A large crowd of people in a stadium

Description automatically generated with low confidence

A news network aims to combat fake news by distinguishing between fake and real news. A machine learning model was developed based on detailed analysis of the news dataset using R in order to predict fake news.

COS60013 Assignment 2

Machine Learning Project

AISHABELLA SHEIKH

Table of Contents

[1. Introduction 2](#_Toc115296880)

[1.1 Data Science Pipeline 2](#_Toc115296881)

[2. Natural Language Processing 3](#_Toc115296882)

[2.1 Activate libraries and Import Data 3](#_Toc115296883)

[2.2 Data Exploration 3](#_Toc115296884)

[2.3 Data Pre-processing 6](#_Toc115296885)

[2.4 Data Visualization 8](#_Toc115296886)

[2.4.1 Word Clouds 8](#_Toc115296887)

[2.4.2 Bar Plots of Term Frequency 9](#_Toc115296888)

[2.5 Bigrams 10](#_Toc115296889)

[Bigrams in Fake News 10](#_Toc115296890)

[Bigrams in Real News 10](#_Toc115296891)

[2.6 Sentiment Analysis 11](#_Toc115296892)

[2.7 Creating a TF-IDF 12](#_Toc115296893)

[2.8 Singular Value Decomposition 13](#_Toc115296894)

[2.9 Cross Validation Results 13](#_Toc115296895)

[3. Random Forest Model 14](#_Toc115296896)

[3.1 Class Imbalance with ROSE & Random Forest 14](#_Toc115296897)

[3.2 Tuning the Random Forest Model 15](#_Toc115296898)

[3.2.1 The Best Mtry value 16](#_Toc115296899)

[3.2.2 ROC Curve 18](#_Toc115296900)

[4. Decision Tree Model 18](#_Toc115296901)

[4.1 Tuning the Decision Tree Model 20](#_Toc115296902)

[5. Predictions on Test Subset 21](#_Toc115296903)

[5.1 Preparing Test Data 21](#_Toc115296904)

[5.2 Random Forest Model Evaluation 22](#_Toc115296905)

[5.3 Decision Tree Evaluation 23](#_Toc115296906)

[6. Recommendations and Conclusions 23](#_Toc115296907)

[7. Appendix 24](#_Toc115296908)

[Appendix 1 – Dataset metadata 24](#_Toc115296909)

[Appendix 2 – Libraries 25](#_Toc115296910)

[Appendix 3 – Fake and Real Document Term Matrices 26](#_Toc115296911)

[Appendix 4 – Technical Issues with R Studio 27](#_Toc115296912)

[Appendix 5 – Word Associations 28](#_Toc115296913)

[Appendix 6 – Cosine Similarity 30](#_Toc115296914)

[Appendix 7 – Cross Validation Extended Results 31](#_Toc115296915)

[Appendix 8 – Class Imbalance Results 33](#_Toc115296916)

[8. References 36](#_Toc115296917)

# Introduction

Every day we face an overwhelming amount of news from various sources such as social media, news platforms, the internet, television and radio stations. Thus, it is becoming more difficult to know whether the news we receive is fake or true. In order to combat this problem, Today’s Network, a news network, has requested the construction of a machine learning model to classify fake news using the programming language R on a dataset from Kaggle (see Appendix 1 for dataset metadata).

Machine learning can be executed with the following stages: Data Collection and Assembly, Data Pre-processing, Data Exploration and Visualization, Model Building, Model Evaluation (Swinburne Online 2021). This report details the method and results of the text analysis using Natural Language Processing techniques, and the execution and evaluation of two predictive models built: Decision Tree and Random Forest.

## Data Science Pipeline

The pipeline for this project included these distinct steps:

1. Activate Required Libraries
2. Import Data
3. Explore and Transform Data
   1. Missing value, duplicate and null label removal
   2. Compare Text Length and Word Count by Label
      1. Visualize features with Histogram and Boxplot
   3. Subset & Shuffle Data
4. Natural Language Processing
   1. Corpus formation and transformation
   2. Document Term Frequency Matrix
   3. Data Partition into Train & Test Sets
   4. Create corpus of fake and real terms
   5. Visualize frequent terms with word clouds and bar plots
   6. Explore Bigrams and Word Associations
   7. Sentiment Analysis
   8. Cross Validation
5. TF-IDF & Cross Validation
   1. Bar plots of Frequent & TF-IDF Weighted Terms by Label
6. Singular Value Decomposition & Cross Validation
7. Model Building & Evaluation
   1. Random Forest Model + ROSE Class imbalance
      1. Evaluate & Finetune Random Forest Model
   2. Decision Tree
      1. Evaluate and Finetune Decision Tree
8. Make Predictions on Test Data with Random Forest Model

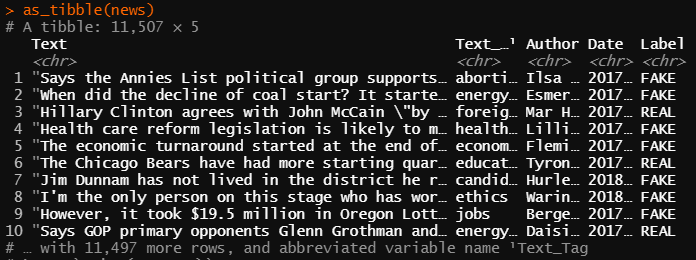
# Natural Language Processing

## Activate libraries and Import Data

The necessary R packages were installed before activating the libraries required to perform the analysis (see Appendix 2 for details of libraries).

The dataset was retrieved from the working directory as a csv file and viewed using the as\_tibble function for an overview of the dataset’s features.





## Data Exploration

Out of the original total of 11507 articles, 6594 were labelled fake, 3637 were labelled real and the remaining unlabelled 1268 were removed, as well as 8 duplicated empty rows.

Text

Description automatically generated with medium confidence

Text

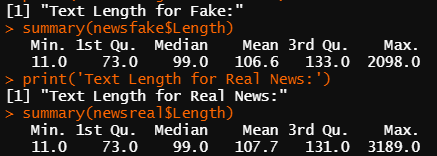
Description automatically generated

A bar plot produced using the ggplot2 library shows the proportion of fake and real news articles:

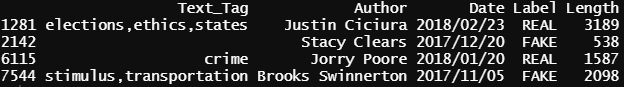
Chart, bar chart

Description automatically generated

A new column was created calculating length of text and it was found that the mean text length is longer for real news:



The text length distribution was visualized with a histogram, however, there were 4 outliers exceeding 500:



Chart, histogram

Description automatically generatedOutliers were evenly distributed between fake and real, so they were removed to produce the following histogram:

Length of text for both labels are both positively skewed, and it was found that the mean text length for fake news was 106.2, while the mean length for real news was 106.4, a very small difference, however real news had a higher maximum length of 395 compared to fake news length of 391.

