Case Study III

James Vasquez, Javier Saldana, Sara Zaheri

Table of Contents

Table of Figures

List of Tables

Case Study III

James Vasquez, Javier Saldana, Sara Zaheri

1. Introduction

Whether it is the rightful Prince of Nigeria who needs $15,000 to take his place in royalty or the latest Viagra product taking the world by storm, spam emails have become an ugly side of emails. While the term ‘spam email’ is not credited to a specific individual, it derives from a Monty Python sketch from the 1980s in which a group of Vikings overtake a conversation by loudly singing the word “spam”. From that sketch, the name for unwanted solicitation message was born. As a means for inexpensive mass marketing, spam email began to soar in popularity in the 1990s. In 1994, attorneys Laurence Canter and Martha Siegel sent a spam message to Usenet newsgroups advertising their immigration legal services with the subject line “Green Card Lottery – Final One?”[1]. While the President of Usenet condemned their advertising actions as a violation of Usenet’s code of conduct[2], Laurence and Martha claim to have generated an additional $100,000 in revenue because of that mass marketing campaign. Ultimately, Laurence was disbarred for his illegal advertising practices[3] and he and his wife, Martha, went on to pursue a career in digital marketing.

In 2019, 56% of all email traffic was spam, with more than 20% of it originating in China[4]. Considering the prevalence of spam in emails, the demand for a strong spam filter has become a necessity instead of a luxury. In this study, we aim to build on the text mining approach by Nolan and Temple Lang which deconstructs over 9,000 emails to into a readable format [1]. We utilize random forest to then read over the data set keywords that trigger an email to be spam and predict if the email is a spam email.

2. Methods

A technique known as Feature Selection or Feature Reduction was implemented to reduce the number of features used for analysis. This technique allows for automating the features used for analysis, it reduces overfitting, improves accuracy, and reduces training time for the model. Along with the Feature Selection a Grid Search was implemented to help train the Decision Tree that was used for feature selection. The most fundamental explanation of a grid search is to find the best hyperparameters of a model that produces the best results. A parameter is a value that can be changed for a model to best tune it for optimal results. In the decision tree model, there are two variables that allow for tuning which are “max_depth” and “min_sample_split”. The

[1] L. Canter, *Green Card Lottery – Final One?*, Google Groups publication, April 12, 1994. Accessed on: June 12. 2020. [Online].
Available: https://groups.google.com/forum/#!msg/alt.pub.coffeehouse.amethyst/XRLnYe0_zK8/l5eFCesyeEcJ

[2] J. Walen, no subject, May 20, 1994. [email]. Available: http://www.amygorin.com/ethics/cands-netcom.html

[3] A. Craddock, *Spamming Lawyer Disbarred*, Wired, July 17, 1997. Accessed on: June 12, 2020. [Online].
Available: https://www.wired.com/1997/07/spamming-lawyer-disbarred/

[4] M. Vergelis, et. al., *Spam and Phishing in 2019*, SecureList, April 8, 2020. Accessed on: June 12, 2020. [Online]
Available: https://securelist.com/spam-report-2019/96527/

maximum depth of the tree is longest path from the tree root (0 is root) to a leaf. The minimal sample split is the minimum number of samples required to split an internal node.

