtering. Examples of such algorithms include Deep Learning, Naïve Bayes, Decision tree, Support Vector Machines, Neural Networks, K-Nearest Neighbor, Rough sets, and Random Forests.

For this case study we are implementing a decision tree algorithm. Decision trees provides some unique advantages over other ML algorithms like Naive Bayes, SVM, kNN etc. Some of the aspect are expored as a part of current work. The approach uses a decision tree to go from observations associated with the email to target classification of being spam or not a spam. We explore a decision tree package in R called rpart, which is short for recursive partitioning.

Objective

Our objective is to investigate and optimize key hyperparameters used in the rpart package in order to classify email messages as spam or Ham email. In order to accomplish this, we fit a default decision tree and an optimized decision tree and compare them.

Data Evaluation / Engineering

The dataset provided for this cases study was preprocessed and labeled already and contains features extracted from after preprocessing. It contains total of 9348 unique emails with 29 predictor variables and 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. This dataset will be used to build a model using decision trees to classify email messages as spam or ham. Details of each feature is as follows:

Factor variables

Feature	Description
isSpam	Target Variable (T=Spam/F=Ham)
isRe	T=If subject starts with Re: F=Otherwise
underscore	T=If FROM email address field contains underscore F=Otherwise
priority	T=If priority key present in the header F=Otherwise
isInReplyTo	T=If inReplyTo present in the header F=Otherwise
sortedRec	T=If Recipient's email addresses are sorted F=Otherwise
subPunc	T=if Subject line contains punctuation F=Otherwise
multipartText	T=If MIME type is multipart/text F=otherwise
hasImages	T=If email body contains images F=Otherwise
isPGPsigned	T=If message contains PGP signature F=Otherwise
subSpamWords	T=If subject contains one of the words from spam vector F=Otherwise
noHost	T=If there is no hostname in header F=Otherwise
numEnd	T=If sender's email address ends with number
isYelling	T=If subject line contains all uppercase letters F=Otherwise
isOrigMsg	T=If Message body contains word original message F=Otherwise
isDear	T=If word dear found in message body F=Otherwise
isWrote	T=if message body contains word Wrote: F=Otherwise

Numeric variables

Feature	Description
numLines	Number of Lines in the message
bodyCharCt	Number of characters in the messsage
subExcCt	Number of Exclamation in Subject line
subQuesCt	Number of question marks in subject line
numAtt	Number of email attachments
numRec	Number of Recipients in the email message
perCaps	Percentage of uppercase letters in the Body of the message
hour	Hour of the day
perHTML	Percentage of HTML tags in the body of the message
subBlanks	Percentage blanks in Subject line
forwards	Number of consecutive forward symbols in the body of the message
avgWordLen	Average length of the word in the body of the message
numDlr	Number of dollar symbol in the message body

Structure of dataframe

As shown below this dataframe contains 30 columns and 9348 observations.

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

Summary of Factor Variables

```
 isSpam   isRe     underscore priority isInReplyTo sortedRec subPunc
 F:6951   F:6343   F:9222     F:9294   F:6556      F: 948    F:9085
 T:2397   T:3005   T: 126     T:  54   T:2792      T:8400    T: 263

 multipartText hasImages isPGPsigned subSpamWords  noHost     numEnd
 F:9020        F:9326    F:9172      F   :8697    F   :9318   F:8209
 T: 328        T:  22    T: 176      T   : 644    T   :  29   T:1139
                                     NA's:   7    NA's:   1
 isYelling   isOrigMsg isDear   isWrote
 F   :9134   F:8988    F:9270   F:7442
 T   : 207   T: 360    T:  78   T:1906
 NA's:   7
```

Summary of Numeric Variables

```
    numLines        bodyCharCt        subExcCt          subQuesCt      
 Min.   :   2.00   Min.   :     6   Min.   : 0.0000   Min.   : 0.0000  
 1st Qu.:  19.00   1st Qu.:   587   1st Qu.: 0.0000   1st Qu.: 0.0000  
 Median :  32.00   Median :  1088   Median : 0.0000   Median : 0.0000  
 Mean   :  66.91   Mean   :  2844   Mean   : 0.1313   Mean   : 0.1378  
 3rd Qu.:  59.00   3rd Qu.:  2192   3rd Qu.: 0.0000   3rd Qu.: 0.0000  
 Max.   :6319.00   Max.   :188505   Max.   :42.0000   Max.   :12.0000  
                                    NA's   :20        NA's   :20       
     numAtt            numRec            perCaps             hour      
 Min.   : 0.00000   Min.   :  0.000   Min.   :  0.000   Min.   : 0.00  
 1st Qu.: 0.00000   1st Qu.:  1.000   1st Qu.:  4.255   1st Qu.: 8.00  
 Median : 0.00000   Median :  1.000   Median :  6.055   Median :13.00  
 Mean   : 0.06579   Mean   :  1.929   Mean   :  8.850   Mean   :12.21  
 3rd Qu.: 0.00000   3rd Qu.:  1.000   3rd Qu.:  9.059   3rd Qu.:18.00  
 Max.   :18.00000   Max.   :311.000   Max.   :100.000   Max.   :23.00  
                    NA's   :282                                        
    perHTML          subBlanks        forwards       avgWordLen    
 Min.   :  0.000   Min.   : 0.00   Min.   : 0.00   Min.   : 1.363  
 1st Qu.:  0.000   1st Qu.:10.53   1st Qu.: 0.00   1st Qu.: 4.208  
 Median :  0.000   Median :13.25   Median : 0.00   Median : 4.455  
 Mean   :  6.517   Mean   :13.87   Mean   :10.45   Mean   : 4.487  
 3rd Qu.:  0.000   3rd Qu.:15.69   3rd Qu.:15.38   3rd Qu.: 4.729  
 Max.   :100.000   Max.   :86.42   Max.   :99.06   Max.   :26.000  
                   NA's   :20                                      
     numDlr        
 Min.   :   0.000  
 1st Qu.:   0.000  
 Median :   0.000  
 Mean   :   1.782  
 3rd Qu.:   0.000  
 Max.   :1977.000  
```