Word Count was also explored, visualized with a boxplot:

Chart, box and whisker chart

Description automatically generated

The distribution for word counts in articles are generally equal between labels.

The clean data set was then shuffled and a subset of 5000 rows was extracted as a new data frame, with reset indices:

A picture containing text

Description automatically generated

Text

Description automatically generated

## Data Pre-processing

The textual data must be prepared for input into a machine learning model by undergoing several processes using the text mining ‘tm’ and ‘SnowballC’ packages.

A bag of words model was first built by creating a corpus containing the text column for analysis. This corpus was created with by removing stop words, punctuation, numbers, white space, changing all words to lowercase and stemming.

Text

Description automatically generated

Two examples of the transformation differences are demonstrated in the table below:

|  |  |
| --- | --- |
| Before Pre-processing | After pre-processing |
| [1] "Says Edward Snowden could have gotten all of the protections of being a whistleblower." | [1] say edward snowden gotten protect whistleblow |
| [2] "Secretary Clinton changes her position on (gun issues) every election year, it seems, having one position in 2000 and then campaigning against President Obama and saying we dont need federal standards." | [2] secretari clinton chang posit gun issu everi elect year seem one posit campaign presid obama say dont need feder standard |

Next, the document term matrix was constructed. At 100% sparsity, there were 5934 terms, and at 99.9% sparsity, the number of terms reduced to 1483.

Text

Description automatically generated

The data was split 70/30 into train and test sets:

Text

Description automatically generated

The final step of pre-processing involves converting the document term matrix into a data-frame and appending the labels from the original dataset and making column names syntactically valid:

Text

Description automatically generated

The training data frame now looks like this:

A picture containing table

Description automatically generated

The raw training data was also split into Fake and Real subsets and underwent the corpus cleaning process (see Appendix 3 for document term matrices summary).

## Data Visualization

### Word Clouds

The term frequencies can be visualized with the Word Cloud library:

Text

Description automatically generated

Text

Description automatically generated

The most prominent word is ‘say’, and it seems that most words have political connotations, including the names of US politicians.

The word cloud code was adjusted to use specific colours from the Spectral brewer palette of the RColorBrewer package. There were also a lot of technical issues with RStudio in creating the word clouds (see Appendix 4).

****

**Text

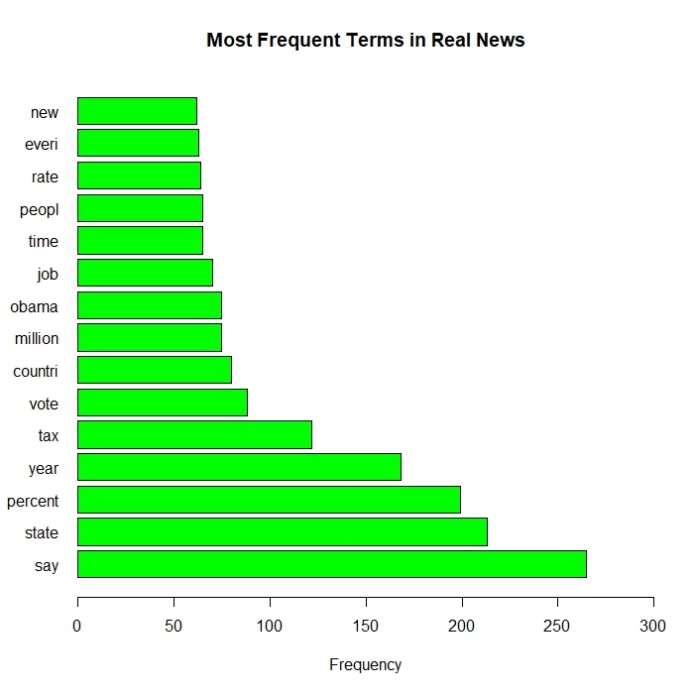
Description automatically generatedText

Description automatically generated Word Cloud of Fake Terms Word Cloud of Real Terms**

From the word clouds, it can be deduced that real news has more emphasis on ‘percent’ which probably means the articles use a lot more statistical evidence, whereas fake news has an emphasis on ‘obama,’ so fake news tends to spread more false information about people.

### Bar Plots of Term Frequency

The 15 most frequent terms in real and fake news are visualized below:

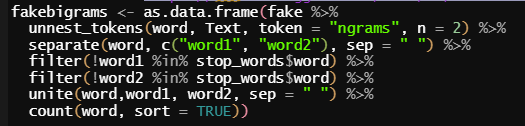
Chart, bar chart

Description automatically generated

Between the two labels, 11 out of 15 top terms are shared. Terms unique to fake news include ‘presid’, ‘health’, ‘care’, and ‘one’.

## 2.5 Bigrams

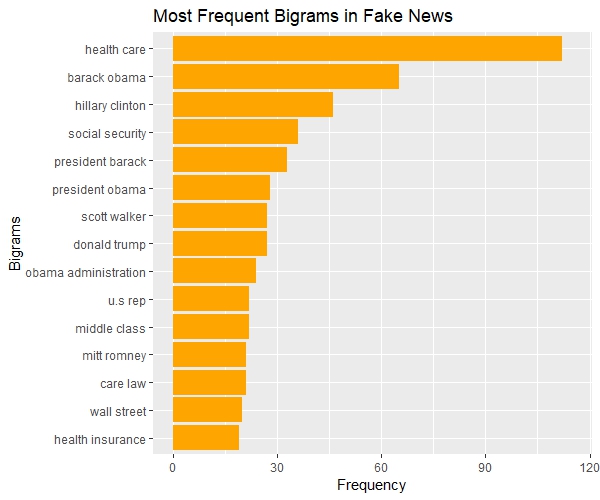
A bigram data frame was constructed separately for fake and real terms using the following code format:



