MSDS 7333 Spring 2021: Case Study 03

Email Spam Detection

Sachin Chavan,Tazeb Abera,Gautam Kapila,Sandesh Ojha

2021 February 18

Introduction

Email spams are unsolicited emails that also referred as Junk email 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]. 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

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. Human can easily categorize suche mails and trash them manaually

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 a decision tree algorithms. Decision tree learning is the predictive modeling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about the item's target value. [Wikipedia]. We explore a decision tree package in R called rpart, which is short for recursive partitioning.

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

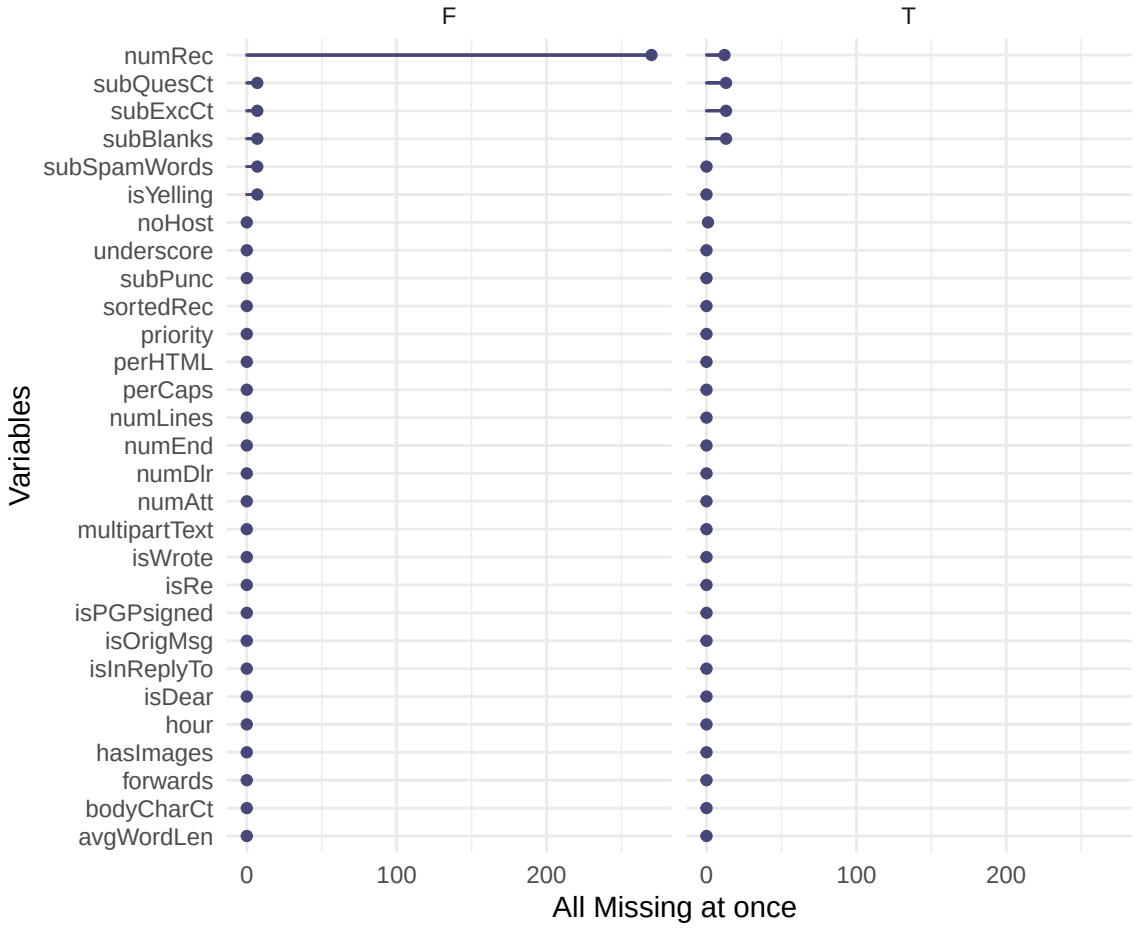

Figure 1: Left : Not Spam, Right : Spam

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

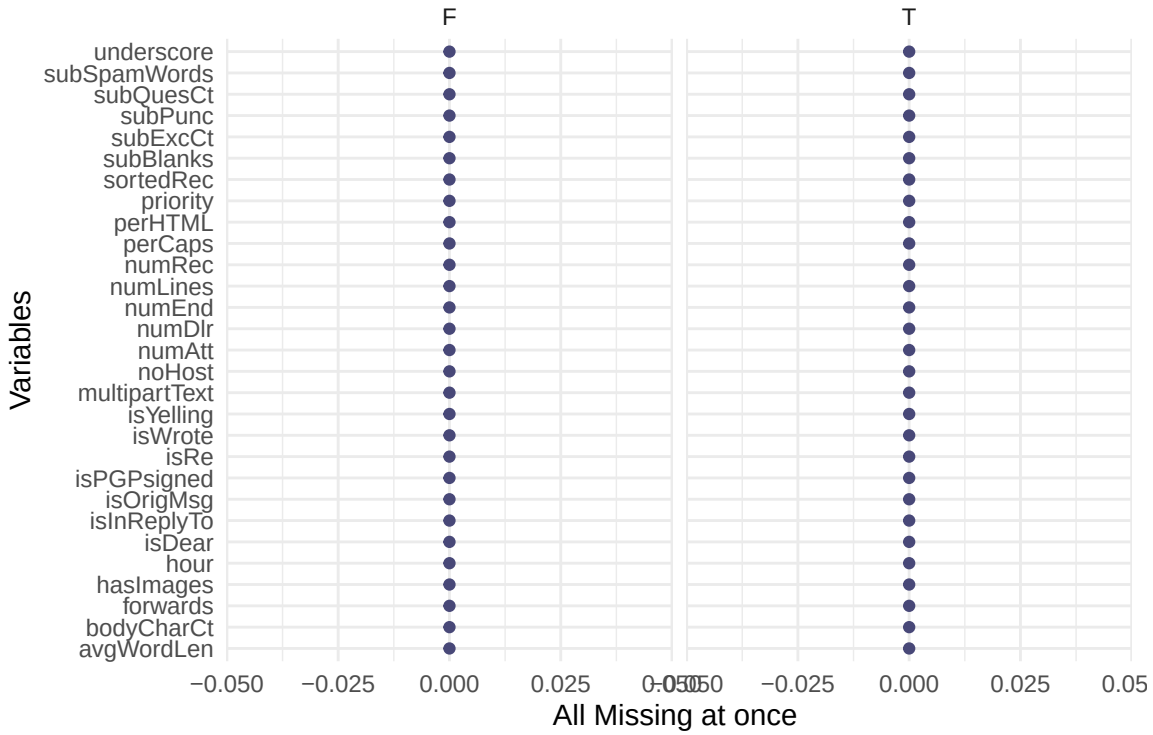

Figure 2: Left : Not Spam, Right : Spam

