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

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.

1.1 Data Science Pipeline

The pipeline for this project included these distinct steps:

1. Activate Required Libraries
2. Import Data
3. Explore and Transform Data

3.1 Missing value, duplicate and null label removal

3.2 Compare Text Length and Word Count by Label

3.2.1 Visualize features with Histogram and Boxplot

3.3 Subset & Shuffle Data

4. Natural Language Processing

4.1 Corpus formation and transformation

4.2 Document Term Frequency Matrix

4.3 Data Partition into Train & Test Sets

4.4 Create corpus of fake and real terms

4.5 Visualize frequent terms with word clouds and bar plots

4.6 Explore Bigrams and Word Associations

4.7 Sentiment Analysis

4.8 Cross Validation

5. TF-IDF & Cross Validation

5.1 Bar plots of Frequent & TF-IDF Weighted Terms by Label

6. Singular Value Decomposition & Cross Validation
7. Model Building & Evaluation

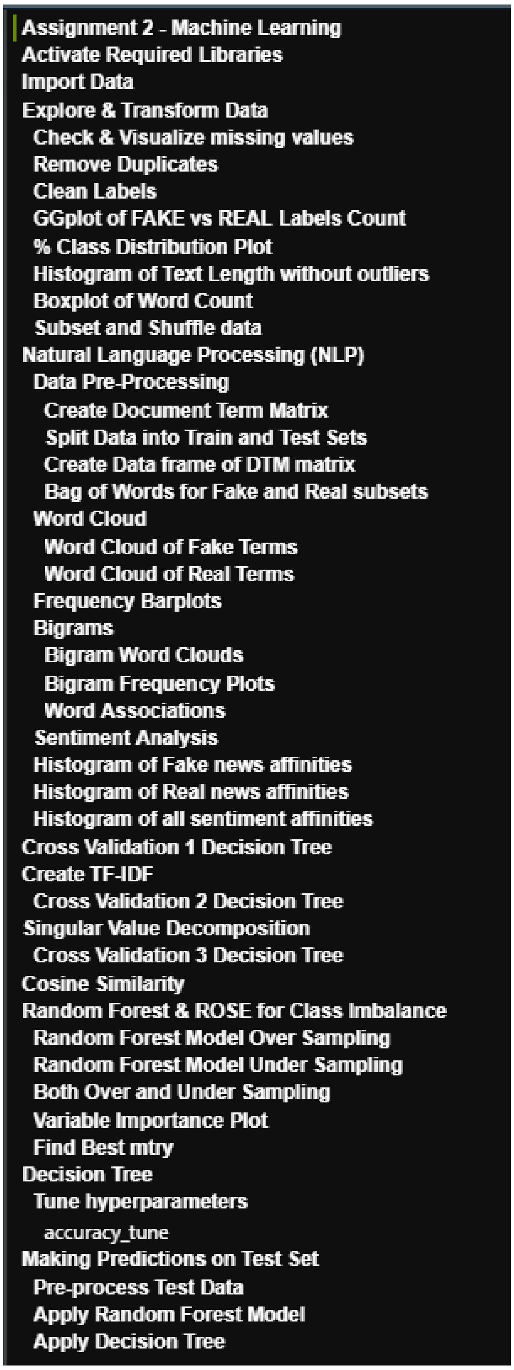
7.1 Random Forest Model + ROSE Class imbalance

7.1.1 Evaluate & Finetune Random Forest Model

7.2 Decision Tree

7.2.1 Evaluate and Finetune Decision Tree

8. Make Predictions on Test Data with Random Forest Model


Assignment 2 - Machine Learning
Activate Required Libraries
Import Data
Explore & Transform Data
Check & Visualize missing values
Remove Duplicates
Clean Labels
GGplot of FAKE vs REAL Labels Count
% Class Distribution Plot
Histogram of Text Length without outliers
Boxplot of Word Count
Subset and Shuffle data
Natural Language Processing (NLP)
Data Pre-Processing
Create Document Term Matrix
Split Data into Train and Test Sets
Create Data frame of DTM matrix
Bag of Words for Fake and Real subsets
Word Cloud
Word Cloud of Fake Terms
Word Cloud of Real Terms
Frequency Barplots
Bigrams
Bigram Word Clouds
Bigram Frequency Plots
Word Associations
Sentiment Analysis
Histogram of Fake news affinities
Histogram of Real news affinities
Histogram of all sentiment affinities
Cross Validation 1 Decision Tree
Create TF-IDF
Cross Validation 2 Decision Tree
Singular Value Decomposition
Cross Validation 3 Decision Tree
Cosine Similarity
Random Forest & ROSE for Class Imbalance
Random Forest Model Over Sampling
Random Forest Model Under Sampling
Both Over and Under Sampling
Variable Importance Plot
Find Best mtry
Decision Tree
Tune hyperparameters
accuracy_tune
Making Predictions on Test Set
Pre-process Test Data
Apply Random Forest Model
Apply Decision Tree

2. Natural Language Processing

2.1 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.

```
# Import Data-----------------------------------------------------------
news <- read.csv (file.choose(), header = T, sep = ";")
```

```
> as_tibble(news)
# A tibble: 11,507 x 5
   Text                                                 Text_…¹ Author Date  Label
   <chr>                                                <chr>   <chr>  <chr> <chr>
 1 "Says the Annies List political group supports…      aborti… Ilsa … 2017… FAKE
 2 "When did the decline of coal start? It starte…      energy… Esmer… 2017… FAKE
 3 "Hillary Clinton agrees with John McCain \"by …      foreig… Mar H… 2017… REAL
 4 "Health care reform legislation is likely to m…      health… Lilli… 2017… FAKE
 5 "The economic turnaround started at the end of…      econom… Flemi… 2017… FAKE
 6 "The Chicago Bears have had more starting quar…      educat… Tyron… 2017… REAL
 7 "Jim Dunnam has not lived in the district he r…      candid… Hurle… 2018… FAKE
 8 "I'm the only person on this stage who has wor…      ethics  Warin… 2018… FAKE
 9 "However, it took $19.5 million in Oregon Lott…      jobs    Berge… 2017… FAKE
10 "Says GOP primary opponents Glenn Grothman and…      energy… Daisi… 2017… REAL
# … with 11,497 more rows, and abbreviated variable name ¹Text_Tag
```

2.2 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.

```
> #check labels
> news_0 %>% group_by(Label) %>% summarise(count=n())
# A tibble: 3 x 2
  Label  count
  <chr>  <int>
1 ""      1268
2 "FAKE"  6594
3 "REAL"  3637
```

```
> subset(news,duplicated(news))
      Text Text_Tag Author Date Label
11500
11501
11502
11503
11504
11505
11506
11507
```

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

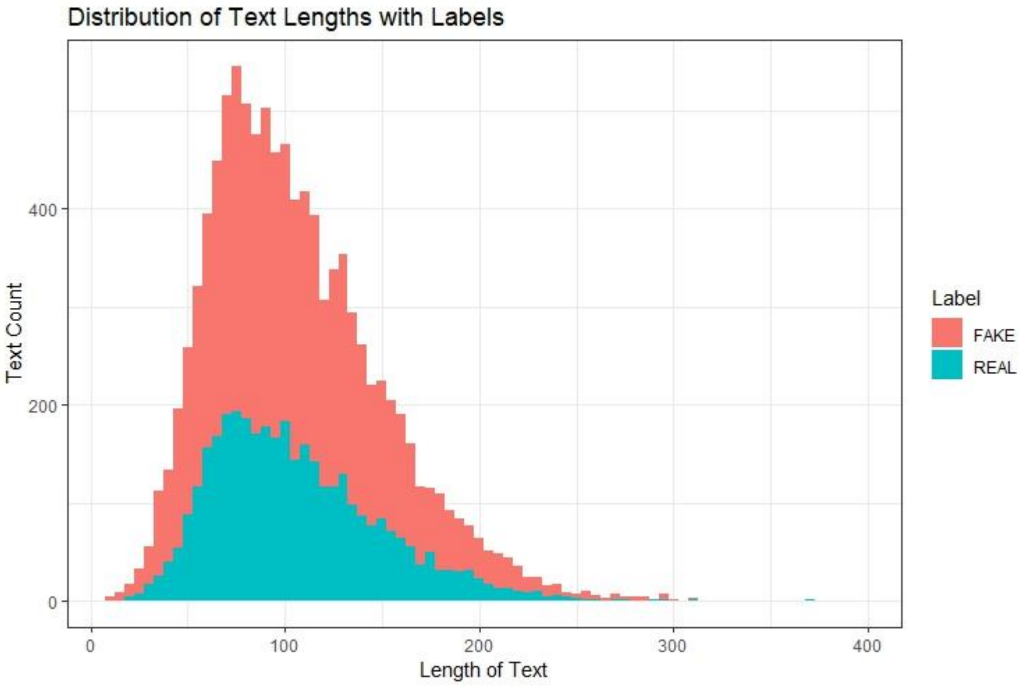
A new column was created calculating length of text and it was found that the mean text length is longer for real news:

```
[1] "Text Length for Fake:"
> summary(newsfake$Length)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   11.0    73.0    99.0   106.6   133.0  2098.0 
> print('Text Length for Real News:')
[1] "Text Length for Real News:"
> summary(newsreal$Length)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   11.0    73.0    99.0   107.7   131.0  3189.0 
```