Summary shows missing values in subSpamWords(7),noHost(1),isYelling(7) from factor variables and in subExcCt(20), subQuesCt(20), numRec(282),subBlanks(20) from numeric fields

Missing Values : Visualization

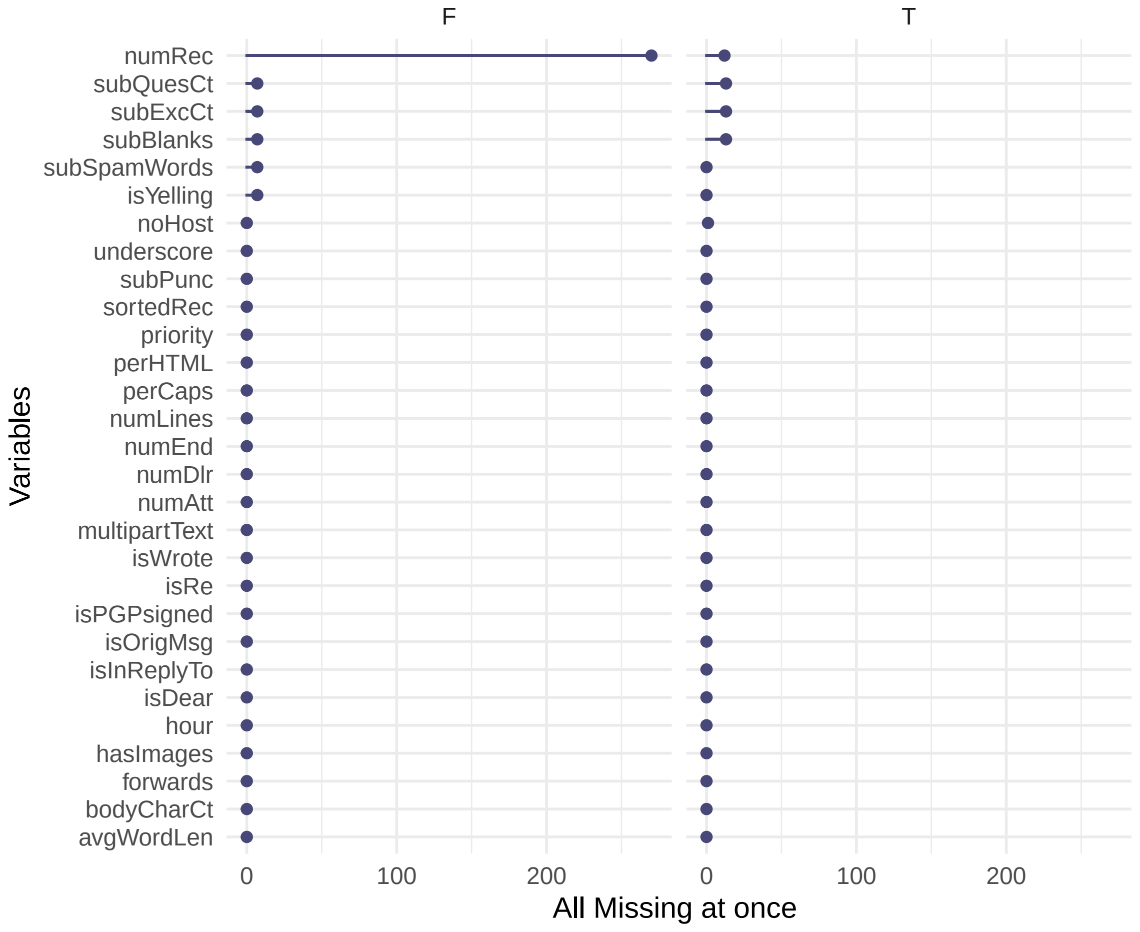


Figure 1: Left : Not Spam, Right : Spam

Interpretation:

- noHost is missing on single record when all other values are present
- subSpamWords & isYelling are missing on same 7 records when all other values are present
- subExcCt, subQuesCt, subBlanks appears to be missing on same 20 records
- It is observed that 13 of 20 rows in subBlanks are NaN.
- numRec has missing values on 282 records most likely due to recipients were added in BCC and only 12 emails have been flagged as Spam and remaining 270 are not spam. These 282 missing values are appears to be random and satisfies MAR (Missing at Random). Missing values will be fixed using mean imputation for both categories (Spam and not Spam) seperately.

Missing Values : Assumptions and Actions

Instead of deleting records with missing values following actions will be taken on the missing data. Imputation will be applied to numRec variable.

FeatureName	DataType	Missing	Assumption	Action
noHost	factor	1	HostName missing	Replace NAs with F
subSpamWords	factor	7	Subject line Missing	Replace NAs with F
isYelling	factor	7	Subject Line Missing	Replace NAs with F
subExcCt	numeric	20	Subject Line without exclamation mark	Replace NAs with 0s
subQuesCt	numeric	20	Subject Line without question mark	Replace NAs with 0s
numRec	numeric	282	Undisclosed recipients	Apply mean imputation to Spam and Not Spam groups seperately
subBlanks	numeric	20	Subject Line Witout Blanks	Replace NAs with 0s