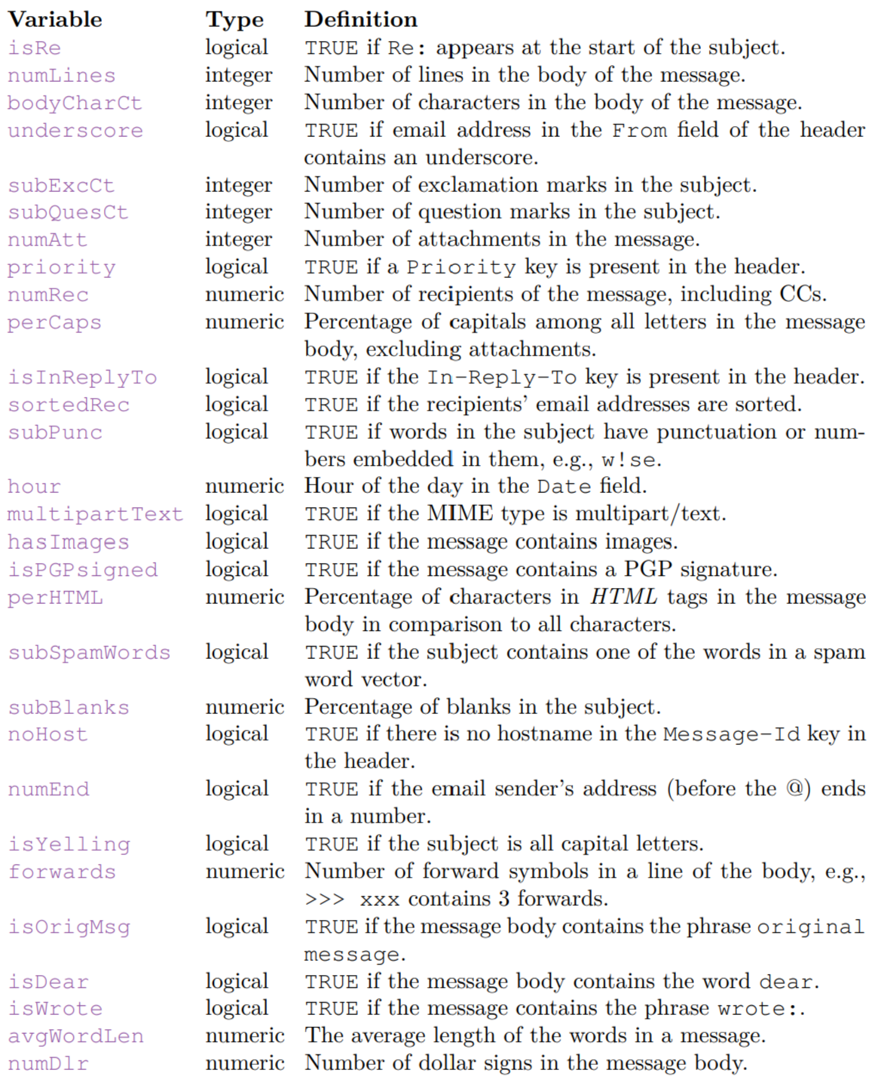
2.1. Data Source

The data set can be found at:

http://www.rdatasciencecases.org/Spam/SpamAssassinMessages.zip

The data was provided pre-formatted as an R data frame and was already deconstructed for text mining purposes. Since this study was conducted in Python, we utilized the `pyreadr.read_r()` function from the pyreadr library to import the data. Each line pertains to a unique email message and each column pertains to a characteristic in the email message. There is a total of 30 columns, which are described in the Nolan and Lang text as depicted in Table 1.

Table 1. Variable description for dataset emailDFrp provided by Nolan and Lang text.



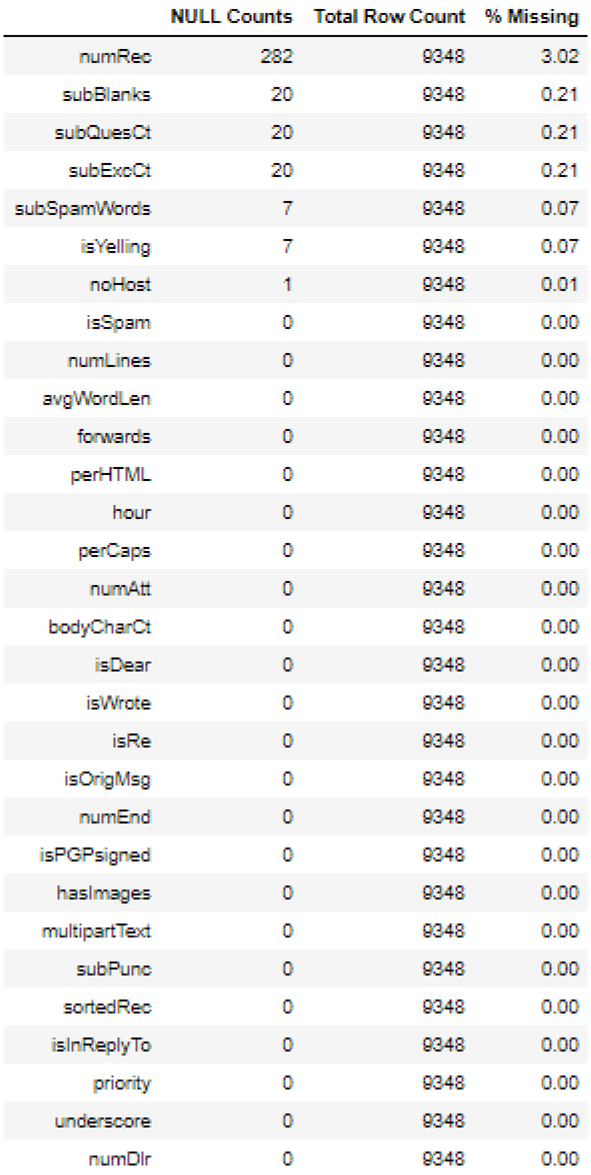
Variable	Type	Definition
isRe	logical	TRUE if Re: appears at the start of the subject.
numLines	integer	Number of lines in the body of the message.
bodyCharCt	integer	Number of characters in the body of the message.
underscore	logical	TRUE if email address in the From field of the header contains an underscore.
subExcCt	integer	Number of exclamation marks in the subject.
subQuesCt	integer	Number of question marks in the subject.
numAtt	integer	Number of attachments in the message.
priority	logical	TRUE if a Priority key is present in the header.
numRec	numeric	Number of recipients of the message, including CCs.
perCaps	numeric	Percentage of capitals among all letters in the message body, excluding attachments.
isInReplyTo	logical	TRUE if the In-Reply-To key is present in the header.
sortedRec	logical	TRUE if the recipients' email addresses are sorted.
subPunc	logical	TRUE if words in the subject have punctuation or numbers embedded in them, e.g., w!se.
hour	numeric	Hour of the day in the Date field.
multipartText	logical	TRUE if the MIME type is multipart/text.
hasImages	logical	TRUE if the message contains images.
isPGPsigned	logical	TRUE if the message contains a PGP signature.
perHTML	numeric	Percentage of characters in *HTML* tags in the message body in comparison to all characters.
subSpamWords	logical	TRUE if the subject contains one of the words in a spam word vector.
subBlanks	numeric	Percentage of blanks in the subject.
noHost	logical	TRUE if there is no hostname in the Message-Id key in the header.
numEnd	logical	TRUE if the email sender's address (before the @) ends in a number.
isYelling	logical	TRUE if the subject is all capital letters.
forwards	numeric	Number of forward symbols in a line of the body, e.g., >>> xxx contains 3 forwards.
isOrigMsg	logical	TRUE if the message body contains the phrase original message.
isDear	logical	TRUE if the message body contains the word dear.
isWrote	logical	TRUE if the message contains the phrase wrote:.
avgWordLen	numeric	The average length of the words in a message.
numDlr	numeric	Number of dollar signs in the message body.