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

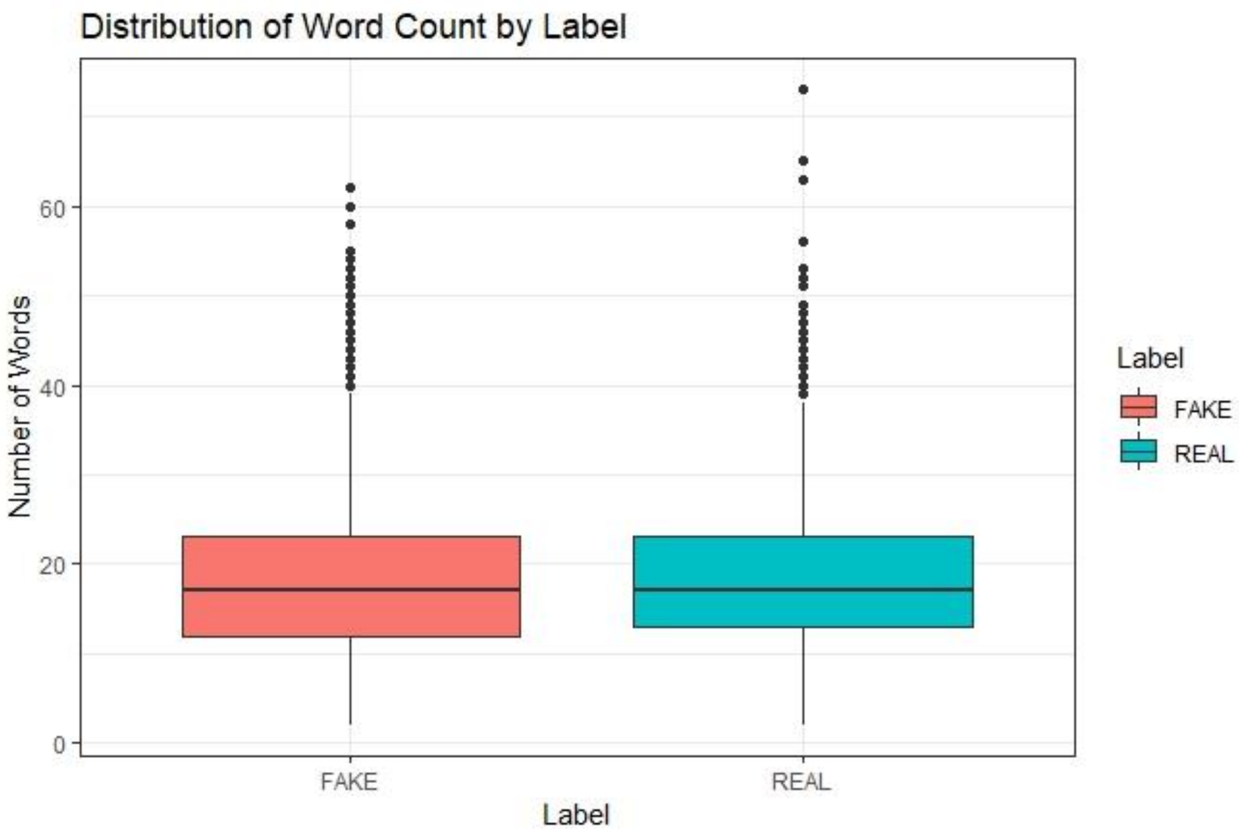
```
                       Text_Tag                Author       Date Label Length
1281 elections,ethics,states        Justin Ciciura 2018/02/23  REAL   3189
2142                                  Stacy Clears 2017/12/20  FAKE    538
6115                      crime        Jorry Poore 2018/01/20  REAL   1587
7544 stimulus,transportation Brooks Swinnerton 2017/11/05  FAKE   2098
```

Outliers 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:



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:

```
#Subset and Shuffle data-----------------------------------------------------------------
set.seed (2022)
news_3 <- news_2[order(runif(n=5000)),]

#convert to data frame
as.data.frame(news_3)
#reset index
rownames(news_3) <- NULL
```

```
> as_tibble(news_3)
# A tibble: 5,000 × 7
   Text                                                 Text_…¹ Author Date  Label Length Words
   <chr>                                                <chr>   <chr>  <chr> <fct>  <int> <int>
 1 Says Edward Snowden could have gotten all of the …   foreig… Kaia … 2018… FAKE      86    14
 2 Secretary Clinton changes her position on (gun is…   candid… Engle… 2018… FAKE     201    31
 3 Only 18 percent of jobs are accessible by transit…   econom… Roobb… 2017… REAL     119    21
 4 After (Jeb) Bushs two terms in office, Floridas g…   educat… Flori… 2018… REAL     106    18
 5 In 2009, his first year as mayor, Julin Castro re…   campai… Binky… 2018… FAKE     195    33
 6 Atlanta now has as many visitors as Las Vegas, Sa…   tourism Sean … 2018… REAL     132    24
 7 China owns more of our bonds than do Americans.      china,… Esmer… 2017… FAKE      47     9
 8 For the first time in history, the share of the n…   electi… Quint… 2017… REAL     116    21
 9 I lost my health insurance and my doctor because …   health… Betty… 2017… FAKE      62    11
10 By voting to approve [Question 1], we can . . . s…   econom… Minnn… 2018… FAKE      76    14
# … with 4,990 more rows, and abbreviated variable name ¹Text_Tag
```

2.3 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.

```
## Create Corpus of Text column
corpus <- Corpus (VectorSource (news_3$Text))

### Change all words to lowercase
news.corpus <- tm_map (corpus, content_transformer (tolower))
### Remove Stop Words
news.corpus <- tm_map (news.corpus, removeWords, stopwords("english"))
### Remove Punctuation
news.corpus <- tm_map (news.corpus, removePunctuation)
### Remove Numbers
news.corpus <- tm_map (news.corpus, removeNumbers)
### Strip white space
news.corpus <- tm_map (news.corpus, stripWhitespace)

### Stem words using SnowballC library
news.corpus <- tm_map (news.corpus, stemDocument)
```

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.

```
> ## Create Document Term Matrix ----------------------------------------
> news.corpus.dtm <- DocumentTermMatrix (news.corpus)
> news.corpus.dtm = removeSparseTerms(news.corpus.dtm, 0.999)
> inspect(news.corpus.dtm)
<<DocumentTermMatrix (documents: 5000, terms: 1483)>>
Non-/sparse entries: 41674/7373326
Sparsity           : 99%
Maximal term length: 15
Weighting          : term frequency (tf)
Sample             :
      Terms
Docs   job million obama percent presid say state tax vote year
  1373   2       0     0       0      0   0     0   1    0    1
  166    0       1     0       0      0   0     0   0    0    1
  1793   0       0     0       1      0   1     2   1    1    0
  1862   0       0     0       0      0   1     0   2    0    0
  1945   2       0     0       0      0   0     0   1    0    1
  1968   0       0     0       0      0   1     0   0    0    1
  2369   0       0     0       0      0   3     0   0    0    0
  4112   4       0     0       0      0   0     0   0    0    0
  79     0       0     0       0      0   0     0   0    0    0
  980    0       0     0       0      0   0     0   0    0    0
```

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

```
## Split Data into Train and Test Sets----------------------------------
#split raw data
n <- nrow (news_3)
raw.news.train <- news_3 [1:round(.7 * n),]
raw.news.test  <- news_3 [(round(.7 * n)+1):n,]

#split corpus
nn <- length (news.corpus)
news.corpus.train <- news.corpus [1:round(.7 * nn)]
news.corpus.test  <- news.corpus [(round(.7 * nn)+1):nn]

#split dtm
nnn <- nrow (news.corpus.dtm)
news.corpus.dtm.train <- news.corpus.dtm[1:round(.7 * nnn),]
news.corpus.dtm.test  <- news.corpus.dtm[(round(.7 * nnn)+1):nnn,]
```

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:

```
newsdf <- as.data.frame(as.matrix(news.corpus.dtm.train))
#append labels to dtm
newsdf <- cbind(Label=raw.news.train$Label, newsdf)
#make column names syntactically valid
colnames(newsdf) <- make.names(colnames(newsdf))
as_tibble(newsdf)
```

The training data frame now looks like this:

```
# A tibble: 3,500 x 1,484
   Label gotten protect   say campaign chang clinton  dont elect everi feder   gun
   <fct>  <dbl>   <dbl> <dbl>    <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 FAKE       1       1     1        0     0       0     0     0     0     0     0
 2 FAKE       0       0     1        1     1       1     1     1     1     1     1
 3 REAL       0       0     0        0     0       0     0     0     0     0     0
 4 REAL       0       0     0        0     0       0     0     0     0     0     0
 5 FAKE       0       0     0        0     0       0     0     0     0     0     0
 6 REAL       0       0     0        0     0       0     0     0     0     0     0
 7 FAKE       0       0     0        0     0       0     0     0     0     0     0
 8 REAL       0       0     0        0     0       0     0     0     0     0     0
 9 FAKE       0       0     0        0     0       0     0     0     0     0     0
10 FAKE       0       0     0        0     0       0     0     0     0     0     0
```

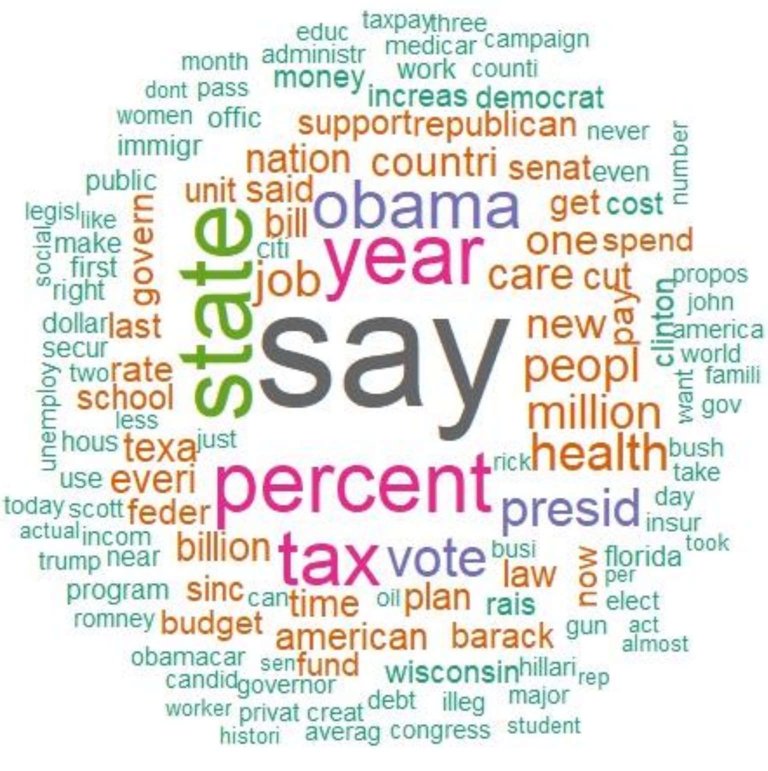
The raw training data was also split into Fake and Real subsets and underwent the corpus cleaning process (see Appendix 3 for document term matrices summary).

2.4 Data Visualization

2.4.1 Word Clouds

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

```
## Word Cloud---------------------------------------------------------
wordcloud(news.corpus.train, min.freq = 50, random.order = FALSE,
          colors=brewer.pal (8, "Dark2"), scale=c(6,.5))
```

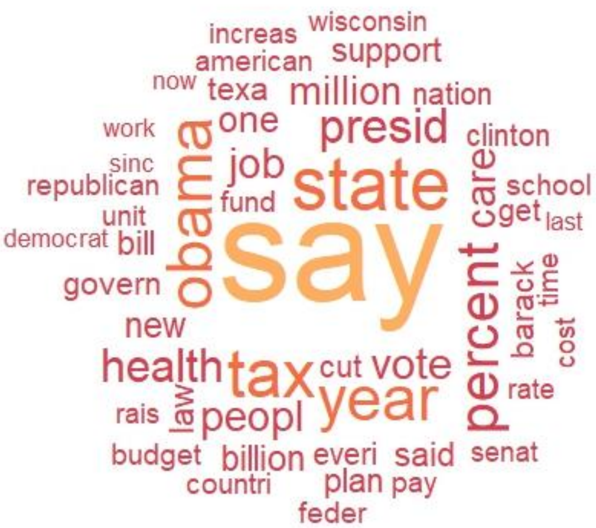


The most prominent word is ‘say’, and it seems that most words have political connotations, including the names of US politicians.