No Missing data in Final DataFrame

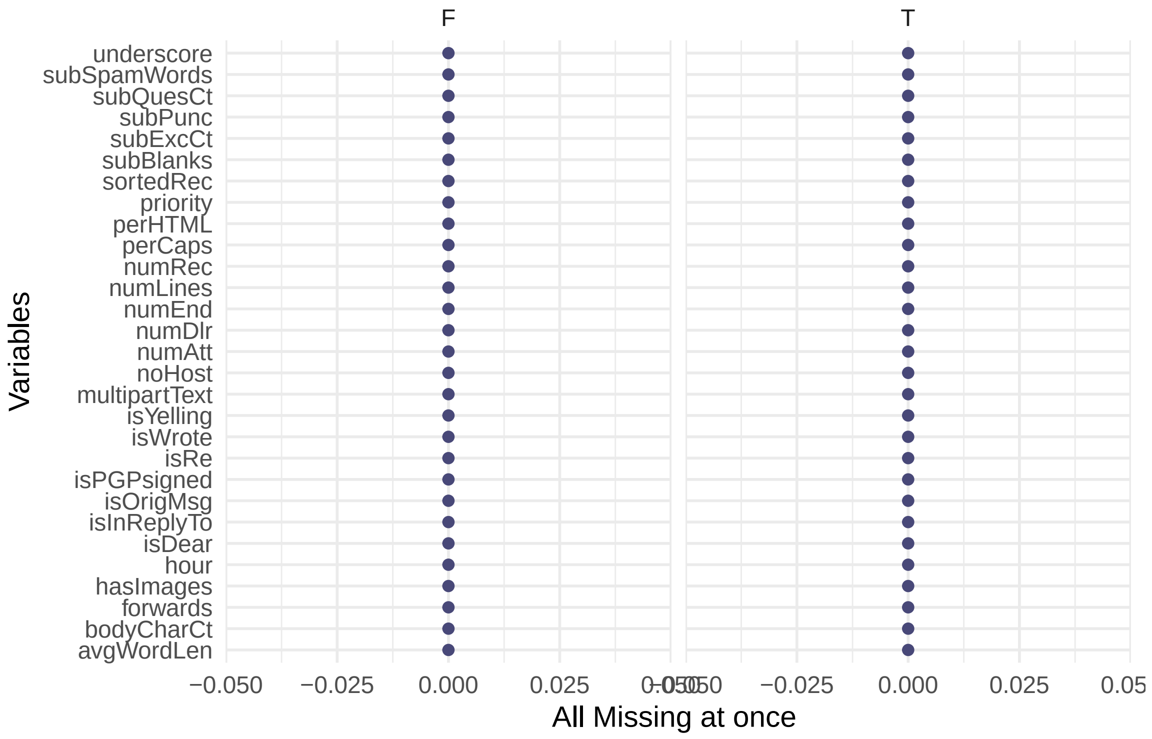


Figure 2: Left : Not Spam, Right : Spam

Class Distribution (Target Variable)

Below chart shows target variable is binary and class lables are not evenly distributed. This is the case of imbalanced dataset.

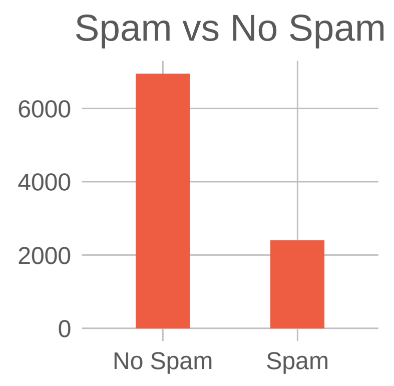


Figure 3: Target Variable

Table 1: Target Percentage Distribution

isSpam	Total	percentage
No Spam	6951	74.35815
Spam	2397	25.64185

Correlation Plot

This plot shows numLines and bodyCharCt are strongly correlated at 90% and one of them can be removed. So bodyCharCt will be removed from further analysis.