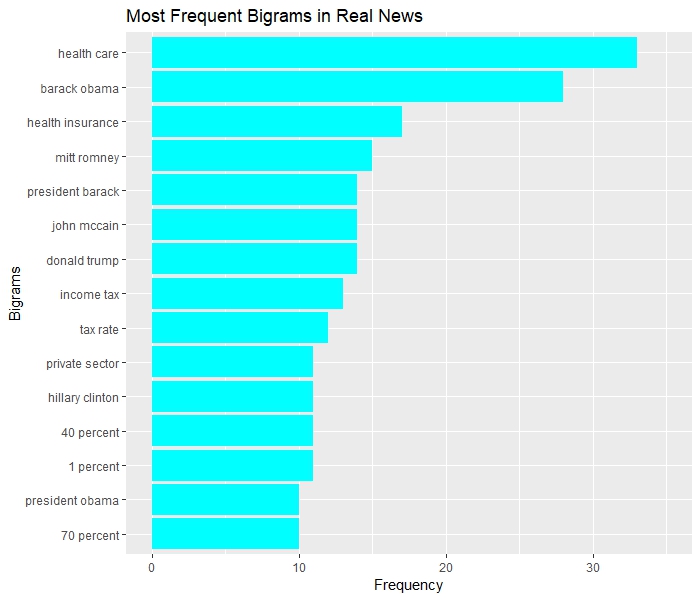
### Bigrams in Fake News

A picture containing diagram

Description automatically generated



### Text, timeline Description automatically generatedBigrams in Real News



As hypothesized, real news articles include more numerical values and statistics, with three top bigrams including 1, 40 and 70 percent. While 7 out of the 15 top bigrams in real news are people’s names, 8 of the 15 in fake news includes names, with one bigram ‘obama administration’ not seen in any of the real news bigram visualizations.

## 2.6 Sentiment Analysis

The ‘textdata’ package contains a lexicon known as ‘affin’ that assigns words values between -5 and 5 according to whether the sentiment is negative or positive (Zhang 2022). The overall sentiment can be calculated by adding all the values of each individual word together, which was applied to the training data with the following code format (Silge 2018):

Chart, bar chart

Description automatically generatedChart, histogram

Description automatically generatedText

Description automatically generated

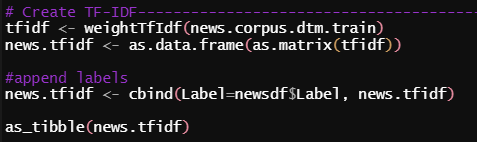
Chart, histogram

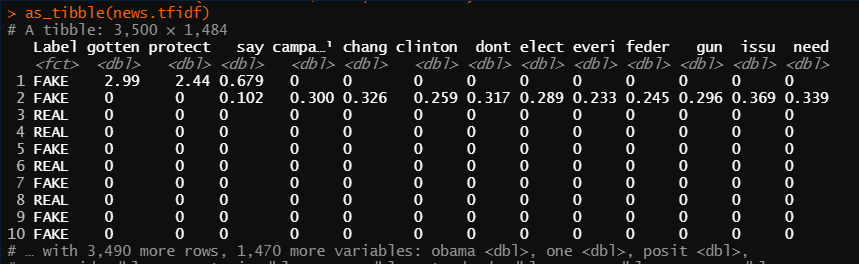
Description automatically generatedIt is interesting to note how more articles labelled real have negative summarized scores, however, real articles have a maximum total affinity at 13, while the highest total affinity for fake articles was 10.

## 2.7 Creating a TF-IDF

TF-IDF normalizes the frequency of a term across documents so that more common words have a lower score and rarer words have a higher score, hence it is a useful way to extract keywords from a text (Zheng & Casari 2015).

The TF-IDF was created using the weightTfIdf function included in the tm package:





Instead of frequency, the TF-IDF vector has been applied. This has altered the top terms for each label by retrieving the sum of TF-IDF:

Chart, bar chart

Description automatically generatedChart

Description automatically generated

The main difference between the TF-IDF weighted bar plots and the frequency bar plots is that the top term for real news has changed to ‘percent’ instead of ‘say’.

## 2.8 Singular Value Decomposition

The most relevant features were extracted from the TF-IDF terms matrix by using the irlba package:

Text

Description automatically generated

The term matrix was transformed into a more compact representation of the data to show approximate right singular vectors (DataScienceDojo 2017). The TF-IDF document was then mapped into the Singular Value Decomposition semantic space:

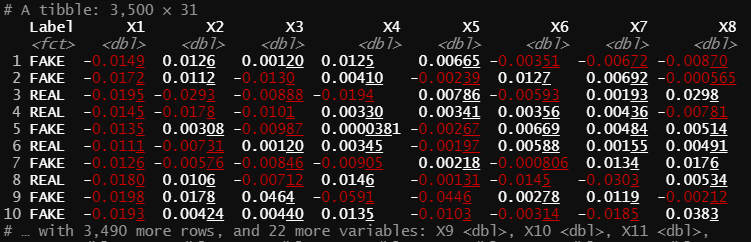
Text

Description automatically generated

The labels were then added to the v matrix and converted into a data frame:

Text

Description automatically generated



So instead of 1483 features, there are now only 30, which makes it quicker and easier to use in predictive modelling.

The cosine similarity was also calculated, however, when visualizing the mean cosine similarity, the similarity between real and fake news was too high, therefore it was not included as a variable (see Appendix 7 for cosine similarity details).

## 2.9 Cross Validation Results

Three cross validation decision trees were run between each transformation of the matrix and the results are in the table below (full results in Appendix 5)

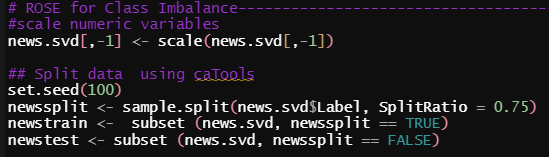
|  |  |  |  |
| --- | --- | --- | --- |
| *Added Features:* | Accuracy | Complexity parameter | Kappa |
| Term Frequency | 62.66% | 0.003921569 | 0.02558987 |
| TF-IDF | 63.39% | 0.009411765 | 0.02739286 |
| SVD | 63.4% | 0.010980392 | 0.06091025 |

The accuracy increased with each added transformation, as well as the complexity parameter.

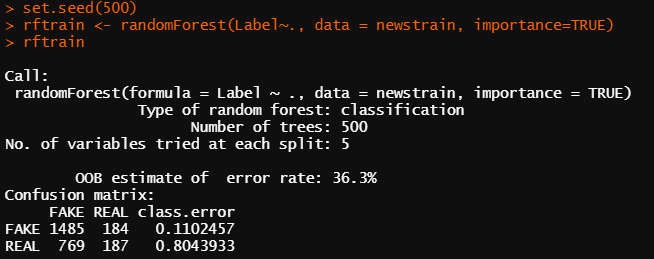
# Random Forest Model

## 3.1 Class Imbalance with ROSE & Random Forest

The training data was split 75/25 using the caTools package to maintain the baseline accuracy of label distribution to explore class imbalance in model results.