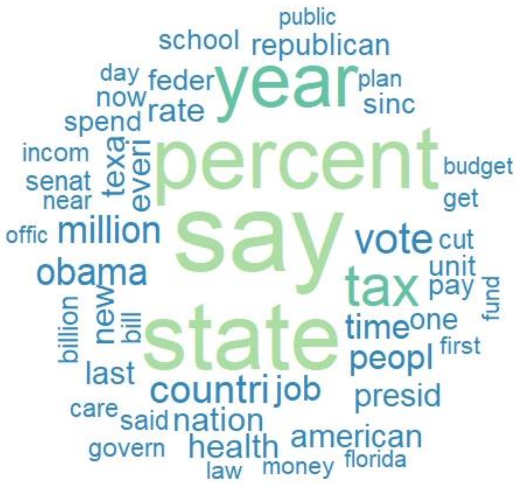
The word cloud code was adjusted to use specific colours from the Spectral brewer palette of the RColorBrewer package. There were also a lot of technical issues with RStudio in creating the word clouds (see Appendix 4).

```
wordcloud (clean.corpus_fake, max.words = 50,
           random.order = FALSE, colors = brewer.pal(8, "Spectral")[1:3],
           scale=c(6,.5))
```

Word Cloud of Fake Terms



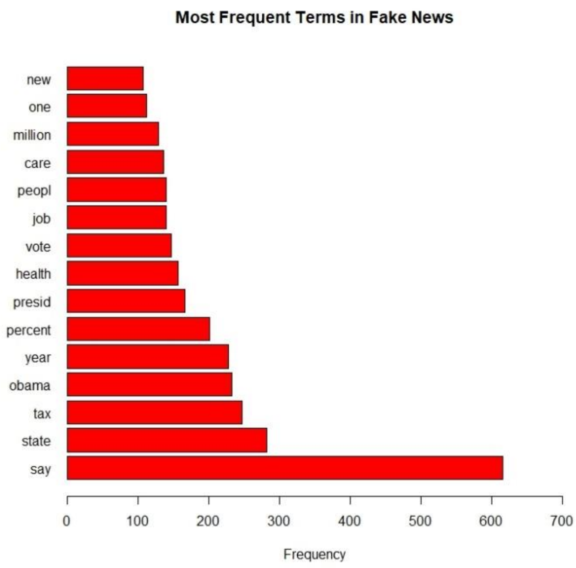
Word Cloud of Real Terms

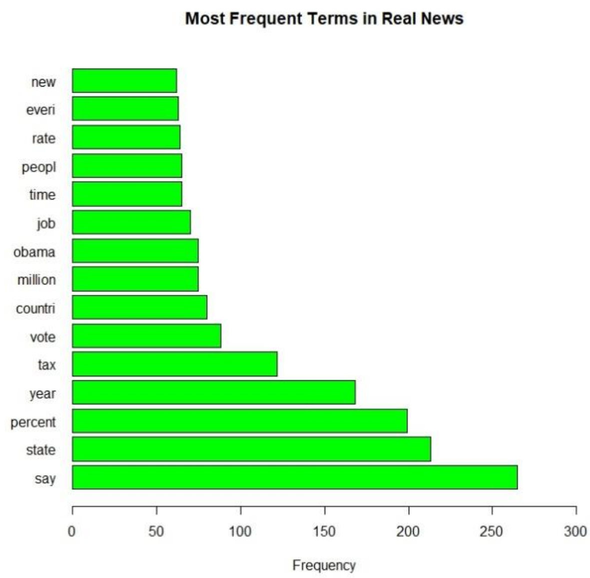


From the word clouds, it can be deduced that real news has more emphasis on ‘percent’ which probably means the articles use a lot more statistical evidence, whereas fake news has an emphasis on ‘obama,’ so fake news tends to spread more false information about people.

2.4.2 Bar Plots of Term Frequency

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





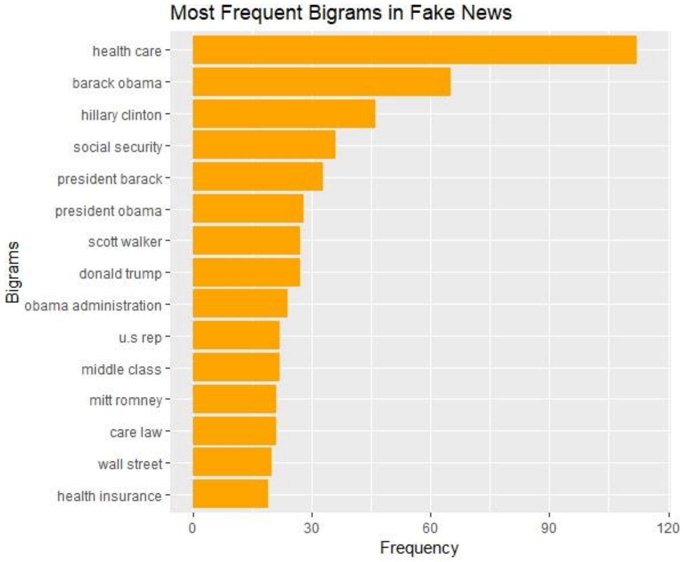
Between the two labels, 11 out of 15 top terms are shared. Terms unique to fake news include ‘presid’, ‘health’, ‘care’, and ‘one’.

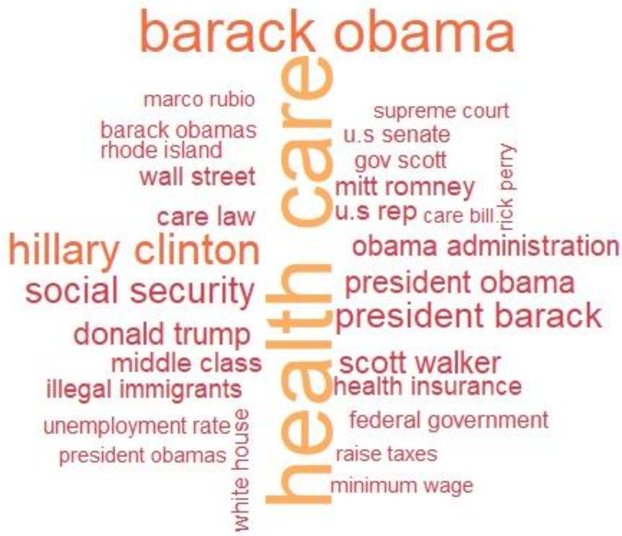
2.5 Bigrams

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

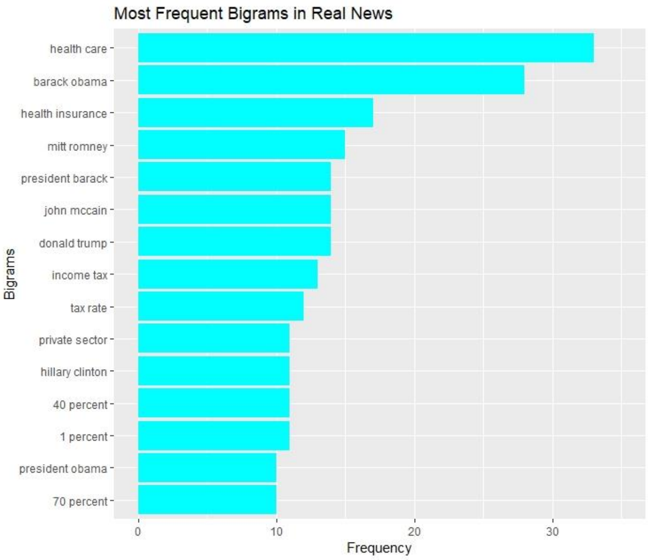
```
fakebigrams <- as.data.frame(fake %>%
  unnest_tokens(word, Text, token = "ngrams", n = 2) %>%
  separate(word, c("word1", "word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  unite(word,word1, word2, sep = " ") %>%
  count(word, sort = TRUE))
```

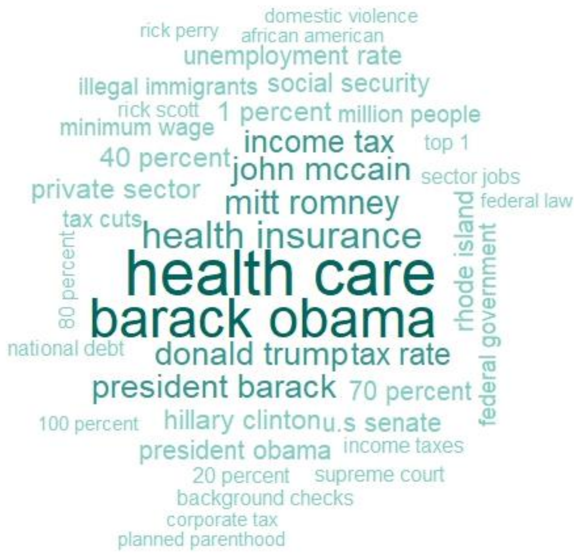
Bigrams in Fake News





Bigrams in Real News



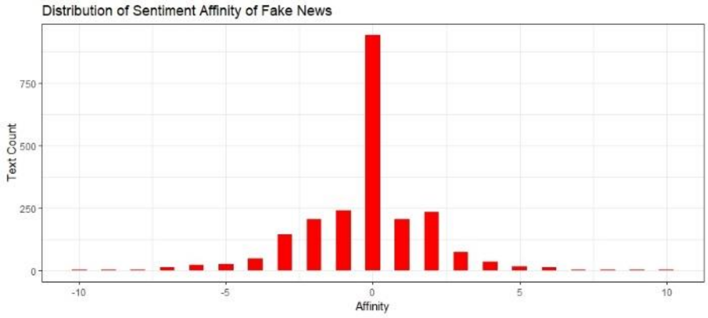


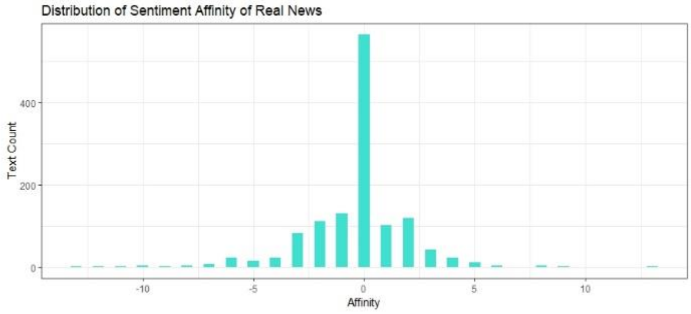
As hypothesized, real news articles include more numerical values and statistics, with three top bigrams including 1, 40 and 70 percent. While 7 out of the 15 top bigrams in real news are people's names, 8 of the 15 in fake news includes names, with one bigram 'obama administration' not seen in any of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