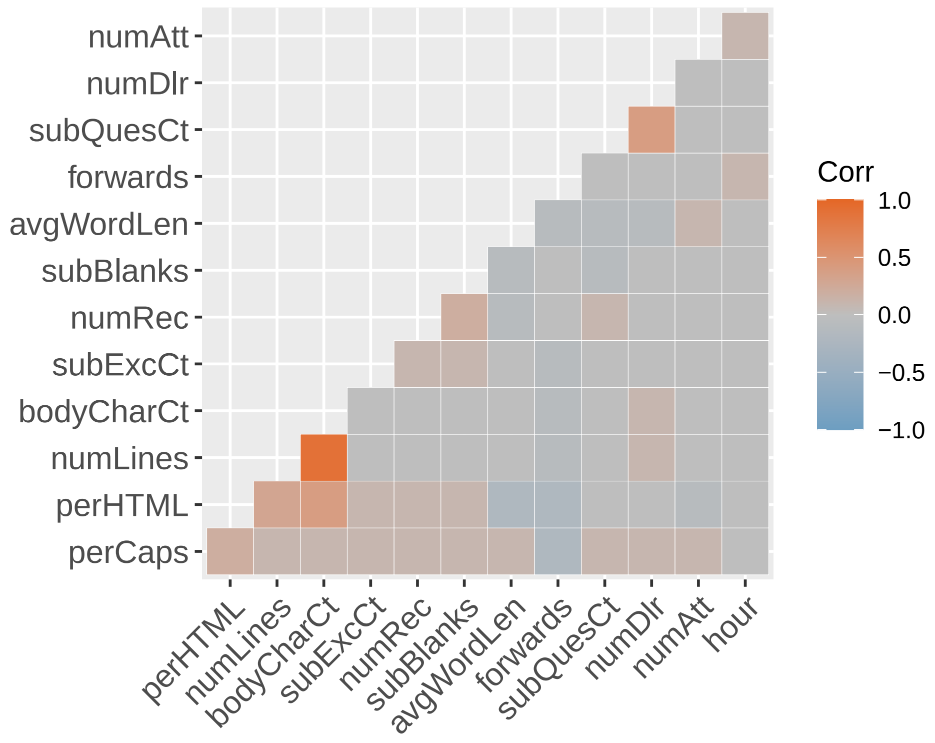


Figure 4: Correlogram

Strcuture of Final DataFrame

```
## 'data.frame':    9348 obs. of  29 variables:
##  $ isSpam       : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isRe         : Factor w/ 2 levels "F","T": 2 1 1 1 2 2 1 2 1 2 ...
##  $ underscore   : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ priority     : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isInReplyTo  : Factor w/ 2 levels "F","T": 2 1 1 1 1 2 1 1 1 2 ...
##  $ sortedRec    : Factor w/ 2 levels "F","T": 2 2 2 2 2 2 2 2 2 2 ...
##  $ subPunc      : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ multipartText: Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ hasImages    : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isPGPsigned  : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ subSpamWords : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ noHost       : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ numEnd       : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isYelling    : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isOrigMsg    : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isDear       : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 1 1 1 ...
##  $ isWrote      : Factor w/ 2 levels "F","T": 1 1 1 1 1 1 1 2 1 1 ...
##  $ numLines     : int  50 26 38 32 31 25 38 39 126 50 ...
##  $ subExcCt     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ subQuesCt    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ numAtt       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ numRec       : num  2 1 1 0 1 1 1 1 1 2 ...
##  $ perCaps      : num  4.45 7.49 7.44 5.09 6.12 ...
##  $ hour         : num  11 11 12 13 13 13 13 14 14 11 ...
##  $ perHTML      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ subBlanks    : num  12.5 8 8 18.9 15.2 ...
##  $ forwards     : num  0 0 0 3.12 6.45 ...
##  $ avgWordLen   : num  4.38 4.56 4.82 4.71 4.23 ...
##  $ numDlr       : int  3 0 0 0 0 0 0 0 0 3 ...
```

Final Dataframe for analysis

Structure of DataFrame has not changed. In previous steps only missing values have been replaced with 0s or Fs based on assumption made. Here is sample DataFrame.

Table 2: DataFrame

isSpam	isRe	underscore	numLines	subExcCt	subQuesCt	numAtt	numRec	perCaps	hour
F	T	F	50	0	0	0	2	4.451039	11
F	F	F	26	0	0	0	1	7.491289	11
F	F	F	38	0	0	0	1	7.436096	12
F	F	F	32	0	0	0	0	5.090909	13
F	T	F	31	0	0	0	1	6.116643	13
F	T	F	25	0	0	0	1	7.625272	13
F	F	F	38	0	0	0	1	6.343714	13

Train/Test split

Original data to be split into train(0.8) and test set (0.2). R package caret provides library function createDataPartition to split data into train/test set and it performs stratified sampling by default as per its documentation. Stratified sampling technique preserves percentage of samples of each class. Target variable in this case is binary i.e. email could either be Spam or not Spam and only approximately 25% emails have been labeled as Spam therefore Stratified sampling is the best way to proceed.

Train - Target Percentage Distribution

createDataPartition function from **caret** packages was used to split data into train/test set.Target variable is binary and distribution of classes is not balanced. i.e. 70% observations are not spam and only 30% are labeled as Spam in this dataset. This is case of imbalanced dataset. Random Stratified sampling has been used to split data into train/test set for building model using decision trees. **Stratified Sampling** preserves the percentage of samples of each class. Scaling and normalization is required for decision tree models.

Table 3: Target Percentage Distribution

isSpam	Total	percentage
No Spam	5561	74.35486
Spam	1918	25.64514

Constraints

Following are limitations of Decision tree algorithms:

- Decision Tree algorithms are greedy algorithms, prone to overfitting.If not handled properly tree tends to grow larger and become complex and difficult to interpret. Overfitting affects generalizability and hence accuracy of prediction goes down significantly.
- Trees are unstable, meaning that even small change in information leads to big change in trees.
- Loses lot of information in splitting process.
- Bad for large datasets

However, there are different ways to tackle these problems. Hyperparameter tunining, building stopping criteria are discussed in next sections.

Modeling Preparations

Approaches:

Two approaches were under consideration for email SPAM classification problem, (1) Naïve Bayes, and (2) decision tree (DT) based approach. These are discussed below and a case is made for use of decision tree for modeling problem. Naïve Bayes (NB) requires or assumes an underlying probabilistic model. This is not a requirement for decision tree. In contrast to NB approach, decision tree algorithms are relatively easier to understand in terms of understanding classification algorithm, and also provides easy identification of important features, and handling of missing values. In DT based approach it is easier to accommodate additional derived features from an email (e.g. number of forwards, number of capital letters in subject line, etc.), and not just the word content. To explore and capture above, DT based approach is pursued in this work.

Classification Metrics:

For the binary classification problem at hand, with goal of predicting if Email is SPAM or NOT, we have following confusion matrix interpretation:

While accuracy is a default metric for better classification, for current problem we need to discuss importance of FP (False Positive) and FN (False Negative) scenarios to identify metrics:

False Positive (FP) rate needs to be minimized.

- It is the number of times Email is classified as SPAM, while actually it was not.
- Goal is that the SPAM filter should not automatically block emails by classifying them as SPAM, when they actually are not. Otherwise, users could end up loosing a lot of important emails.
- This implies that specificity given by $\frac{TN}{TN+FP}$ = ~1, or as high as possible.

False Negative (FN) rate should be minimized, but is less critical than FP rate.

- This is the number of times Email is classified as Ham, while actually it was SPAM.
- FN rate is less critical than FP rate, as we can occasionally allow SPAM messages to reach end user, as long as we get better model, i.e. a very low FP rate.
- It is assumed that that the end person can exercise judgement and categorize such emails as SPAM, and take corrective actions as per company's IT policy.

Above arguments lead us to select following metrics to be considered while choosing the best model:

- Specificity, or True Negative Rate (TNR)
  - Given $TNR = \frac{TN}{TN+FP}$
  - Minimizing FP, implies we need to select model with highest TNR values
- Precision, or Positive Predictive Value (PPV)
  - Given $PPV = \frac{TP}{TP+FP}$
  - Minimizing FP, implies we need to select model with highest PPV values

There is additional flexibility in choosing model with a slightly poorer recall, or sensitivity $\frac{TP}{TP+FN}$ as cost of FP is larger than FN, and focus is minimizing FP over FN.