To prepare the dataset for further analysis, the logical variables were all converted to numeric variables where True = 1 and False = 0.

2.2. Exploratory Data Analysis (EDA)

The email data set (emailDFrp) contains a binary response column “isSpam”. The “isSpam” features indicates weather or not an email is spam or not. The data set has a total 357 missing values.

Table 2. The table lists all missing values within data set.



	NULL Counts	Total Row Count	% Missing
numRec	282	9348	3.02
subBlanks	20	9348	0.21
subQuesCt	20	9348	0.21
subExcCt	20	9348	0.21
subSpamWords	7	9348	0.07
isYelling	7	9348	0.07
noHost	1	9348	0.01
isSpam	0	9348	0.00
numLines	0	9348	0.00
avgWordLen	0	9348	0.00
forwards	0	9348	0.00
perHTML	0	9348	0.00
hour	0	9348	0.00
perCaps	0	9348	0.00
numAtt	0	9348	0.00
bodyCharCt	0	9348	0.00
isDear	0	9348	0.00
isWrote	0	9348	0.00
isRe	0	9348	0.00
isOrigMsg	0	9348	0.00
numEnd	0	9348	0.00
isPGPsigned	0	9348	0.00
hasImages	0	9348	0.00
multipartText	0	9348	0.00
subPunc	0	9348	0.00
sortedRec	0	9348	0.00
isInReplyTo	0	9348	0.00
priority	0	9348	0.00
underscore	0	9348	0.00
numDlr	0	9348	0.00

From plotting the total counts of the “isSpam” column we can clearly see that the data set is an unbalanced data set.