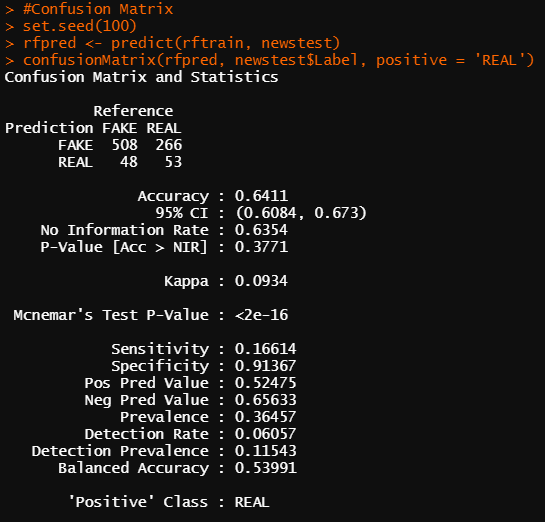


First, a random forest model was built on the training data:



From the confusion matrix, it looks like the model is better at classifying fake articles as fake, with only 11% error, while there is 80.4% error in classifying real articles as fake, so there are a lot of false negatives.

The random forest model was tested on the test set and the following results were obtained:



The accuracy is 64.11%, which is higher than the last cross validation (63.49%). High specificity and low sensitivity imply that there are a lot of false negatives.

Using the following confusion matrix table as a reference for calculations:

|  |  |  |
| --- | --- | --- |
|  | Reference | |
| Prediction | FAKE | REAL |
| FAKE | True Negatives (TN) | False Positives (FP) - **Type I Error**  (FP Rate = 1-Specificity) |
| REAL | False Negatives (FN) – **Type II Error**  (FN Rate = 1-Sensitivity) | True Positives (TP) |

The false positive rate of the random forest model is 8.63%, which means that 8.63% of real articles are correctly classified as real, and the false negative rate is 83.38%, so 83.38% of real articles were classified as fake.

The results of the oversampling and under-sampling as well as the ‘both’ method of the ovun.sample function in the ROSE package are summarized in the table below (see Appendix 8 for full results):

|  |  |  |  |
| --- | --- | --- | --- |
|  | Over Sampling | Under Sampling | Both |
| Accuracy | **65.26%** | 56.57% | 57.6% |
| False Positive Rate | **9.35%** | 42.09% | 34.71% |
| False Negative Rate | 78.997% | **45.77%** | 55.8% |
| Precision (TP/(TP+FP) | 21% | **54.2%** | 44.2% |
| Recall (TP/(TP+FN) | **56.3%** | 42.5% | 42.21% |
| F1 score (2TP / (2TP + FP + FN) | 30.58% | **47.64%** | 43.18% |
| OOB Error Rate | **13.81%** | 45.03% | 15.05% |

* The bolded cells represent the best results. It seems that over sampling has better accuracy and lower estimated OOB error rate, however, under-sampling has a lower false negative rate, and the best precision.

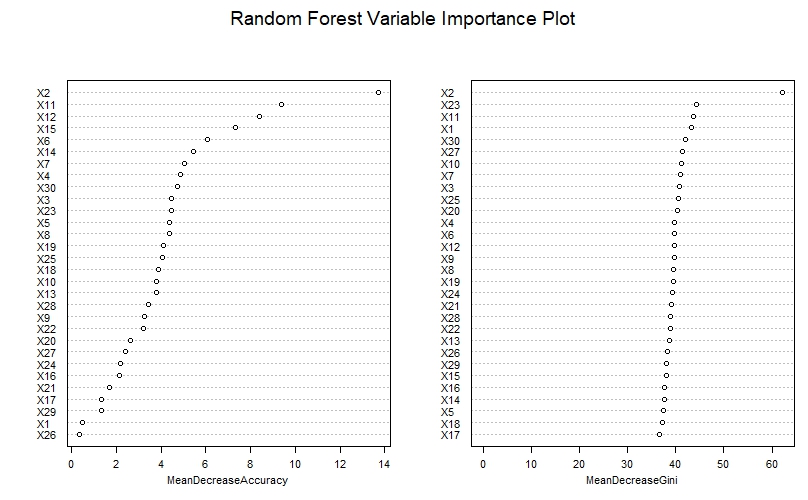
## 3.2 Tuning the Random Forest Model

A variable importance plot was constructed from the original random forest model to see which variables contribute the most to the accuracy:

Graphical user interface, text

Description automatically generated

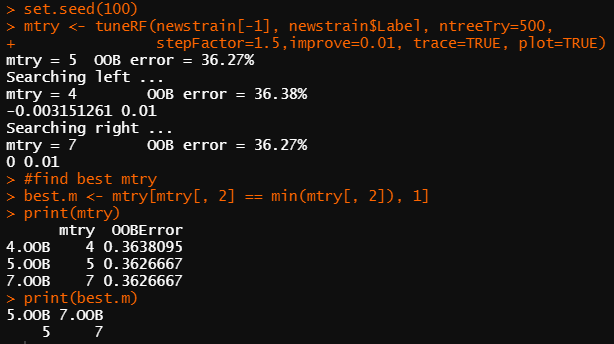
The variable importance plots for the over, under and both sampling methods are found in Appendix 8.



The top variables that influence accuracy are X2, X11, X12 and X15.

### 3.2.1 The Best Mtry value

To improve the Random Forest model, the best mtry must be selected, which is the number of random variables used in each tree (Bhalla D, 2014).



Chart

Description automatically generated

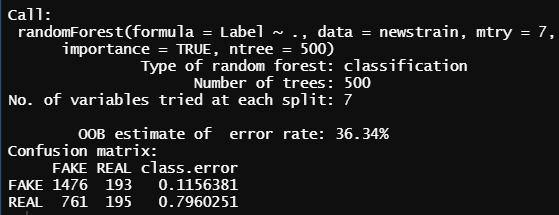
It appears there are 2 best mtry values, which means the lowest value will be selected and used in a new random forest model:



Text

Description automatically generated

The default mtry value was also 5, and the difference from the first random forest model is that the class error for fake news decreased, while the class error for classifying real news increased.

However, when using an mtry value of 7, the class error for real decreased by around 0.2%.

### 3.2.2 ROC Curve

The following curve depicts the performance of the improved random forest model with mtry = 5.

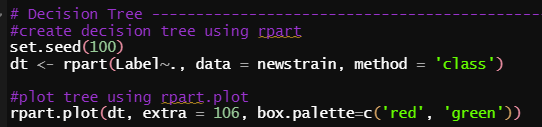
Chart, line chart

Description automatically generated

Ideally, the curve would be further away from the midline, however, this model is not an ideal model for classifying fake and real news.