Lastly, we have an imbalanced classification problem at hand. Refer Figure above, that shows that ~75% outcomes are HAM, while ~25% outcomes are SPAM in modeling dataset. Due to above, and to provide a better balance between precision and recall, F1 metric is additionally chosen as a metric for choosing the best model. F1 is harmonic mean of 'Precision' and 'Recall' and given by

$\frac{2*Precision*Recall}{Precision+Recall}$

F1 score penalizes if any one of precision or recall falls low, which may also happen due to imbalanced classification problem at hand when attempting k-fold cross validation. Basic Tree Model & Parameters

Parameters tuned for our basic tree are mentioned below along with description: minsplit - the minimum number of observations that must exist in a node in order for a split to be attempted minbucket - the minimum number of observations in any terminal node cp - complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. A high cp value implies a split is attempted only if information gain is larger by amount cp. Choosing a higher value of cp reduces number of tree splits, as fewer splits will meet the high bar on information gain as set by high cp parameter.

In our analysis above parameters are varied.

Model Building & Evaluation

Decision tree algorithm are prone to overfit quickly and loses generalizability for prediction i.e. prediction performance become poor. Solution to this problem is hyperparameter tuning. Decision trees have several hyparameters that can be tuned to get optimal model. Below table lists important hyperparameters of which minsplit,minbucket, cp and maxdepth will be used here to tune the model. These are the stopping criterion for the trees which can be applied at each stage during tree-building process.

HyperParameters	Description
minsplit	Minimum number of observations required to split the given node. If it is set to 20 indicates that if data has less than 20 records then root node becomes leaf node.
minbucket	Minimum number of observations required at leaf node. default is minbucket = minsplit/3
cp	Complexity Parameter to control size of the tree and also known as cost of adding another variable to decision tree
maxdepth	Set the maximum depth of any node of the final tree
maxsurrogate	Used if data contains missing values. useful to determine which split can be used for missing data
usesurrogate	Controls use of surrogate split
surrogatestyle	Controls selection of best surrogate

getParamSet(tree) function from **R package mlr** returns following list of hyper-parameters for the decision tree.

```
                   Type len  Def   Constr Req Tunable Trafo
minsplit        integer   -   20 1 to Inf   -    TRUE     -
minbucket       integer   -    - 1 to Inf   -    TRUE     -
cp              numeric   - 0.01   0 to 1   -    TRUE     -
maxcompete      integer   -    4 0 to Inf   -    TRUE     -
maxsurrogate    integer   -    5 0 to Inf   -    TRUE     -
usesurrogate   discrete   -    2    0,1,2   -    TRUE     -
surrogatestyle discrete   -    0      0,1   -    TRUE     -
maxdepth        integer   -   30  1 to 30   -    TRUE     -
xval            integer   -   10 0 to Inf   -   FALSE     -
parms           untyped   -    -        -   -    TRUE     -
```

Tuned Parameters

Parameters are tuned with following criteria

- 5 <= minsplit <=20
- 3 <= minbucket <=10
- 3 <= maxdepth <=10
- 0.01 <= cp <=0.1
- 10 fold cross validation
- Random search Max iterations 200

Here are tuning results:

```
Tune result:
Op. pars: minsplit=8; minbucket=3; cp=0.0125; maxdepth=9
f1.test.mean=0.9379524,tpr.test.mean=0.9496755,ppv.test.mean=0.9266441,tnr.test.mean=0.7823693
```

Impact of hyper parameter tuning on model's performance

Model performance metric variation as a function of tuning parameters is analyzed through plots below.

Impact due to Complexity Parameter (cp)

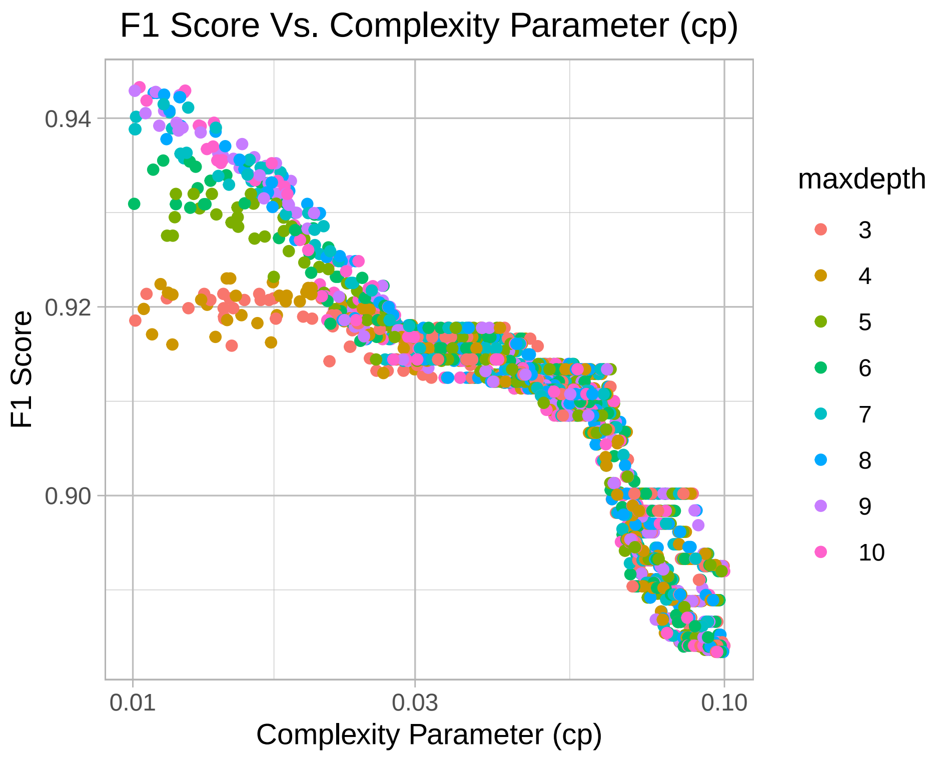
Complexity parameter controls the tree growth, and additional splits at a level are attempted only if information gain is larger by amount cp.