2.6 Sentiment Analysis

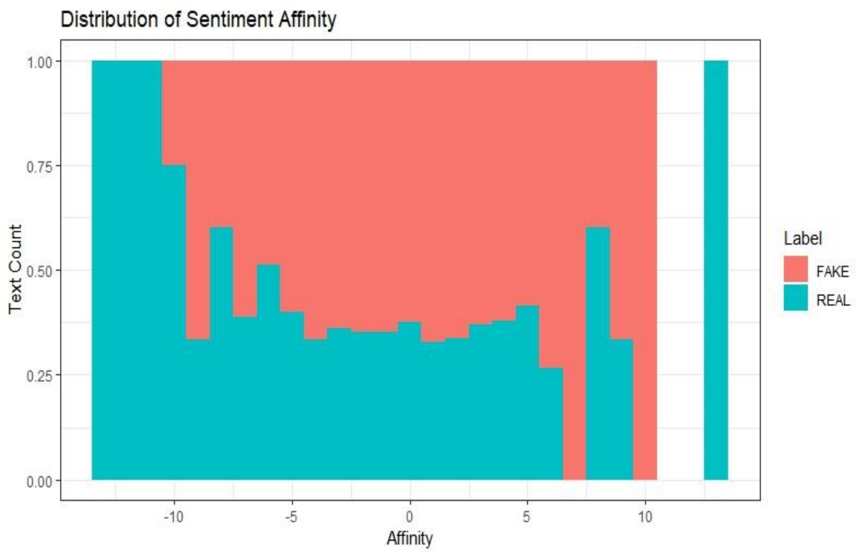
The ‘textdata’ package contains a lexicon known as ‘affin’ that assigns words values between -5 and 5 according to whether the sentiment is negative or positive (Zhang 2022). The overall sentiment can be calculated by adding all the values of each individual word together, which was applied to the training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

```
# Fake sentiments
#add index column
fake$index <- 1:nrow(fake)
#calculate score for each token in text
fake_sentiment <- fake %>%
  unnest_tokens(word, Text, token = "words") %>%
  inner_join(get_sentiments("afinn"))

#add scores together
fake_sentiments <- fake %>% left_join(fake_sentiment %>%
                  group_by(index) %>%
                  summarise(score = sum(value))) %>%
  replace_na(list(score = 0))

#remove index column
fake_sentiments <- fake_sentiments[-3]
as_tibble(fake_sentiments)
summary(fake_sentiments)
```





It 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

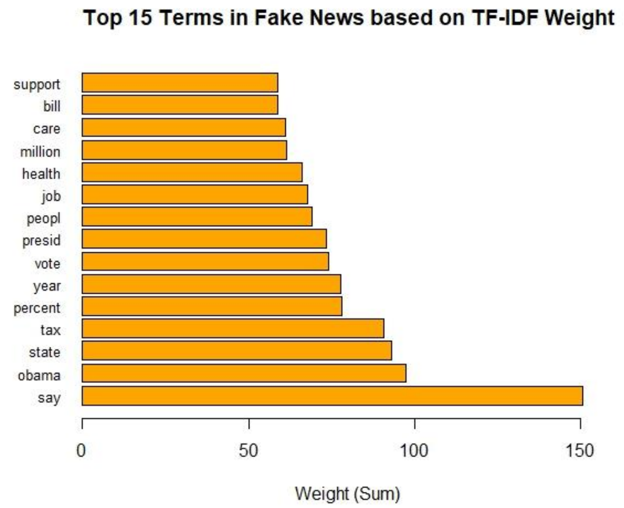
TF-IDF normalizes the frequency of a term across documents so that more common words have a lower score and rarer words have a higher score, hence it is a useful way to extract keywords from a text (Zheng & Casari 2015).

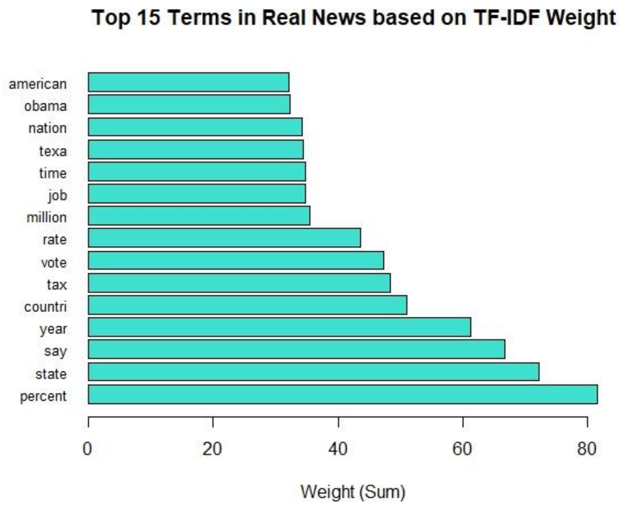
The TF-IDF was created using the weightTfIdf function included in the tm package:

```
# Create TF-IDF----------------------------------------
tfidf <- weightTfIdf(news.corpus.dtm.train)
news.tfidf <- as.data.frame(as.matrix(tfidf))

#append labels
news.tfidf <- cbind(Label=newsdf$Label, news.tfidf)

as_tibble(news.tfidf)
```

```
> as_tibble(news.tfidf)
# A tibble: 3,500 x 1,484
   Label gotten protect   say campa…¹ chang clinton  dont elect everi feder   gun  issu  need
   <fct>  <dbl>   <dbl> <dbl>   <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 FAKE    2.99    2.44 0.679   0     0       0     0     0     0     0     0     0     0
 2 FAKE    0       0    0.102   0.300 0.326   0.259 0.317 0.289 0.233 0.245 0.296 0.369 0.339
 3 REAL    0       0    0       0     0       0     0     0     0     0     0     0     0
 4 REAL    0       0    0       0     0       0     0     0     0     0     0     0     0
 5 FAKE    0       0    0       0     0       0     0     0     0     0     0     0     0
 6 REAL    0       0    0       0     0       0     0     0     0     0     0     0     0
 7 FAKE    0       0    0       0     0       0     0     0     0     0     0     0     0
 8 REAL    0       0    0       0     0       0     0     0     0     0     0     0     0
 9 FAKE    0       0    0       0     0       0     0     0     0     0     0     0     0
10 FAKE    0       0    0       0     0       0     0     0     0     0     0     0     0
# … with 3,490 more rows, 1,470 more variables: obama <dbl>, one <dbl>, posit <dbl>,
```

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:





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:

```
tfidf_m <- as.matrix(news.tfidf[-1]) #convert news.tfidf to matrix
as_tibble(tfidf_m)
#use library irlba to perform SVD for LSA
train.irlba <- irlba(t(tfidf_m), nv = 30, maxit = 300)
summary(train.irlba$v)
```

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:

```
#project new data into SVD semantic space
sigma.inverse <- 1/train.irlba$d #d maps to sigma
u.transpose <- t(train.irlba$u) #take u matrix and transpose
document <- tfidf_m[1,] #take first document of tfidf
document.hat <- sigma.inverse * u.transpose %*% document #multiply 3 above together
```

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

```
# create new feature data frame using document semantic space of 30 features
news.svd <- data.frame(Label = news.tfidf$Label, train.irlba$v)
as_tibble(news.svd)
```

```
# A tibble: 3,500 × 31
   Label      X1      X2       X3        X4       X5        X6       X7        X8
   <fct>   <dbl>   <dbl>    <dbl>     <dbl>    <dbl>     <dbl>    <dbl>     <dbl>
 1 FAKE  -0.0149  0.0126  0.00120  0.0125     0.00665 -0.00351  -0.00672 -0.00870
 2 FAKE  -0.0172  0.0112 -0.0130   0.00410   -0.00239  0.0127    0.00692 -0.000565
 3 REAL  -0.0195 -0.0293 -0.00888 -0.0194     0.00786 -0.00593   0.00193  0.0298
 4 REAL  -0.0145 -0.0178 -0.0101   0.00330    0.00341  0.00356   0.00436 -0.00781
 5 FAKE  -0.0135  0.00308 -0.00987 0.0000381 -0.00267  0.00669   0.00484  0.00514
 6 REAL  -0.0111 -0.00731  0.00120 0.00345   -0.00197  0.00588   0.00155  0.00491
 7 FAKE  -0.0126 -0.00576 -0.00846 -0.00905   0.00218 -0.000806  0.0134   0.0176
 8 REAL  -0.0180  0.0106 -0.00712  0.0146    -0.00131 -0.0145   -0.0303   0.00534
 9 FAKE  -0.0198  0.0178  0.0464  -0.0591    -0.0446   0.00278   0.0119  -0.00212
10 FAKE  -0.0193  0.00424  0.00440  0.0135   -0.0103  -0.00314  -0.0185   0.0383
# … with 3,490 more rows, and 22 more variables: X9 <dbl>, X10 <dbl>, X11 <dbl>,
```

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.