# Decision Tree Model

Decision trees can perform both classification and regression tasks and can be visualized using the rpart.plot package.



Diagram

Description automatically generated

The decision tree uses X2 as the first node and again as a last node, and from the variable importance plot using Random Forest, X2 had the highest importance in determining accuracy. A total of 4 different variables were used for this tree.

The model was evaluated by predicting on the test subset:

Text

Description automatically generated

Text

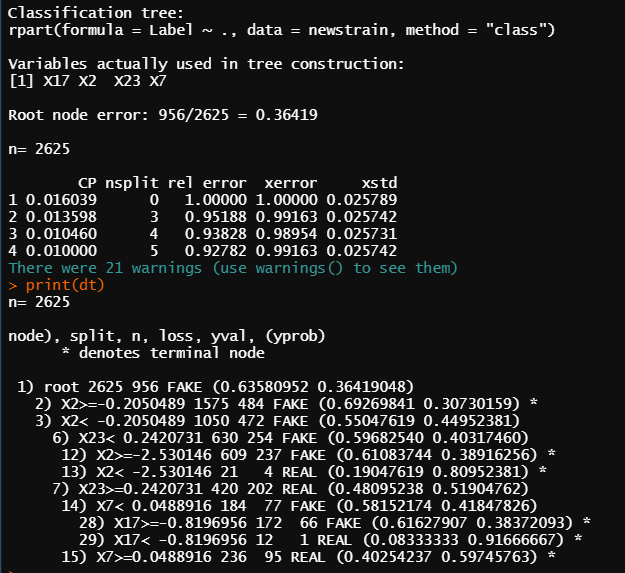
Description automatically generated

The accuracy is lower than the random forest model, with a lot of false positives. However, the model is better at predicting the fake class, perhaps due to it being a majority.

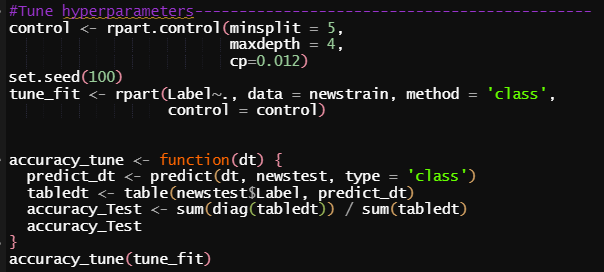
## Tuning the Decision Tree Model

Chart, box and whisker chart

Description automatically generatedThe complexity parameters of the decision tree were evaluated with the plotcp() and printcp() functions. The following figure is a pruning plot of the cross validation error against the complexity parameter values.

The best place to split the tree is when the x-error, which is the cross-validation error, is the lowest, in this case it is at the 5th node, where x-error = 0.989.

The hyper-parameters were tuned by using rpart.control (Johnson 2022)



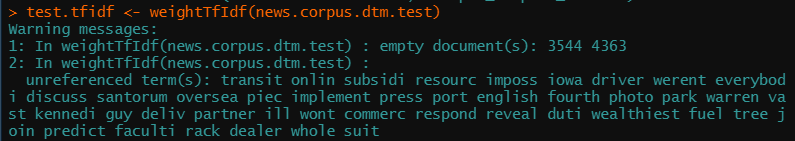
Using these parameters on the decision tree model with a minsplit of 5, a maxdepth of 4 and a cp of 0.012, the accuracy increased from 62.97% to 63.2%.



# Predictions on Test Subset

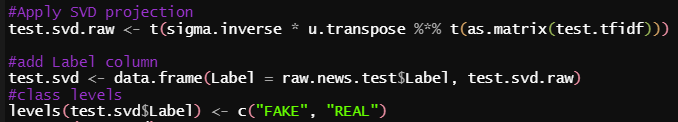
## 5.1 Preparing Test Data

The document term matrix test set saved under the variable ‘news.corpus.dtm.test’ will be used, as it is unseen data with the same terms, so all the same matrix transformations applied on the training set can be applied.



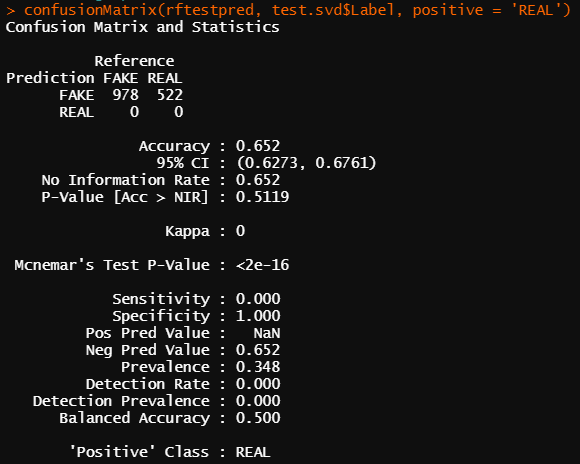
There were 2 documents that were completely empty but will be kept in the document in order to keep the TF-IDF valid and align labels accordingly.

The SVD projection was applied to the matrix by multiplying the vectors calculated on the training data.



## 5.2 Random Forest Model Evaluation

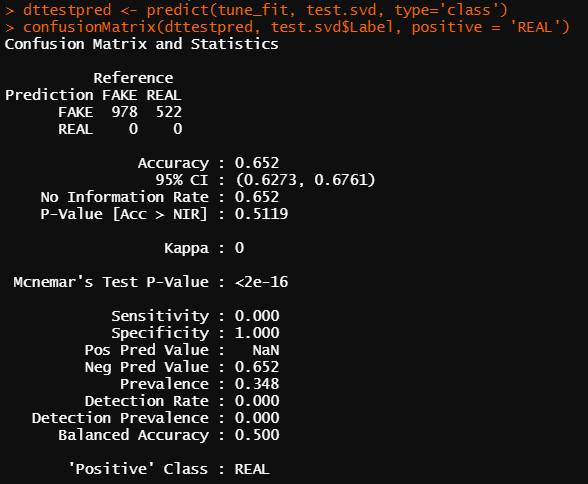
The finetuned random forest was applied to the test data:



It seems the model did not perform well, despite having a 65.2% accuracy, all articles were classified as fake due to the proportion of classes within the test set.

## 5.3 Decision Tree Evaluation

The same problem occurred with the decision tree model, where there was a 100% false positive rate and 100% true negative rate.



The reason the models did not work might be due to the projection of the same singular value decomposition values to the test data, because the term frequencies may be different and would therefore affect the model. Another possible cause could be related to the TF-IDF weighting, as a separate TF-IDF was calculated for the test data.