- Higher cp value puts a larger requirement on information gain with every split, leading to smaller trees.
- Smaller cp values lower thre requirement on information gain with every split, leading to larger trees.

As can be seen from figure below:

- Smaller cp values, lead to better or improved metric (F1 score)
- For maxdepth upto 4, reducing cp does not improve the metric, which is intuitive, as tree stops to grow. However increasing maxdepth > 4 allows better metric.

Clearly maxdepth and cp directly help control and influence the classification model metrics.



Impact due to Minsplit and Minbucket

Minsplit is the minimum number of observations that must exist in a node in order for a split to be attempted.

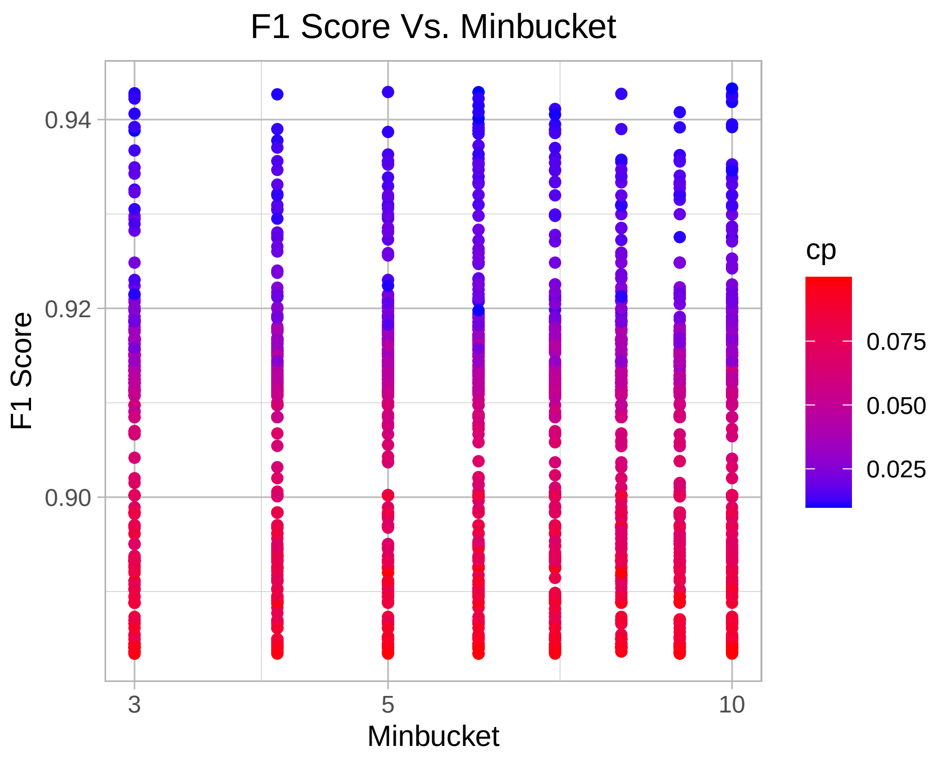
- Smaller value is expected to lead to tree growth.

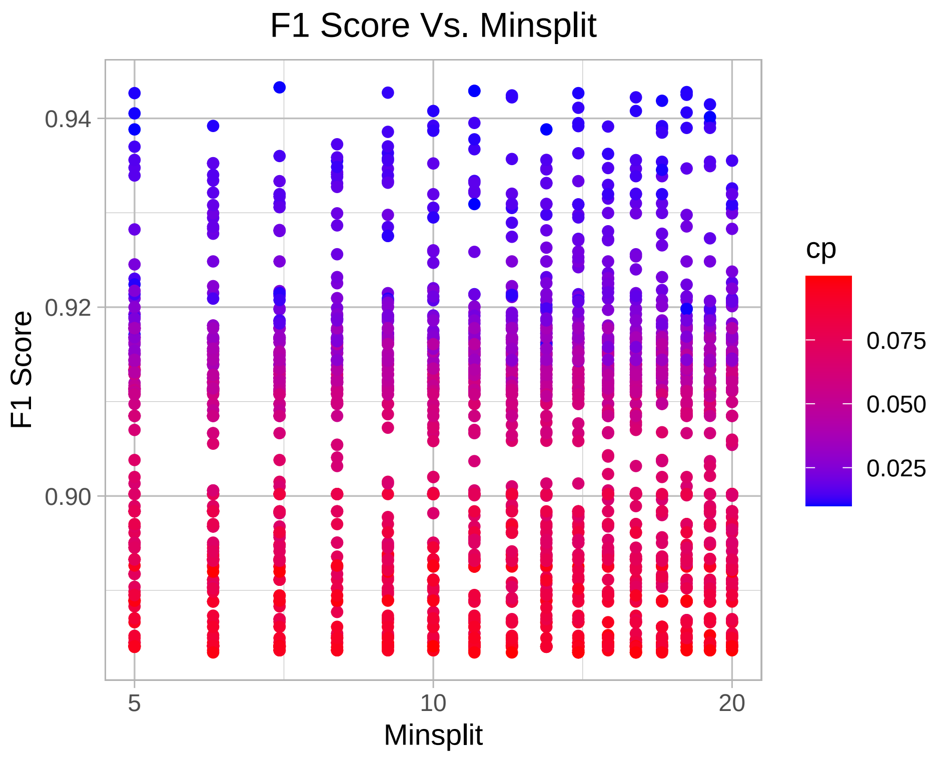
Minbucket is the minimum number of observations in any terminal node

- Smaller value is expected to lead to tree growdth

However, Minsplit and Minbucket oppose each other sometimes, and may render other value redundant. E.g. minsplit has to be larger than minbucket for both of the parameters to operate independently.

Looking at impact of Minsplit and Minbucket on F1 metric, refer figure below, does not indicate that they play a major role in determining model score like F1. It is clear that cp value is what drives the model performance.