3. 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.

```
# ROSE for Class Imbalance-----------------------------------
#scale numeric variables
news.svd[,-1] <- scale(news.svd[,-1])

## Split data  using caTools
set.seed(100)
newssplit <- sample.split(news.svd$Label, SplitRatio = 0.75)
newstrain <-  subset (news.svd, newssplit == TRUE)
newstest <- subset (news.svd, newssplit == FALSE)
```

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

```
> set.seed(500)
> rftrain <- randomForest(Label~., data = newstrain, importance=TRUE)
> rftrain

Call:
 randomForest(formula = Label ~ ., data = newstrain, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 36.3%
Confusion matrix:
     FAKE REAL class.error
FAKE 1485  184   0.1102457
REAL  769  187   0.8043933
```

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:

```
> #Confusion Matrix
> set.seed(100)
> rfpred <- predict(rftrain, newstest)
> confusionMatrix(rfpred, newstest$Label, positive = 'REAL')
Confusion Matrix and Statistics

          Reference
Prediction FAKE REAL
      FAKE  508  266
      REAL   48   53

               Accuracy : 0.6411
                 95% CI : (0.6084, 0.673)
    No Information Rate : 0.6354
    P-Value [Acc > NIR] : 0.3771

                  Kappa : 0.0934

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.16614
            Specificity : 0.91367
         Pos Pred Value : 0.52475
         Neg Pred Value : 0.65633
             Prevalence : 0.36457
         Detection Rate : 0.06057
   Detection Prevalence : 0.11543
      Balanced Accuracy : 0.53991

       'Positive' Class : REAL
```

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%

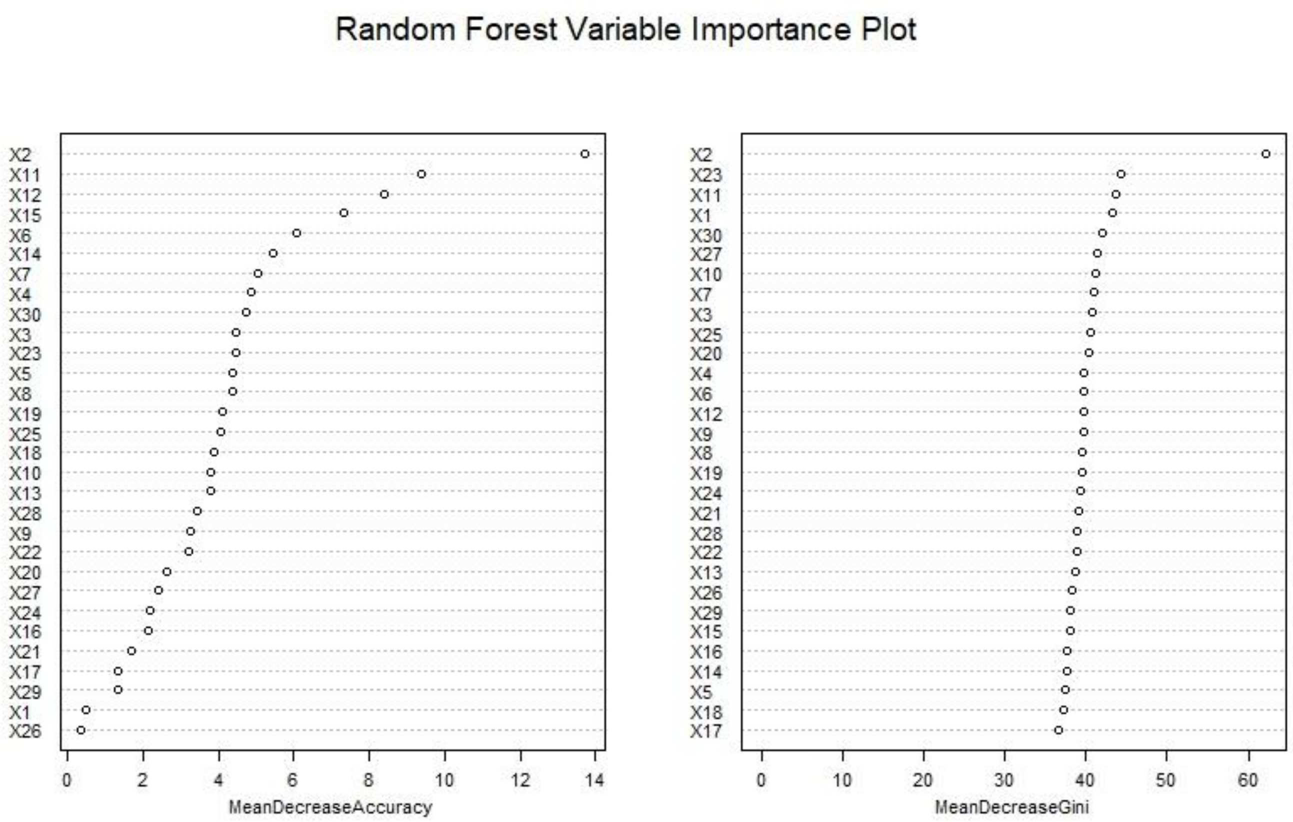
- The bolded cells represent the best results. It seems that over sampling has better accuracy and lower estimated OOB error rate, however, under-sampling has a lower false negative rate, and the best precision.

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:

```
## Variable Importance Plot------------------
importance(rftrain)
varImpPlot(rftrain, scale=TRUE, cex = 0.7,
           main = 'Random Forest Variable Importance Plot')
```

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

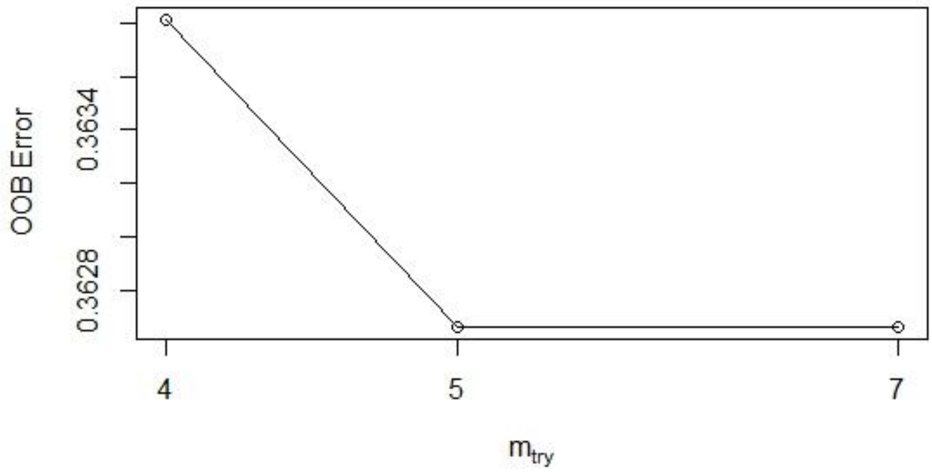


The top variables that influence accuracy are X2, X11, X12 and X15.

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).

```
> set.seed(100)
> mtry <- tuneRF(newstrain[-1], newstrain$Label, ntreeTry=500,
+                stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
mtry = 5  OOB error = 36.27%
Searching left ...
mtry = 4        OOB error = 36.38%
-0.003151261 0.01
Searching right ...
mtry = 7        OOB error = 36.27%
0 0.01
> #find best mtry
> best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
> print(mtry)
      mtry  OOBError
4.OOB    4 0.3638095
5.OOB    5 0.3626667
7.OOB    7 0.3626667
> print(best.m)
5.OOB 7.OOB
    5     7
```



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

```
#Build model again with best mtry value
rf <- randomForest(Label~., data = newstrain, mtry=best.m, importance=TRUE, ntree=500)
rf
```

```
Call:
 randomForest(formula = Label ~ ., data = newstrain, mtry = best.m,
     importance = TRUE, ntree = 500)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 36.3%
Confusion matrix:
     FAKE REAL class.error
FAKE 1495  174   0.1042540
REAL  779  177   0.8148536
```

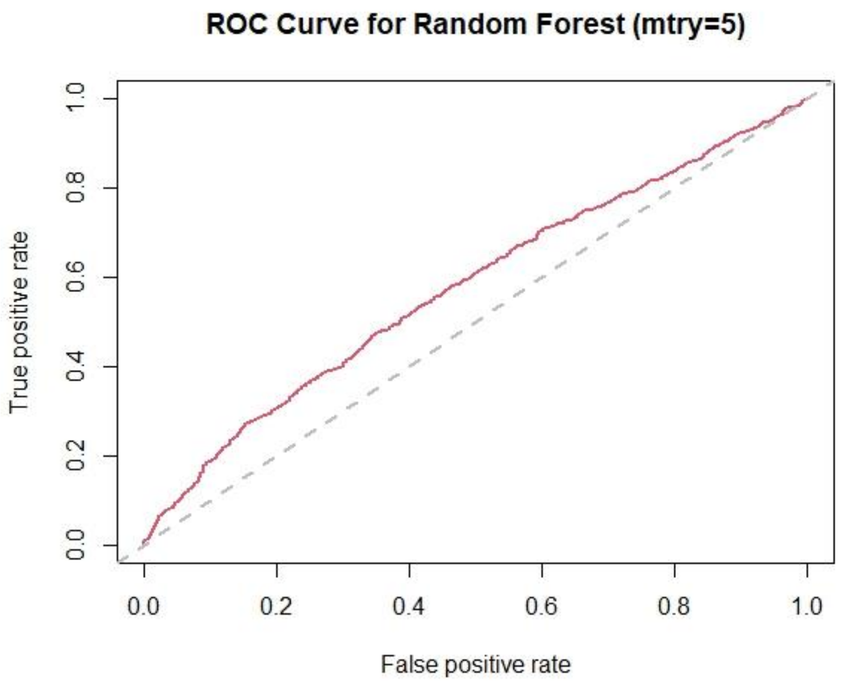
The default mtry value was also 5, and the difference from the first random forest model is that the class error for fake news decreased, while the class error for classifying real news increased.

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

```
Call:
 randomForest(formula = Label ~ ., data = newstrain, mtry = 7,
      importance = TRUE, ntree = 500)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 7

        OOB estimate of  error rate: 36.34%
Confusion matrix:
     FAKE REAL class.error
FAKE 1476  193   0.1156381
REAL  761  195   0.7960251
```

3.2.2 ROC Curve

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

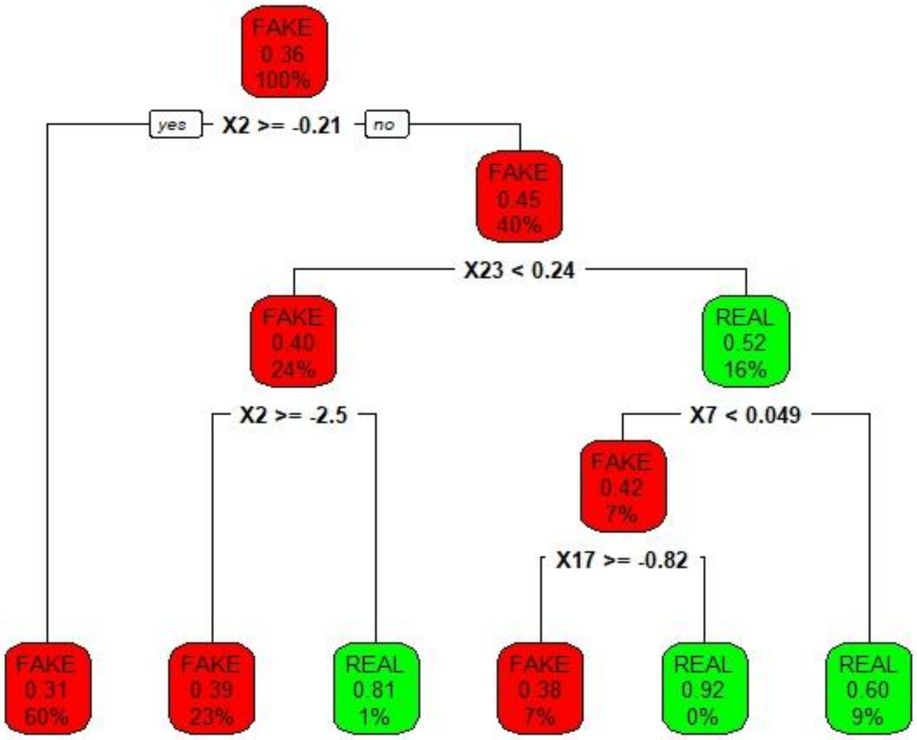


Ideally, the curve would be further away from the midline, however, this model is not an ideal model for classifying fake and real news.