# Recommendations and Conclusions

From the data exploration, there did not seem to be a large difference in fake and real news when looking at statistics such as number of characters in a text or word count. However, when using natural language processing techniques such as bigrams and sentiment analysis, there was a more evident pattern in how words were used. For example, fake news tend to centre around people, which in this case were mostly politicians, while real news were more likely to provide statistical evidence, as the word ‘percent’ was the top term when the TF-IDF weighting was applied.

Sentiment analysis using the affinity method may not be effective as some words used have no weighting, and a sentence containing many negative words may be negative, especially in the real news case where a tragic incident is being reported.

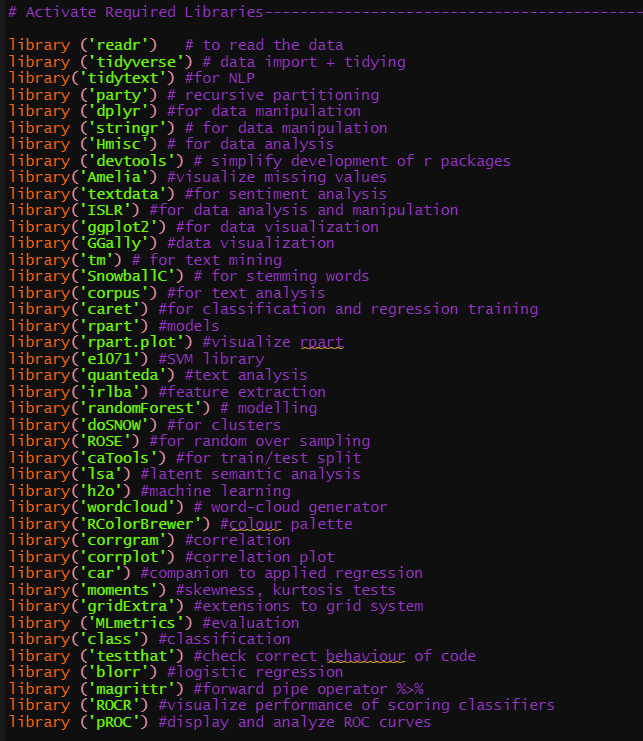
The random forest model oversampling method had the highest accuracy, however, the recurring problem in all models was the high number of false negatives, as the similarity between real and fake news is hard to distinguish in machine learning without adding more features and using more sophisticated machine learning algorithms, perhaps combining a few models in order to get the highest accuracy. Fake news will always exist, but the best way to detect it would be by building a more complex model than a random forest or decision tree alone.

# Appendix

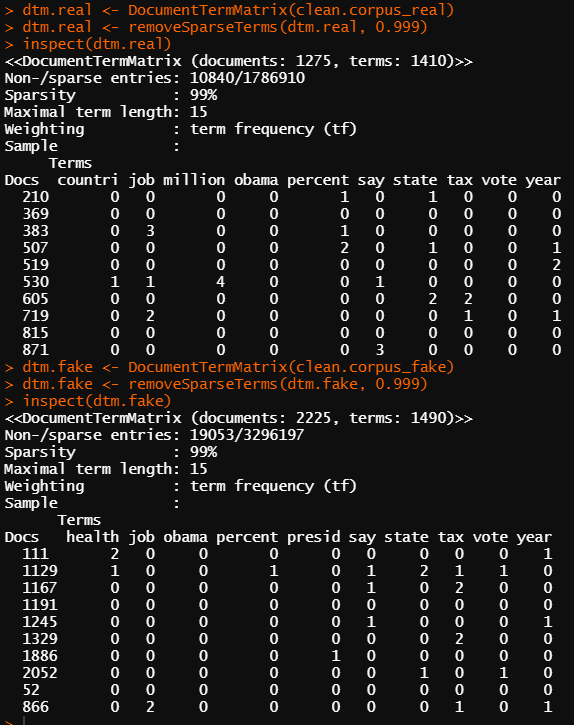
## Appendix 1 – Dataset metadata

|  |  |  |  |
| --- | --- | --- | --- |
| news.csv | **Column** | Sample record | Interpretation of columns |
| Text | Says the Annies List political group supports third-trimester abortions on demand | Raw content from social media or news platforms |
| Text\_tag | abortion | Different types of content tags |
| Author | Ilsa Mathiasen | Name of the author |
| Date | 2017/08/30 | Publication |
| Labels | FAKE | Indication of fake or real news |