Model Performance of Tuned Model

Tuned decision tree has following hyperparamter values:

- minsplit = 8
- minbucket = 3
- cp = 0.0125
- maxdepth = 9

Below model performance summary captures all metrics associated with classification task.

```
Confusion Matrix and Statistics


predTest    T    F
       T  353   69
       F  126 1321

               Accuracy : 0.8957
                 95% CI : (0.8809, 0.9092)
    No Information Rate : 0.7437
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7152

 Mcnemar's Test P-Value : 6.066e-05

            Sensitivity : 0.7370
            Specificity : 0.9504
         Pos Pred Value : 0.8365
         Neg Pred Value : 0.9129
              Precision : 0.8365
                 Recall : 0.7370
                     F1 : 0.7836
             Prevalence : 0.2563
         Detection Rate : 0.1889
   Detection Prevalence : 0.2258
      Balanced Accuracy : 0.8437

       'Positive' Class : T
```

Metrics used to evaluate the model are:

- Precision : 0.8365
- Specificity : 0.9504
- F1 score : 0.7836
- Recall : 0.7370
- Accuracy : 0.8957

Observations

- Model has good Specificity and Precision, indicative of smaller False Positive which is desired.
- The F1 score shows a good balance between precision and recall.

Model Interpretability & Explainability

Most important features from the model that was built after 10 fold cross validation and hyperparameter tuning is as shown below. While building model using decision trees assigns scores to these variables. Feature scores gives us clear picture which features tree found most important. So for this particular model we can clearly see feature perCaps (percentatge uppercase letters) is the most important feature followed by numLines (number of lines per message),perHTML(percentage HTML tags),numDlr (Number of dollar symbol). As per this model perCaps is the most deciding factor to know if email is spam or not then number of lines and percentage of HTML tags.

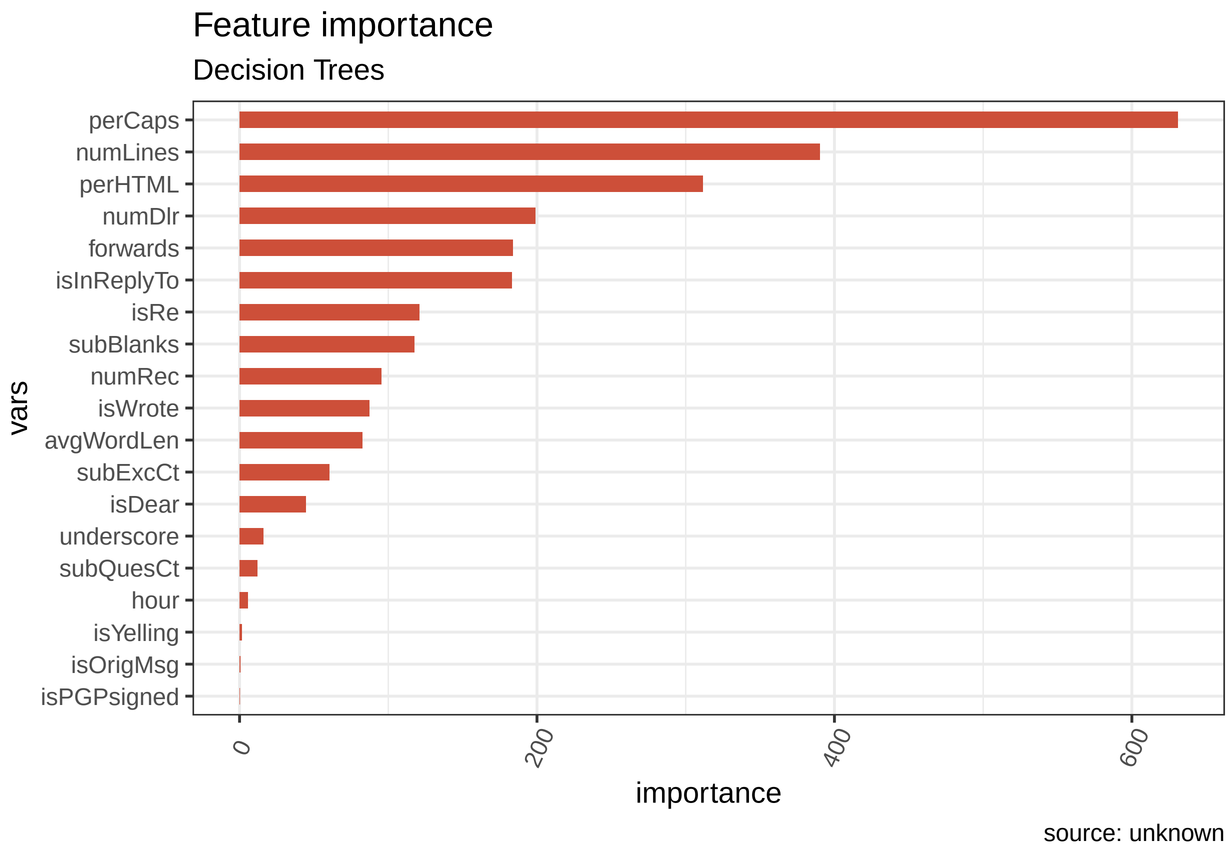


Figure 5: Feature importance

The most importatn feature is perCaps (Percentage capital letters in the body of the message) and least important feature is isPGPsigned (Signature). perCaps contributes more than any other features. Almost 600 emails have been flagged as a Spam while numLines contributes little less than 400 towards Spams out of total emails. This is quite what we see on daily basis all such Spams contains capital letters. This bar plot is quite intuative and self explanatory.