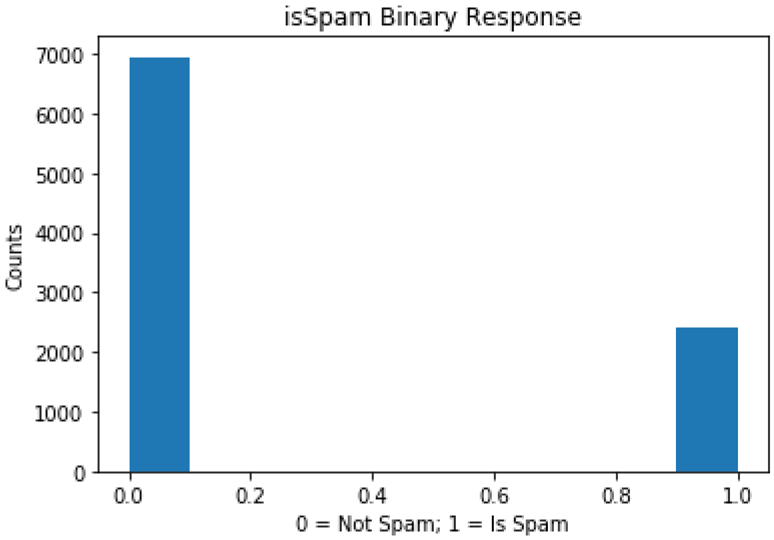


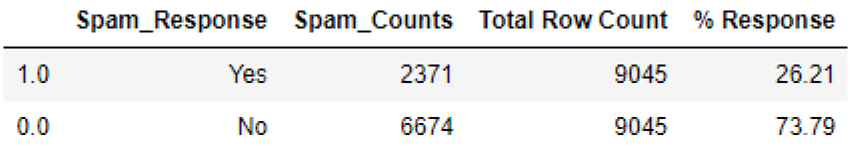
Figure 1. Total counts of the "isSpam" feature prior to dealing with NULLs.

Due to the missing data being very minimal the team decided to remove the rows rather than using a method to fill in the data.

- Original Data Frame
  - df_data Row Counts: 9348
  - df_data Columns Counts: 30
- Pruned Data Frame
  - df_pruned Row Counts: 9045
  - df_pruned Columns Counts: 30
- What was removed
  - Count of Rows Removed: 303
  - % Rows Removed: 3.24

After the rows were removed the binary response changed slightly from what is seen in figure 1.

Table 3. The table are the counts after dealing with NULLs.



	Spam_Response	Spam_Counts	Total Row Count	% Response
1.0	Yes	2371	9045	26.21
0.0	No	6674	9045	73.79

2.3. Feature Selection

Prior to conducting the Feature Selection a grid search was implemented to determine the best 'max_depth' and 'min_sample_splits' to better fit the decision tree model for feature selection. The grid search iterated through max depths of 4 to 31 with an increment of 2, minimal sample splits used a 2 to 21 with an increment of 2 to run through.

The best parameters from grid search were:

- Max_depth = 26
- Min_sample_split = 2

To easily group the features into a usable selection, we group them by quartiles which allows the team to report the most important features to be used.

Features & Variance

- 25% Quartile include 20 features and explains 99.104% of the variance
- 50% Quartile include 14 features and explains 94.237% of the variance
- 75% Quartile include 7 features and explains 74.707% of the variance

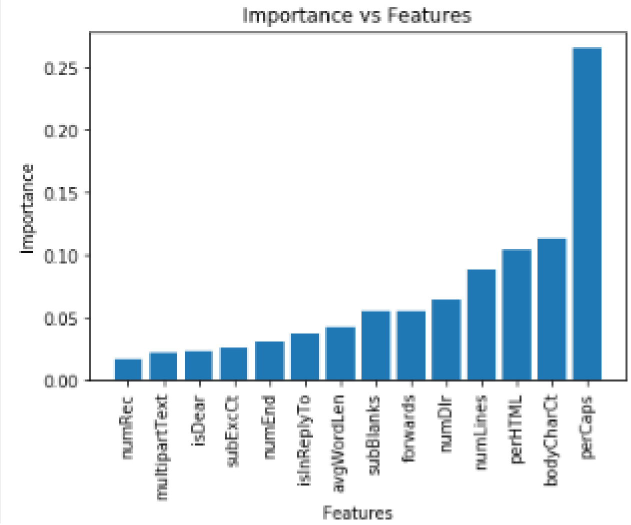


Figure 2. The completed Feature Selection which shows "perCaps" is the most important feature for predicting the "isSpam" binary response.

2.4. Analysis

After having the features selected we begin to fit random forest models. Random Forest is one of the most well-known tree based algorithms. It is indeed an ensemble method. In other words, we design several trees and let them vote for our classification problem. It results in better accuracy and avoids overfitting.

As presented in figure1 we need to keep 14 features to explain almost 95% of variance. The most important variable is ‘perCaps’. Those 14 features are able to classify sufficiently between spam and non-spam emails. I list them below:

['numRec', 'multipartText', 'isDear', 'subExcCt', 'numEnd', 'isInReplyTo', 'avgWordLen', 'subBlanks', 'forwards', 'numDlr', 'numLines', 'perHTML', 'bodyCharCt', 'perCaps']

There are parameters to be tuned in our random forest, namely, number of estimators, max depth, min sample split, and min sample leafs. We already learned the best values for min sample split and max depth for one tree. Here we study all parameters including min sample split and max depth for a random forest.