4. Decision Tree Model

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

```
# Decision Tree -----------------------------------------
#create decision tree using rpart
set.seed(100)
dt <- rpart(Label~., data = newstrain, method = 'class')

#plot tree using rpart.plot
rpart.plot(dt, extra = 106, box.palette=c('red', 'green'))
```



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:

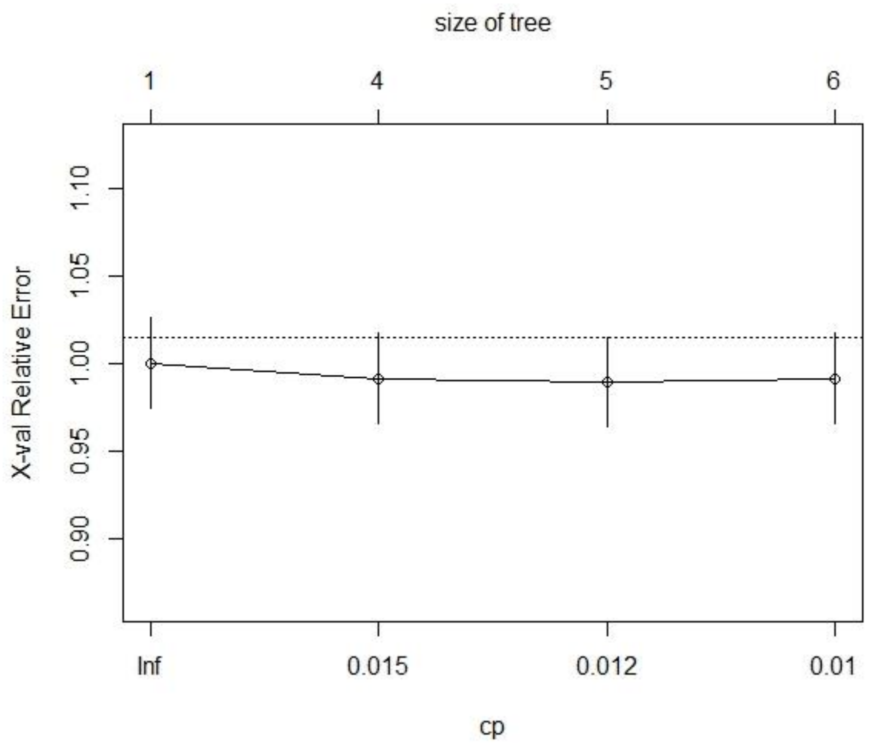
```
#Evaluate Decision Tree
predict_class <- predict(dt, newstest, type = 'class')
confusionMatrix(data=predict_class, reference=newstest$Label,
                positive='REAL')
```

```
> confusionMatrix(data=predict_class, reference=newstest$Label,
 positive='REAL')
Confusion Matrix and Statistics

          Reference
Prediction FAKE REAL
      FAKE  511  279
      REAL   45   40

               Accuracy : 0.6297
                 95% CI : (0.5968, 0.6618)
    No Information Rate : 0.6354
    P-Value [Acc > NIR] : 0.6514

                  Kappa : 0.0527

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.12539
            Specificity : 0.91906
```

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.

4.1 Tuning the Decision Tree Model

The 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.

```
Classification tree:
rpart(formula = Label ~ ., data = newstrain, method = "class")

Variables actually used in tree construction:
[1] X17 X2  X23 X7

Root node error: 956/2625 = 0.36419

n= 2625

        CP nsplit rel error  xerror     xstd
1 0.016039      0   1.00000 1.00000 0.025789
2 0.013598      3   0.95188 0.99163 0.025742
3 0.010460      4   0.93828 0.98954 0.025731
4 0.010000      5   0.92782 0.99163 0.025742
There were 21 warnings (use warnings() to see them)
> print(dt)
n= 2625

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2625 956 FAKE (0.63580952 0.36419048)
   2) X2>=-0.2050489 1575 484 FAKE (0.69269841 0.30730159) *
   3) X2< -0.2050489 1050 472 FAKE (0.55047619 0.44952381)
     6) X23< 0.2420731 630 254 FAKE (0.59682540 0.40317460)
      12) X2>=-2.530146 609 237 FAKE (0.61083744 0.38916256) *
      13) X2< -2.530146 21   4 REAL (0.19047619 0.80952381) *
     7) X23>=0.2420731 420 202 REAL (0.48095238 0.51904762)
      14) X7< 0.0488916 184  77 FAKE (0.58152174 0.41847826)
        28) X17>=-0.8196956 172  66 FAKE (0.61627907 0.38372093) *
        29) X17< -0.8196956 12   1 REAL (0.08333333 0.91666667) *
      15) X7>=0.0488916 236  95 REAL (0.40254237 0.59745763) *
```

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

```
#Tune hyperparameters------------------------------------------------
control <- rpart.control(minsplit = 5,
                         maxdepth = 4,
                         cp=0.012)
set.seed(100)
tune_fit <- rpart(Label~., data = newstrain, method = 'class',
                  control = control)


accuracy_tune <- function(dt) {
  predict_dt <- predict(dt, newstest, type = 'class')
  tabledt <- table(newstest$Label, predict_dt)
  accuracy_Test <- sum(diag(tabledt)) / sum(tabledt)
  accuracy_Test
}
accuracy_tune(tune_fit)
```

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%.

```
> accuracy_tune(tune_fit)
[1] 0.632
```

5. 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.

```
> test.tfidf <- weightTfIdf(news.corpus.dtm.test)
Warning messages:
1: In weightTfIdf(news.corpus.dtm.test) : empty document(s): 3544 4363
2: In weightTfIdf(news.corpus.dtm.test) :
  unreferenced term(s): transit onlin subsidi resourc imposs iowa driver werent everybod
i discuss santorum oversea piec implement press port english fourth photo park warren va
st kennedi guy deliv partner ill wont commerc respond reveal duti wealthiest fuel tree j
oin predict faculti rack dealer whole suit
```

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.

```
#Apply SVD projection
test.svd.raw <- t(sigma.inverse * u.transpose %*% t(as.matrix(test.tfidf)))

#add Label column
test.svd <- data.frame(Label = raw.news.test$Label, test.svd.raw)
#class levels
levels(test.svd$Label) <- c("FAKE", "REAL")
```

5.2 Random Forest Model Evaluation

The finetuned random forest was applied to the test data:

```
## Apply Random Forest Model-------------------------------------------
rftestpred <- predict(rf, test.svd)
confusionMatrix(rftestpred, test.svd$Label, positive = 'REAL')
```

```
> confusionMatrix(rftestpred, test.svd$Label, positive = 'REAL')
Confusion Matrix and Statistics

          Reference
Prediction FAKE REAL
      FAKE  978  522
      REAL    0    0

               Accuracy : 0.652
                 95% CI : (0.6273, 0.6761)
    No Information Rate : 0.652
    P-Value [Acc > NIR] : 0.5119

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.000
            Specificity : 1.000
         Pos Pred Value :   NaN
         Neg Pred Value : 0.652
             Prevalence : 0.348
         Detection Rate : 0.000
   Detection Prevalence : 0.000
      Balanced Accuracy : 0.500

       'Positive' Class : REAL
```

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.

```
> dttestpred <- predict(tune_fit, test.svd, type='class')
> confusionMatrix(dttestpred, test.svd$Label, positive = 'REAL')
Confusion Matrix and Statistics

          Reference
Prediction FAKE REAL
      FAKE  978  522
      REAL    0    0

               Accuracy : 0.652
                 95% CI : (0.6273, 0.6761)
    No Information Rate : 0.652
    P-Value [Acc > NIR] : 0.5119

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.000
            Specificity : 1.000
         Pos Pred Value :   NaN
         Neg Pred Value : 0.652
             Prevalence : 0.348
         Detection Rate : 0.000
   Detection Prevalence : 0.000
      Balanced Accuracy : 0.500

       'Positive' Class : REAL
```

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.

6. 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.

7. 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

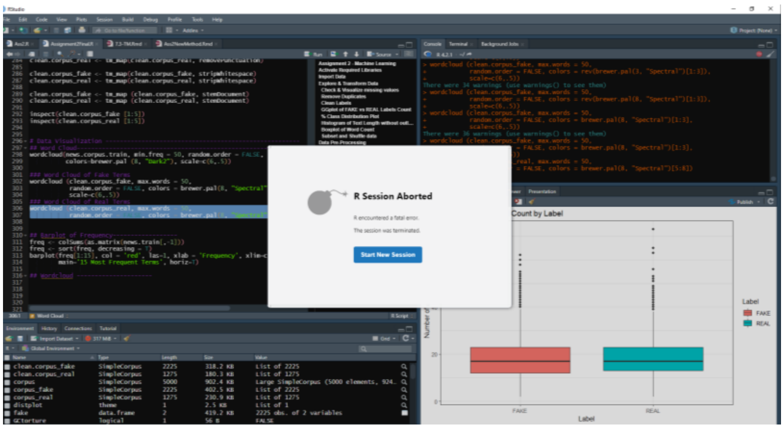
```
# Activate Required Libraries-------------------------------------------

library ('readr')   # to read the data
library ('tidyverse') # data import + tidying
library('tidytext') #for NLP
library ('party') # recursive partitioning
library ('dplyr') #for data manipulation
library ('stringr') # for data manipulation
library ('Hmisc') # for data analysis
library ('devtools') # simplify development of r packages
library('Amelia') #visualize missing values
library('textdata') #for sentiment analysis
library('ISLR') #for data analysis and manipulation
library('ggplot2') #for data visualization
library('GGally') #data visualization
library('tm') # for text mining
library('SnowballC') # for stemming words
library('corpus') #for text analysis
library('caret') #for classification and regression training
library('rpart') #models
library('rpart.plot') #visualize rpart
library('e1071') #SVM library
library('quanteda') #text analysis
library('irlba') #feature extraction
library('randomForest') # modelling
library('doSNOW') #for clusters
library('ROSE') #for random over sampling
library('caTools') #for train/test split
library('lsa') #latent semantic analysis
library('h2o') #machine learning
library('wordcloud') # word-cloud generator
library('RColorBrewer') #colour palette
library('corrgram') #correlation
library('corrplot') #correlation plot
library('car') #companion to applied regression
library('moments') #skewness, kurtosis tests
library('gridExtra') #extensions to grid system
library ('MLmetrics') #evaluation
library('class') #classification
library ('testthat') #check correct behaviour of code
library ('blorr') #logistic regression
library ('magrittr') #forward pipe operator %>%
library ('ROCR') #visualize performance of scoring classifiers
library ('pROC') #display and analyze ROC curves
```