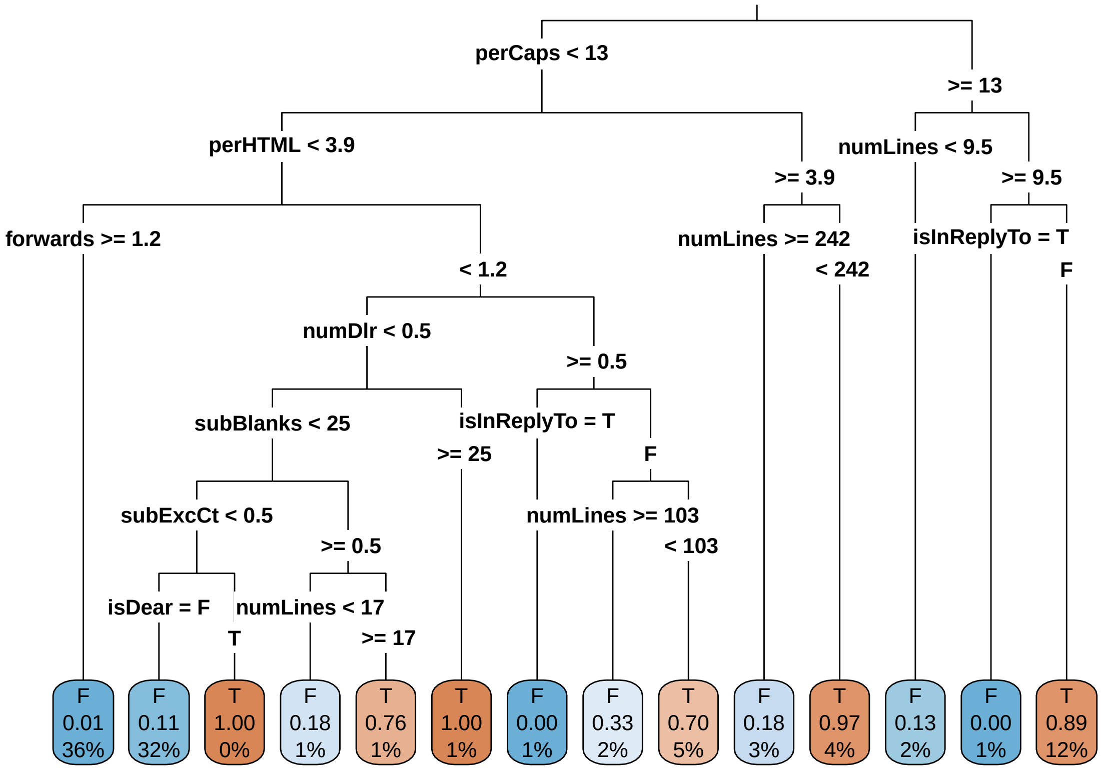


Figure 6: Decision Tree

Interpreting trees is quite straightforward. One just need to answer series of questions to go to bottom of the tree to classify message as Spam or not Spam.

Example 1

Here is one example randomly picked up from the dataset and let's walkthrough the tree from top to bottom of the tree by answering questions.

```
  isSpam  perCaps numLines  perHTML numDlr forwards isInReplyTo isRe
1      T 38.47865      101 42.38749      0 1.980198           F    F
```

Q1 - Is perCaps > 13

A1 - Yes PerCaps = 38.47 > 13 move to right branch

Q2 - is numLines > 9.5

A2 - Indeed numLines = 101 >9.5 move to right branch

Q3 - is isInReplyTo T or F

A3 - its False go to the right branch and this is leaf node.

By answering just three questions we concluded that email is Spam email. This is real advantage of decision trees. They superbly interpretable compared to other models.

Example 2

Here is another example for email that is not a spam.

```
  isSpam  perCaps numLines perHTML numDlr forwards isInReplyTo isRe
1      F 4.451039       50       0      3        0           T    T
```

Q1 - Is perCaps < 13

A1 - Yes perCaps= $4.45 < 13$ move towards left Subtree

Q2 - Is perHTML < 3.9

A2 - Yes perHTML=$0 < 3.9$ move towards left Subtree

Q3 - Is forwards > 1.2

A3 - No forwards=$0 < 1.2$ move towards right Subtree

Q4 - Is isInReplyTo = T ?

A4 - isInReplyTo = T move towards left subtree and this is leaf node which tells us that this email is not a Spam.

Again by answering just four questions we concluded that email is not a Spam.

Conclusion

An Email SPAM filtering model is developed that is capable of accurately identifying SPAM 90% of time, and with a true negative rate, or specificity at 95%. This implies very few times a non-SPAM email gets classified as SPAM.

A decision tree based classification approach is developed that can be deployed alongside automatic reading of email and doing a real time check to categorize email as a SPAM or not SPAM.

The model used following variables for classification of email in SPAM:

- forwards
- perCaps
- perHTML
- numLines
- numDlr
- subBlanks
- isInReplyTo

An email is classifies as SPAM when any of following rule is met:

- perCaps >= 13 , and numLines > 9.5, and, isInReplyTo is false
- perCaps < 13 , and perHTML > 3.9, and numLines > 242
- perCaps < 13 , and perHTML < 3.9, and #forwards < 1.2., and numDlr >= 0.5, isInReplyTo is false, and, numLines < 103
- perCaps < 13 , and perHTML < 3.9, and #forwards < 1.2., and numDlr < 0.5 , isInReplyTo is true, and, subBlanks >=25
- perCaps < 13 , and perHTML < 3.9, and #forwards < 1.2., and numDlr < 0.5 , isInReplyTo is true, and, subBlanks < 25, and subExcCt >=0.5 and numLines > 17
- perCaps < 13 , and perHTML < 3.9, and #forwards < 1.2., and numDlr < 0.5 , isInReplyTo is true, and, subBlanks < 25, and subExcCt < 0.5 and isDear = 2

Decision trees are super interpretable compared to other models/algorithms, For future work Naive Bayes, SVM, Stockastic models could also be explored and benchmarked for model performance.

References

1. Deborah Nolan; Duncan Temple Lang. Data Science in R.Chapman and Hall/CRC, 2015.
2. The statistica report
3. The Paper I Email spam filtering using decision tree algorithm.
4. The Paper II Identifying Valid Email Spam Emails Using Decision Tree.
5. Connection for Businesss website