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.

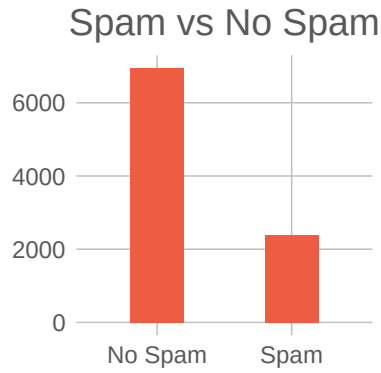

Figure 3: Target Variable

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.

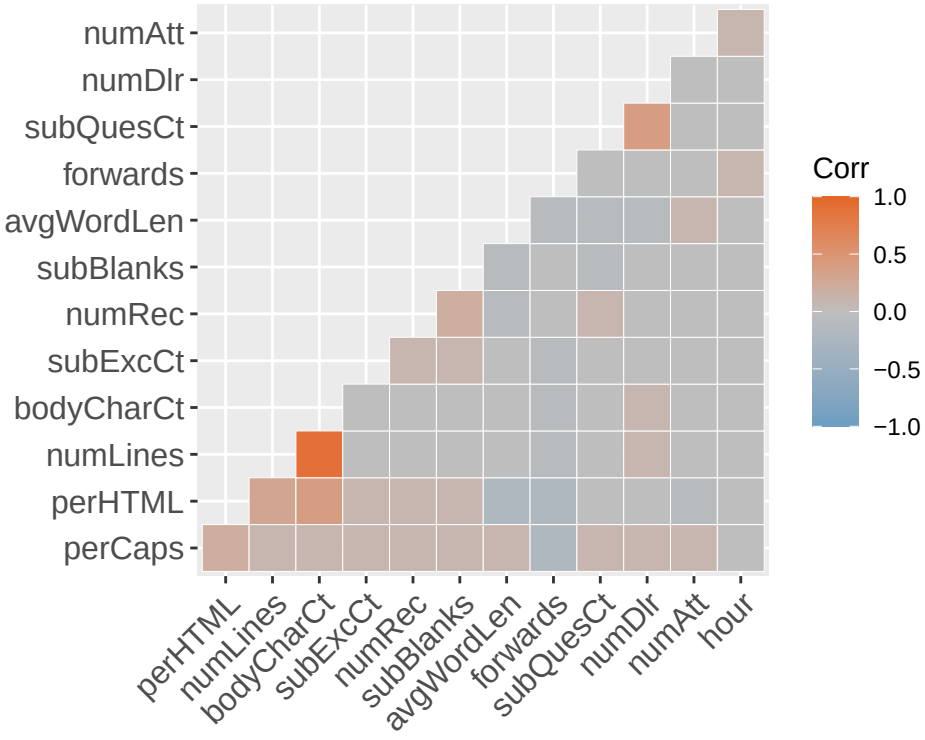

Figure 4: Correlogram

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

The primary objective of this case study is to detect Spam emails. This is binary classification problem. There are bunch of algorithms in the machine learning toolkit to solve classification problems. Logistic regression, Naive Bayes, kNN, SVM and Decision Trees

Hyperparameters

Here are decision tree hyperparameters

	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	-

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=7; minbucket=9; cp=0.0111; maxdepth=9
f1.test.mean=0.9416169,tpr.test.mean=0.9536787,ppv.test.mean=0.9299075,tnr.test.mean=0.7904280
```

```
Resample Result
Task: train.data
Learner: classif.rpart.tuned
Aggr perf: mmce.test.mean=0.0901186
Runtime: 1024.82
```

Model Interpretability & Explainability

Conclusion

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.