To evaluate the parameters we need to have an evaluation for the performance of a tree. In our case which is a spam recognition problem the best scores to evaluate the model are recall, precision, and f-score. We choose F-score to specify the best parameters of the random forest model. Hence, we separate our data set between test and train set (33% test and 66% train) randomly, then fit the model on the train set and finally computes the f-score for the predicted values of the model on the test set.

We begin with tuning the number of estimators. Number of estimators refers to the number of trees designed to vote. We search this number on the following set: [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]. The result of F-score is presented in Figure 3. N = 64 has the best performance.

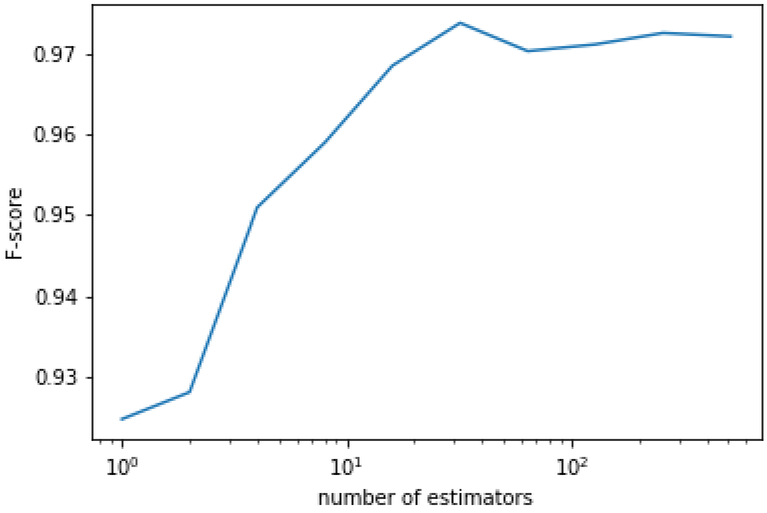


Figure 3. F-Score vs number of estimators for the random forest algorithm

Max depth refers to how deep each tree of our model could be. Searching within integers from 1 to 64 indicates that the best F-score obtains when max_depth=28. It confirms the previous search for trees in the feature selection part. I illustrate the F-score change vs max_depth in the Figure 4.

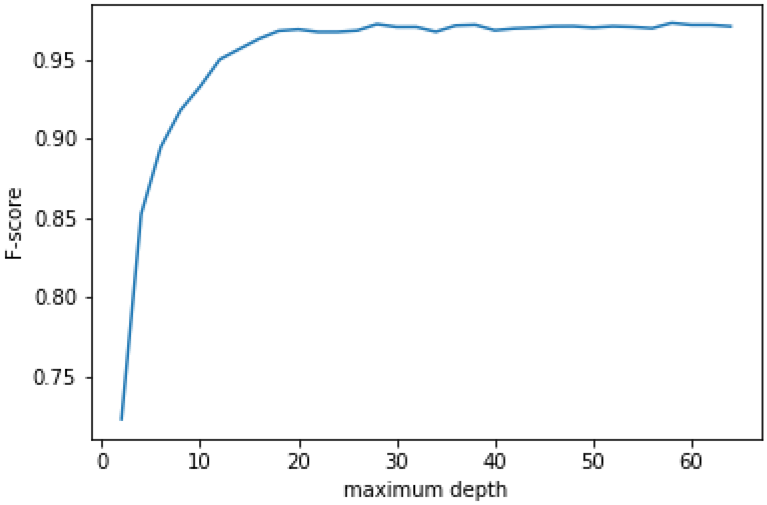


Figure 4. F-Score vs maximum depth of trees for the random forest algorithm

We consider the min sample split and min sample leafs parameters as a fraction of the size of our data set. In our case that we have almost 9000 cases, for instance, the 0.01 of them is 90 cases. Min sample split is the minimum number of samples required to split an internal node and min sample leaf is the minimum number of samples required to be at a leaf node. As we mentioned, those two numbers are considered as the fraction of the whole data set. Varying the mentioned parameters from 0.01 to 0.2 affects F-Score as it presented in Figure 6 and Figure 6.