## Appendix 2 – Libraries

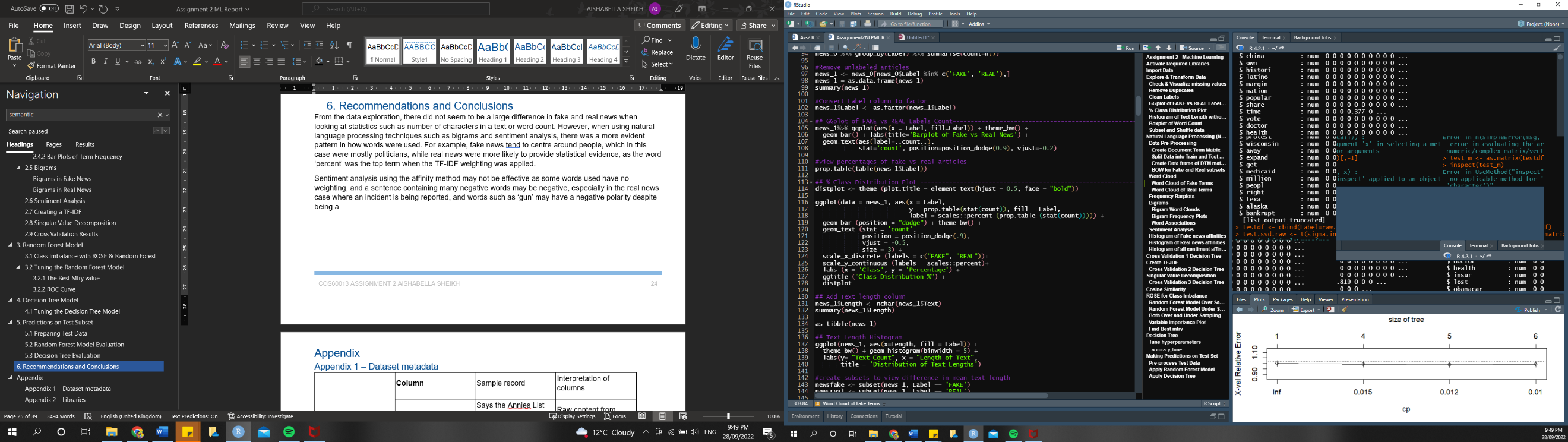


## Appendix 3 – Fake and Real Document Term Matrices



## Appendix 4 – Technical Issues with R Studio

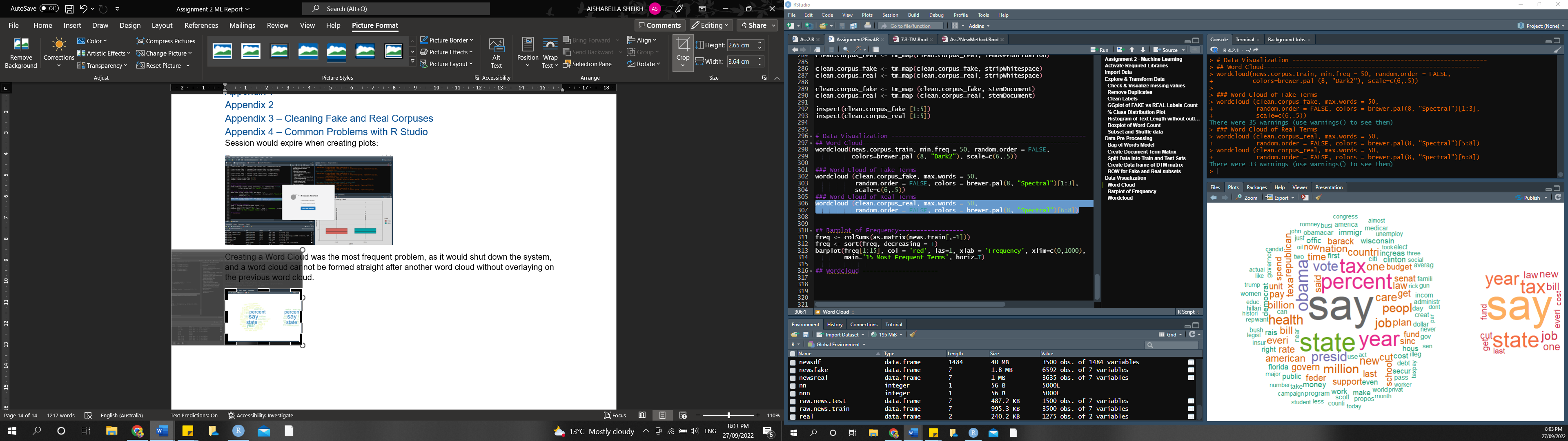
The RStudio Session would terminate and encounter a fatal error frequently, especially when creating plots, and then the entire workspace must be reloaded again. Another problem included the glitching and freezing, which may be due to the computer that was used for the machine learning.

Graphical user interface

Description automatically generated

Creating a Word Cloud was the most recurring problem, as it would shut down RStudio, and a word cloud or any other type of plot cannot be formed straight after another word cloud without overlaying on the previous word cloud. Another problem was the printing of certain plots cropping out half the graph and not loading properly.

Text

Description automatically generated with medium confidence

Chart, histogram

Description automatically generatedChart, funnel chart

Description automatically generated

Another issue is reproducibility of results, as each time the model runs in a new session, regardless of setting the seed, the model will produce different results.

These recurring issues were what can cause machine learning to take a long time, and it can be troublesome when dealing with large datasets.

## Appendix 5 – Word Associations

**Word Associations between Fake Terms:**

Text

Description automatically generated

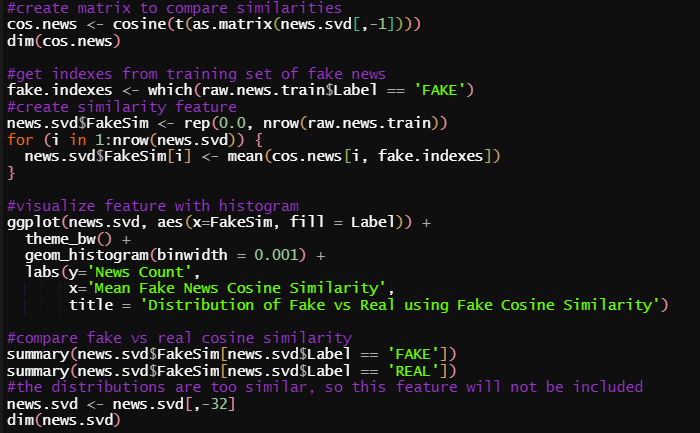
**Word Associations between Real Terms**

Calendar

Description automatically generated

## Appendix 6 – Cosine Similarity

Code snippet sourced from DataScienceDojo:



Chart, line chart

Description automatically generated

Text

Description automatically generated with medium confidence

Since the distribution of both real and fake news similarity was too high, the cosine similarity feature was not included in the data.

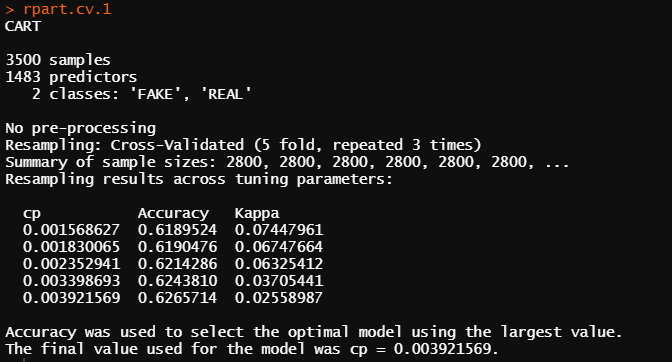
## Appendix 7 – Cross Validation Extended Results

**Cross Validation 1: after creating the term frequency matrix**

Text

Description automatically generated

The caret package ‘train’ function was used to build a single rpart decision tree for repeated cross validation, which is a fast and efficient way to train the data (DataScienceDojo 2017). The socket clusters are added to allow for parallel processing and are registered using the doSNOW package so caret can recognize the clusters and train in parallel (DataScienceDojo 2017). 2 logical cores are used as the computer used to operate this only has a maximum of 4 cores, so in order to not overload the CPU, only 2 will be used, leaving 2 for the operating system. (DataScienceDojo 2017).