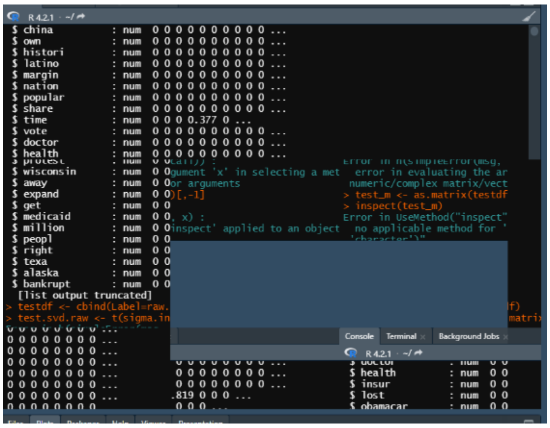
Appendix 3 – Fake and Real Document Term Matrices

```
> dtm.real <- DocumentTermMatrix(clean.corpus_real)
> dtm.real <- removeSparseTerms(dtm.real, 0.999)
> inspect(dtm.real)
<<DocumentTermMatrix (documents: 1275, terms: 1410)>>
Non-/sparse entries: 10840/1786910
Sparsity           : 99%
Maximal term length: 15
Weighting          : term frequency (tf)
Sample             :
     Terms
Docs  countri job million obama percent say state tax vote year
  210       0   0       0     0       1   0     1   0    0    0
  369       0   0       0     0       0   0     0   0    0    0
  383       0   3       0     0       1   0     0   0    0    0
  507       0   0       0     0       2   0     1   0    0    1
  519       0   0       0     0       0   0     0   0    0    2
  530       1   1       4     0       0   1     0   0    0    0
  605       0   0       0     0       0   0     2   2    0    0
  719       0   2       0     0       0   0     0   1    0    1
  815       0   0       0     0       0   0     0   0    0    0
  871       0   0       0     0       0   3     0   0    0    0
> dtm.fake <- DocumentTermMatrix(clean.corpus_fake)
> dtm.fake <- removeSparseTerms(dtm.fake, 0.999)
> inspect(dtm.fake)
<<DocumentTermMatrix (documents: 2225, terms: 1490)>>
Non-/sparse entries: 19053/3296197
Sparsity           : 99%
Maximal term length: 15
Weighting          : term frequency (tf)
Sample             :
      Terms
Docs   health job obama percent presid say state tax vote year
  111       2   0     0       0      0   0     0   0    0    1
  1129      1   0     0       1      0   1     2   1    1    0
  1167      0   0     0       0      0   1     0   2    0    0
  1191      0   0     0       0      0   0     0   0    0    0
  1245      0   0     0       0      0   1     0   0    0    1
  1329      0   0     0       0      0   0     0   2    0    0
  1886      0   0     0       0      1   0     0   0    0    0
  2052      0   0     0       0      0   0     1   0    1    0
  52        0   0     0       0      0   0     0   0    0    0
  866       0   2     0       0      0   0     0   1    0    1
> 
```

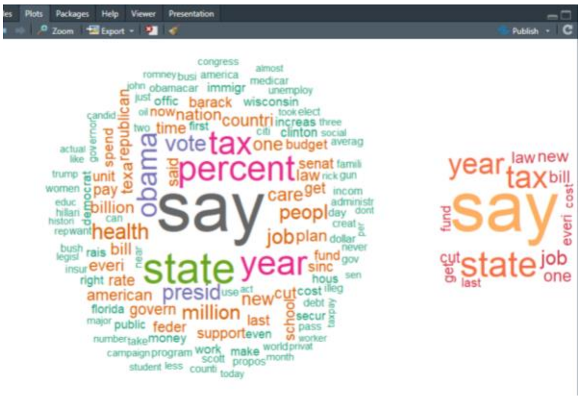
Appendix 4 – Technical Issues with R Studio

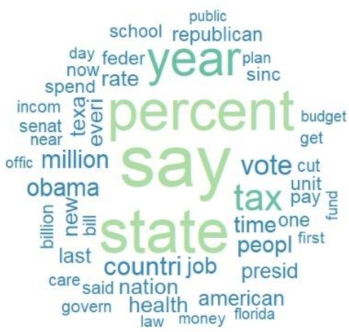
The RStudio Session would terminate and encounter a fatal error frequently, especially when creating plots, and then the entire workspace must be reloaded again. Another problem included the glitching and freezing, which may be due to the computer that was used for the machine learning.

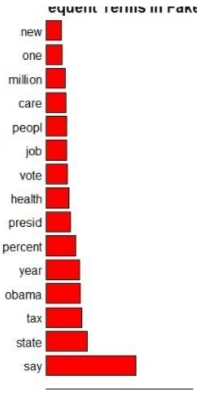


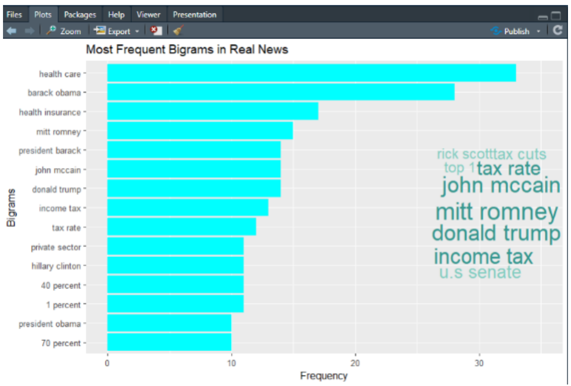


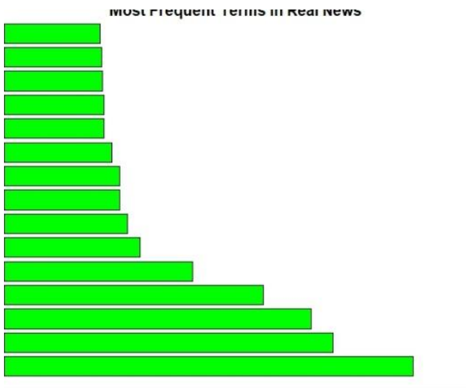
Creating a Word Cloud was the most recurring problem, as it would shut down RStudio, and a word cloud or any other type of plot cannot be formed straight after another word cloud without overlaying on the previous word cloud. Another problem was the printing of certain plots cropping out half the graph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











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:

```
> findAssocs (dtm.fake, c("say","state", 'obama', 'health', 'percent', 'vote', 'tax', 'one'),
+            corlimit=0.1)
$say
 hillari flipflop  clinton
    0.13     0.11     0.10

$state
    unit    resid    texan     texa  employe    everi approxim
    0.41     0.13     0.13     0.11     0.11     0.10     0.10

$obama
    barack      presid  administr      produc  pakistan sayspresid       sign
      0.59        0.39       0.23        0.11      0.11       0.11       0.10

$health
         care          insur         reform         coverag           plan         takeov
         0.76           0.28           0.21            0.19           0.17           0.17
         bill            law         access        medicaid governmentrun          stood
         0.16           0.16           0.14            0.13           0.12           0.12
         forc            buy           risk          govern          women
         0.11           0.11           0.11            0.10           0.10

$percent
unemploy     rate  graduat   infant  increas     corn    price     rise lowincom      gdp
    0.20     0.14     0.14     0.13     0.12     0.12     0.11     0.11     0.11     0.11
   popul
    0.10

$vote
   earli  patrick    measur      kay santorum    marco    rubio   repeat   imposs   murphi
    0.16     0.15      0.15     0.13     0.13     0.13     0.13     0.12     0.11     0.11
   senat democrat    jeann  shaheen   restor lowincom      sen
    0.11     0.11     0.11     0.11     0.11     0.11     0.10

$tax
    tobacco          rais       incom     properti         mike        sale          pay
       0.33          0.28        0.28         0.25         0.23        0.21         0.20
 millionair      internet       break        middl        class middleclass      famili
       0.20          0.20        0.19         0.19         0.17        0.16         0.15
       hike      wealthi          cut         nurs      increas        huge     governor
       0.15          0.15        0.13         0.13         0.12        0.12         0.11
     higher          code          gas         fail
       0.11          0.11        0.10         0.10

$one
   there forward    everi   attend   except   affect
    0.16    0.12     0.11     0.11     0.11     0.10
```