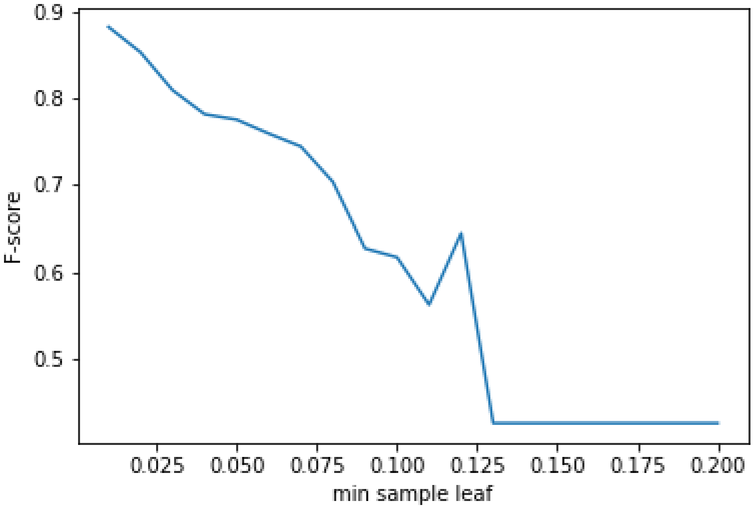


Figure 5. F-Score vs minimum sample leaf for the random forest algorithm

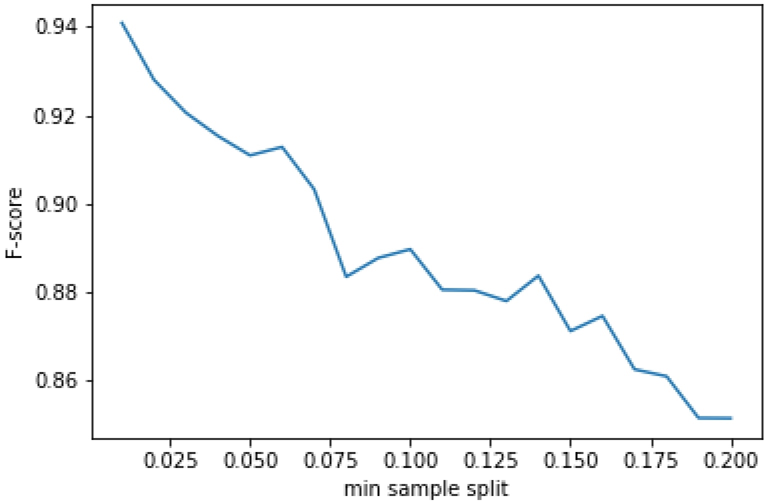


Figure 6. F-Score vs minimum sample split for the random forest algorithm

Both plots draw the conclusion that the F-score is decreasing when increasing the fraction of min sample split and min sample leaf. This also confirms our grid search result for the best min sample split in trees which was 2 nodes which is a very little fraction of our data set (2/9000 is less than 0.001). Therefore, we do not restrict our model on those two parameters.

3. Results

We have seen that the best parameters for maximum depth and number of estimators are 28 and 64 respectively. We build a random forest algorithm with them. The following scores are recall, precision and f-score of this algorithm on our test data set:

- Recall: 0.976
- Precision: 0.968
- F-Score: 0.972

As there are 64 trees and each tree could be of depth of 28 and there are also 14 variables involved in classification, it would be extremely hard to visualize a tree with detail. I inserted the picture of one of them in our random forest algorithm in Figure 9 to give an intuition about what our trees are.

On the other hand, we can investigate the trees in terms of first nodes. I present two statistics of the number of nodes in which a feature is asked amongst 64 trees, across the first or second level of tree. For just the first node we have the following statistics:

{'isYelling': 0.0, 'numRec': 5.0, 'multipartText': 0.0, 'isDear': 0.0, 'numEnd': 4.0, 'subExcCt': 7.0, 'isInReplyTo': 8.0, 'avgWordLen': 1.0, 'forwards': 9.0, 'subBlanks': 6.0, 'numDlr': 0.0, 'numLines': 2.0, 'perHTML': 8.0, 'bodyCharCt': 3.0, 'perCaps': 11.0}

And for the first and second level:

{'isYelling': 6.0, 'numRec': 9.0, 'multipartText': 4.0, 'isDear': 1.0, 'numEnd': 11.0, 'subExcCt': 19.0, 'isInReplyTo': 12.0, 'avgWordLen': 4.0, 'forwards': 19.0, 'subBlanks': 8.0, 'numDlr': 6.0, 'numLines': 8.0, 'perHTML': 29.0, 'bodyCharCt': 26.0, 'perCaps': 30.0}

One can see that the last three features (perCaps, bodyCharCt, perHTML) which were determined the most promising ones by our feature importance analysis occur more than others. I also present a figure of three levels of one of the estimator trees. X[i] means the (i+1)th feature in the list. X[12], X[13], X[14] appear in early levels of the tree as we expected.