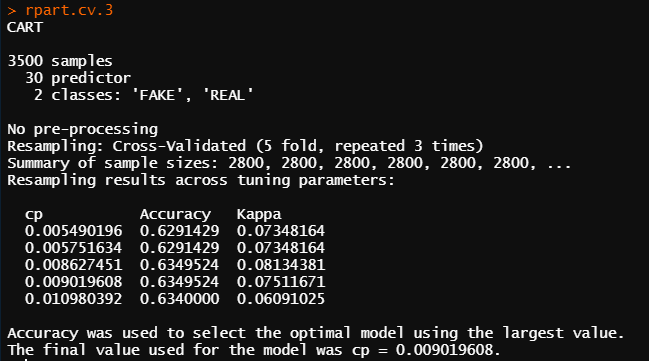


**Cross Validation 2: after adding the TF-IDF weighted vectors**

**Text

Description automatically generated**

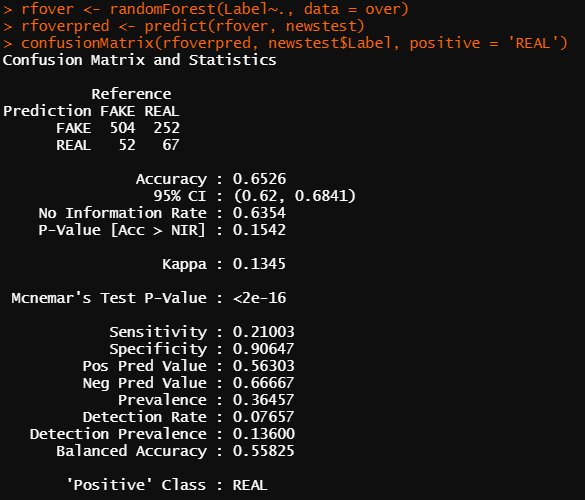
**Cross Validation 3: after adding the Singular Value Decomposition**

****

## Appendix 8 – Class Imbalance Results

**Over-sampling**

* **Confusion Matrix:**



* **Variable Importance Plot for Over Sampling Method**

**A picture containing table

Description automatically generated**

**Under-sampling**

* **Confusion Matrix**

**Text

Description automatically generated**

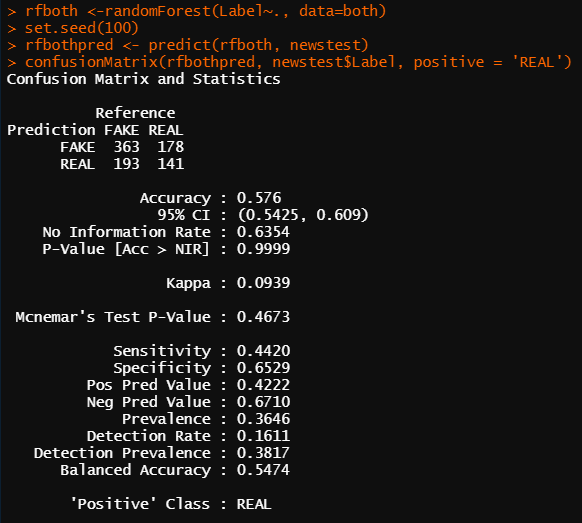
* **Variable Importance Plot for under sampling method:**

**A picture containing table

Description automatically generated**

**Both Sampling Methods**

* **Confusion Matrix**

****

* **Variable Importance Plot for both sampling method**

**A picture containing chart

Description automatically generated**

# References

stevanovicigor 2018, *Fake news on internet in modern digital age stock photo*, viewed 12 September, 2022, <https://www.istockphoto.com/photo/fake-news-on-internet-in-modern-digital-age-gm987914054-267889904>.

Swinburne Online 2022, *7.2 Introduction to text mining and NLP*, viewed 12 September, 2022, <https://swinburneonline.instructure.com/courses/3892/pages/7-dot-2-introduction-to-text-mining-and-nlp?module\_item\_id=320190>.

Seif, G 2022, *An Introductory Guide to NLP for Data Scientists with 7 Common Techniques - KDnuggets*, viewed 12 September, 2022, <https://www.kdnuggets.com/2020/01/intro-guide-nlp-data-scientists.html>.

Techvidvan 2022, *NLP Techniques in Data Science with Real Life Case Studies*, viewed 12 September, 2022, <https://techvidvan.com/tutorials/nlp-techniques-in-data-science/>.

RS, A 2019, *How to create unigrams, bigrams and n-grams of App Reviews | R-bloggers*, viewed 25 September, 2022, <https://www.r-bloggers.com/2019/08/how-to-create-unigrams-bigrams-and-n-grams-of-app-reviews/>.

Zhang, Z 2022, *Text Mining for Social and Behavioral Research Using R*, viewed 26 September, 2022, <https://books.psychstat.org/textmining/sentiment-analysis.html>.

Silge, J 2018, *Sentiment analysis (AFINN) in R*, viewed 25 September, 2022, <https://stackoverflow.com/questions/50200788/sentiment-analysis-afinn-in-r>.

Verma, Y 2022, *A Guide to Term-Document Matrix with Its Implementation in R and Python*, viewed 25 September, 2022, <https://analyticsindiamag.com/a-guide-to-term-document-matrix-with-its-implementation-in-r-and-python/>.

DataScienceDojo 2017, *Text Analytics with R: TF-IDF*, viewed 16 September, 2022, <https://online.datasciencedojo.com/course/text-analytics-with-r/model-building-and-evaluation/tf-idf>.

DataScienceDojo 2017, *Text Analytics with R: SVD with R*, viewed 21 September, 2022, <https://online.datasciencedojo.com/course/text-analytics-with-r/model-building-and-evaluation/svd-with-r>.

Zheng, A & Casari, A 2015, *Feature Engineering for Machine Learning*, viewed 26 September, 2022, <https://www.oreilly.com/library/view/feature-engineering-for/9781491953235/ch04.html>.

DataScienceDojo 2022, *Text Analytics with R: Sample Model Building*, viewed 21 September, 2022, <https://online.datasciencedojo.com/course/text-analytics-with-r/model-building-and-evaluation/sample-model-building>.

DataScienceDojo, D 2017, *Text Analytics with R: Cosine Similarity*, viewed 22 September, 2022, <https://online.datasciencedojo.com/course/text-analytics-with-r/model-building-and-evaluation/cosine-similarity>.

Wikipedia 2022, *Sensitivity and specificity - Wikipedia*, viewed 26 September, 2022, <https://en.wikipedia.org/wiki/Sensitivity\_and\_specificity>.

finnstats 2021, *Class Imbalance-Handling Imbalanced Data in R | R-bloggers*, viewed 14 September, 2022, <https://www.r-bloggers.com/2021/05/class-imbalance-handling-imbalanced-data-in-r/>.

Bhalla, D 2014, *A complete guide to Random Forest in R*, viewed 17 September, 2022, <https://www.listendata.com/2014/11/random-forest-with-r.html#id-56fce3>.

Johnson, D 2022, *Decision Tree in R: Classification Tree with Example*, viewed 25 September, 2022, <https://www.guru99.com/r-decision-trees.html#5>.