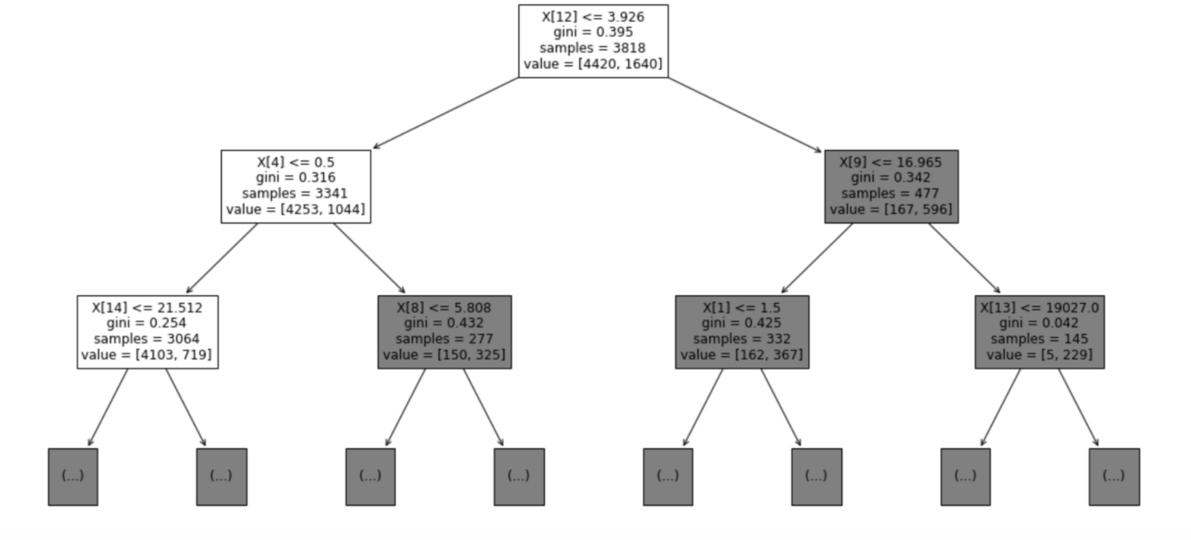


Figure 7. The first three levels of an example of estimator trees

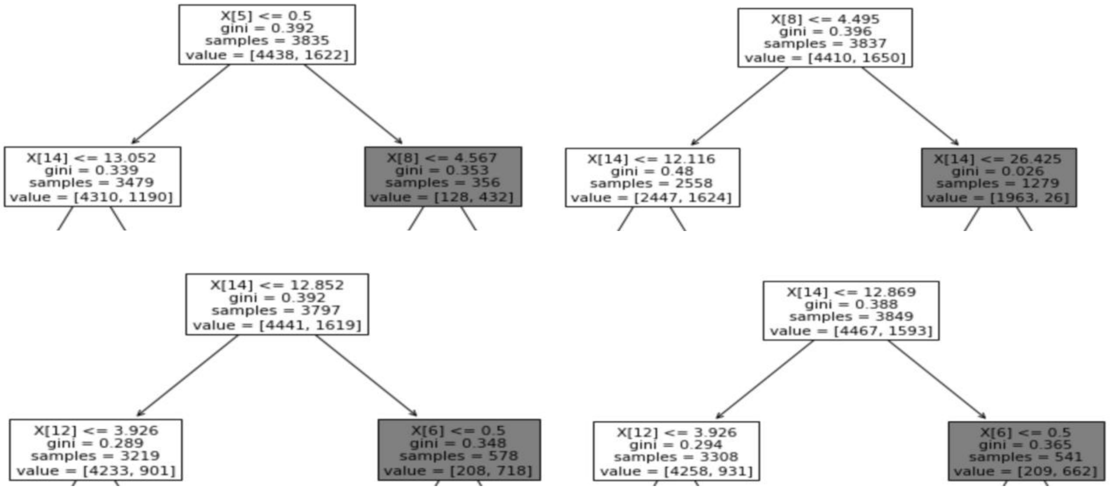


Figure 8. The first and second levels of four arbitrary trees in our random forest algorithm. It can be observed that X[14] has a significant role.

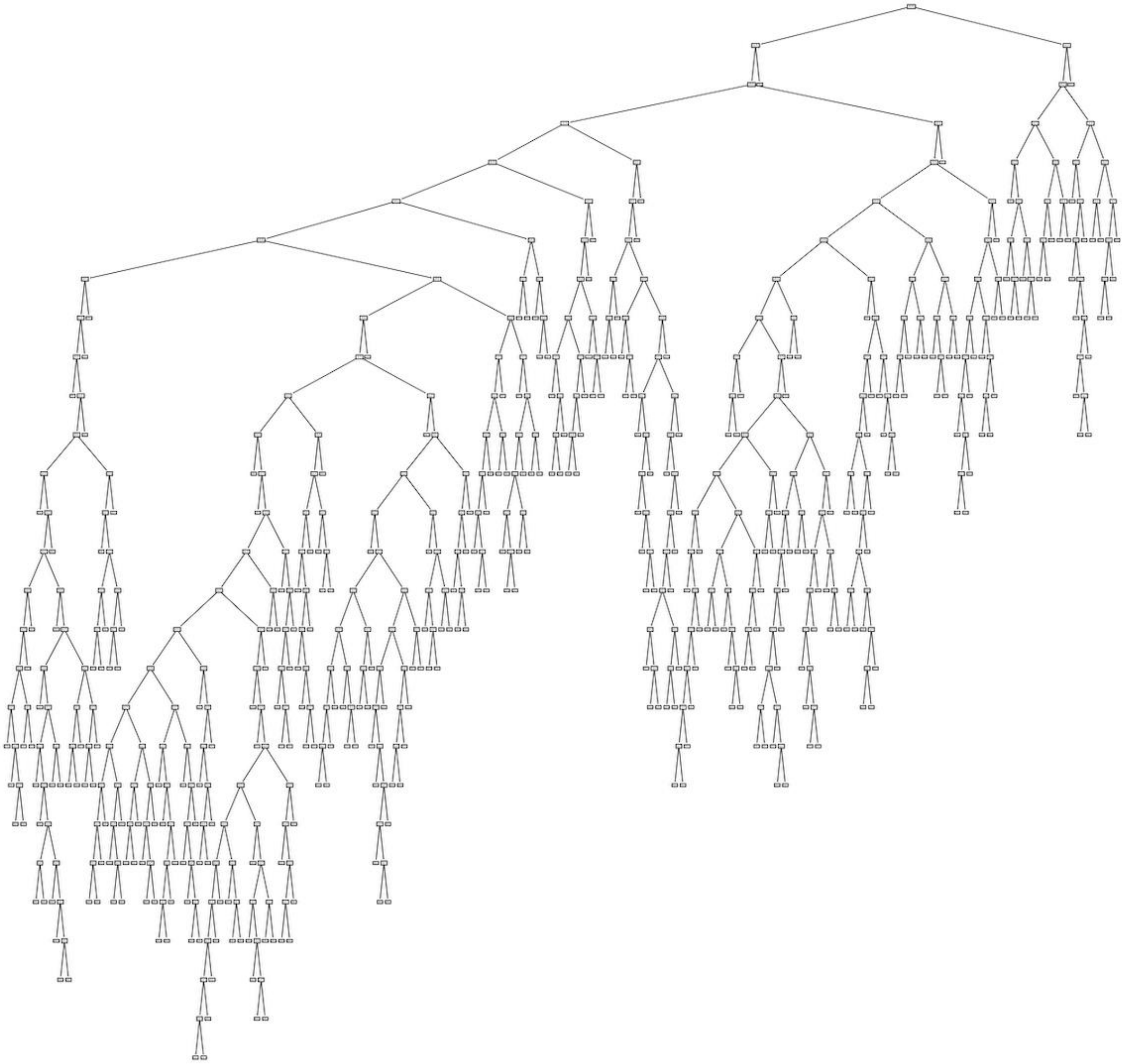


Figure 9. The general picture of an estimator trees in our random forest algorithm

4. Conclusion

Random forest is a strong ensemble learning algorithm when it comes to having too many features and the output depends highly non-linearly on some of them. In our case, we have a lot of binary variables which are also well handled by a decision tree. We begin with finding the most promising features and tune our model parameters based on F-score, which is a common evaluation criteria for problems such as spam recognition in literature.

In conclusion, the random forest model fitted to the training data set has a great performance on the test set and its f-score is 0.97. To wrap it up, contemplating all the aforementioned remarks, the random forest models provide an acceptable classifier for spam recognition.

5. References

1) Nolan, D., Temple Lang, D. DATA SCIENCE IN R: a Case Studies Approach to Computational Reasoning and Problem Solving. CRC PRESS, 2017.
2) Ho, Tin Kam (1995). *Random Decision Forests*, Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.
3) Ho TK (1998). "The Random Subspace Method for Constructing Decision Forests". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **20** (8): 832–844. doi:10.1109/34.709601
4) Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning* (2nd ed.). Springer. ISBN 0-387-95284-5.

6. Appendix (Code)

The code is implemented in Jupyter notebook. We leave this part blank and submit the .ipython file together with the project.