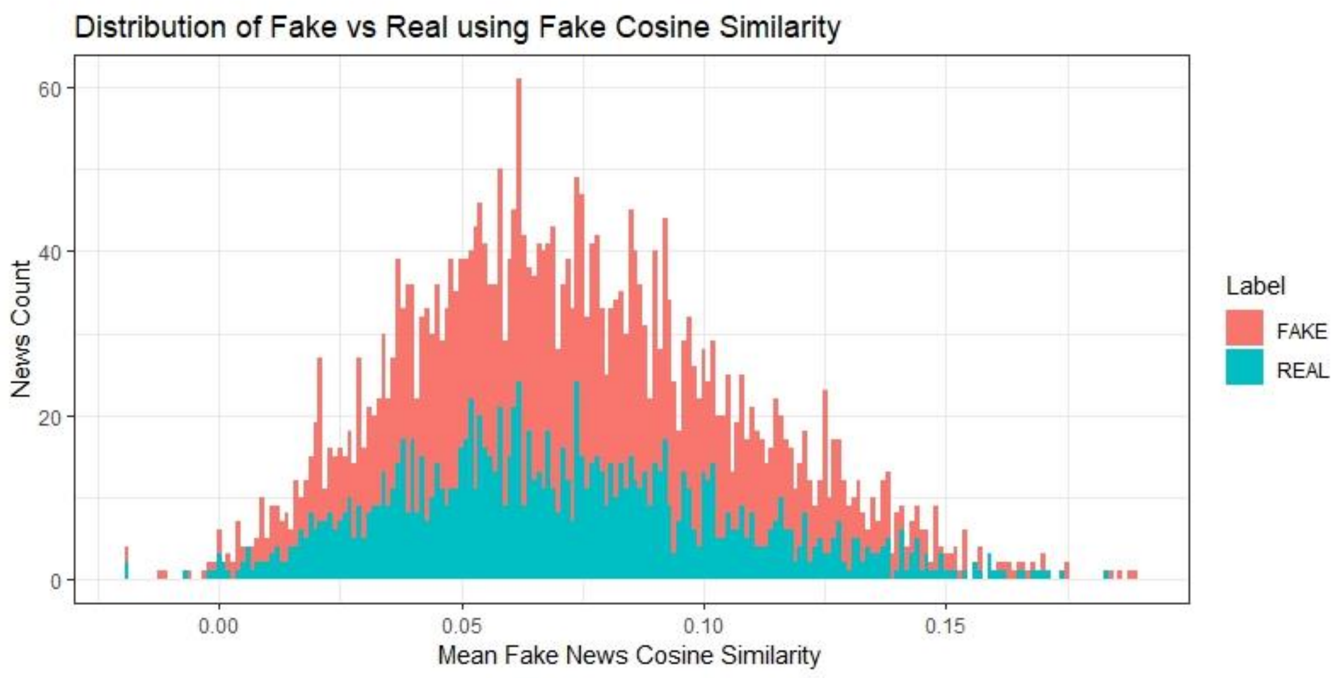
Word Associations between Real Terms

```
> findAssocs (dtm.real, c("say","state", 'obama', 'health', 'percent', 'vote', 'time', 'tax'),
+             corlimit=0.1)
$say
  financ campaign      rep     rule     fair     mitt    refus    scott     mean
    0.17     0.15     0.13     0.13     0.12     0.11     0.11     0.10     0.10

$state
     unit       blue overwhelm    popular wisconsin        red       larg     burden    treasur
     0.41       0.17      0.17       0.16      0.15       0.13       0.13       0.13       0.13
 virginia      union    budget       best      york        per       hour      child
     0.12       0.11      0.11       0.11      0.10       0.10       0.10       0.10

$obama
   barack     presid    illinoi   unilater       leak administr       iowa     mandat     promis
     0.62       0.44       0.19       0.16       0.16       0.15       0.13       0.13       0.13
      arm healthcar        pac     church      seven      polit       said     associ        hes
     0.13       0.13       0.13       0.13       0.12       0.12       0.11       0.11       0.11
     flag   campaign
     0.11       0.10

$health
    care     insur  aspirin    provis       bar    hispan   account    matern    either       law
    0.68      0.45     0.24      0.24      0.19      0.19      0.16      0.16      0.16      0.15
  mandat    expand     cost   coverag  hampshir      away  individu      save    expens       can
    0.13      0.12     0.12      0.12      0.12      0.11      0.11      0.11      0.11      0.10
   spend   subsidi
    0.10      0.10

$percent
     incom      earner          pay wealthiest         top      total       miss      wealth
      0.24        0.23         0.21       0.21        0.20       0.17       0.16        0.15
    declin      consum          gdp     bottom       gross      rough      kerri      histor
      0.14        0.14         0.13       0.12        0.12       0.12       0.12        0.12
     found     decreas
      0.12        0.12

$vote
    regist       share      margin        miss    congress      record         won      nelson
      0.22        0.21        0.20        0.19        0.18        0.18        0.17        0.16
      cast  republican    statewid     smaller       elect         sen      resign        kept
      0.16        0.15        0.14        0.13        0.13        0.13        0.13        0.13
     kerri     richard      within        mark      latino     popular       onlin        john
      0.13        0.13        0.12        0.12        0.11        0.11        0.11        0.11
    mccain        came     johnson      career      engag    straight    opportun   provision
      0.11        0.11        0.11        0.11        0.11        0.11        0.11        0.11
     brave       insid      ballot
      0.11        0.11        0.10
```

Appendix 6 – Cosine Similarity

Code snippet sourced from DataScienceDojo:

```
#create matrix to compare similarities
cos.news <- cosine(t(as.matrix(news.svd[,-1])))
dim(cos.news)

#get indexes from training set of fake news
fake.indexes <- which(raw.news.train$Label == 'FAKE')
#create similarity feature
news.svd$FakeSim <- rep(0.0, nrow(raw.news.train))
for (i in 1:nrow(news.svd)) {
  news.svd$FakeSim[i] <- mean(cos.news[i, fake.indexes])
}

#visualize feature with histogram
ggplot(news.svd, aes(x=FakeSim, fill = Label)) +
  theme_bw() +
  geom_histogram(binwidth = 0.001) +
  labs(y='News Count',
       x='Mean Fake News Cosine Similarity',
       title = 'Distribution of Fake vs Real using Fake Cosine Similarity')
```




```
> summary(news.svd$FakeSim[news.svd$Label == 'FAKE'])
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.01857  0.04883  0.07080  0.07344  0.09571  0.18926
> summary(news.svd$FakeSim[news.svd$Label == 'REAL'])
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.01857  0.04630  0.06764  0.07093  0.09226  0.18332
```

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

```
# Cross Validation 1 Decision Tree-----------------------------
#create stratified folds
set.seed(100)
cv.folds <- createMultiFolds(newsdf$Label, k = 5, times=3)

#set value of K parameter
cv.cntrl <- trainControl(method = "repeatedcv", number = 5,
                         repeats = 3, index = cv.folds)

#create cluster using doSNOW to work on 2 logical cores
cl <- makeCluster(2, type = 'SOCK')
registerDoSNOW(cl)

#create a single decision tree
rpart.cv.1 <- train(Label~., data = newsdf, method = 'rpart',
                    trControl = cv.cntrl, tuneLength = 5)
# when pre-processing is done, stop cluster
stopCluster(cl)

#results
rpart.cv.1
```

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).

```
> rpart.cv.1
CART

3500 samples
1483 predictors
   2 classes: 'FAKE', 'REAL'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 2800, 2800, 2800, 2800, 2800, 2800, ...
Resampling results across tuning parameters:

  cp           Accuracy   Kappa
  0.001568627  0.6189524  0.07447961
  0.001830065  0.6190476  0.06747664
  0.002352941  0.6214286  0.06325412
  0.003398693  0.6243810  0.03705441
  0.003921569  0.6265714  0.02558987

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.003921569.
```

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

```
> rpart.cv.2
CART

3500 samples
1483 predictors
   2 classes: 'FAKE', 'REAL'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 2800, 2800, 2800, 2800, 2800, 2800, ...
Resampling results across tuning parameters:

  cp           Accuracy   Kappa
  0.003137255  0.6241905  0.08394301
  0.003921569  0.6276190  0.05862579
  0.004313725  0.6276190  0.05862579
  0.004444444  0.6276190  0.05255269
  0.009411765  0.6339048  0.02739286

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.009411765.
```

Cross Validation 3: after adding the Singular Value Decomposition

```
> rpart.cv.3
CART

3500 samples
  30 predictor
   2 classes: 'FAKE', 'REAL'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 2800, 2800, 2800, 2800, 2800, 2800, ...
Resampling results across tuning parameters:

  cp           Accuracy   Kappa
  0.005490196  0.6291429  0.07348164
  0.005751634  0.6291429  0.07348164
  0.008627451  0.6349524  0.08134381
  0.009019608  0.6349524  0.07511671
  0.010980392  0.6340000  0.06091025

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.009019608.
```

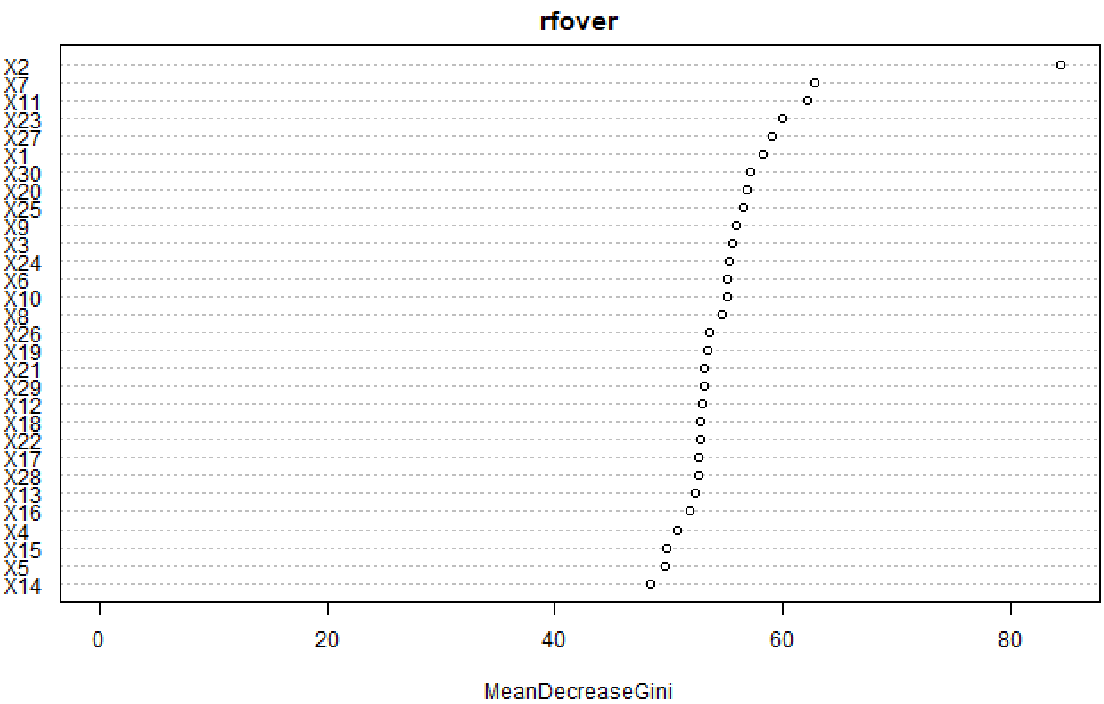
Appendix 8 – Class Imbalance Results

Over-sampling

- **Confusion Matrix:**

```
> rfover <- randomForest(Label~., data = over)
> rfoverpred <- predict(rfover, newstest)
> confusionMatrix(rfoverpred, newstest$Label, positive = 'REAL')
Confusion Matrix and Statistics

          Reference
Prediction FAKE REAL
      FAKE  504  252
      REAL   52   67

               Accuracy : 0.6526
                 95% CI : (0.62, 0.6841)
    No Information Rate : 0.6354
    P-Value [Acc > NIR] : 0.1542

                  Kappa : 0.1345

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.21003
            Specificity : 0.90647
         Pos Pred Value : 0.56303
         Neg Pred Value : 0.66667
             Prevalence : 0.36457
         Detection Rate : 0.07657
   Detection Prevalence : 0.13600
      Balanced Accuracy : 0.55825

       'Positive' Class : REAL
```

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

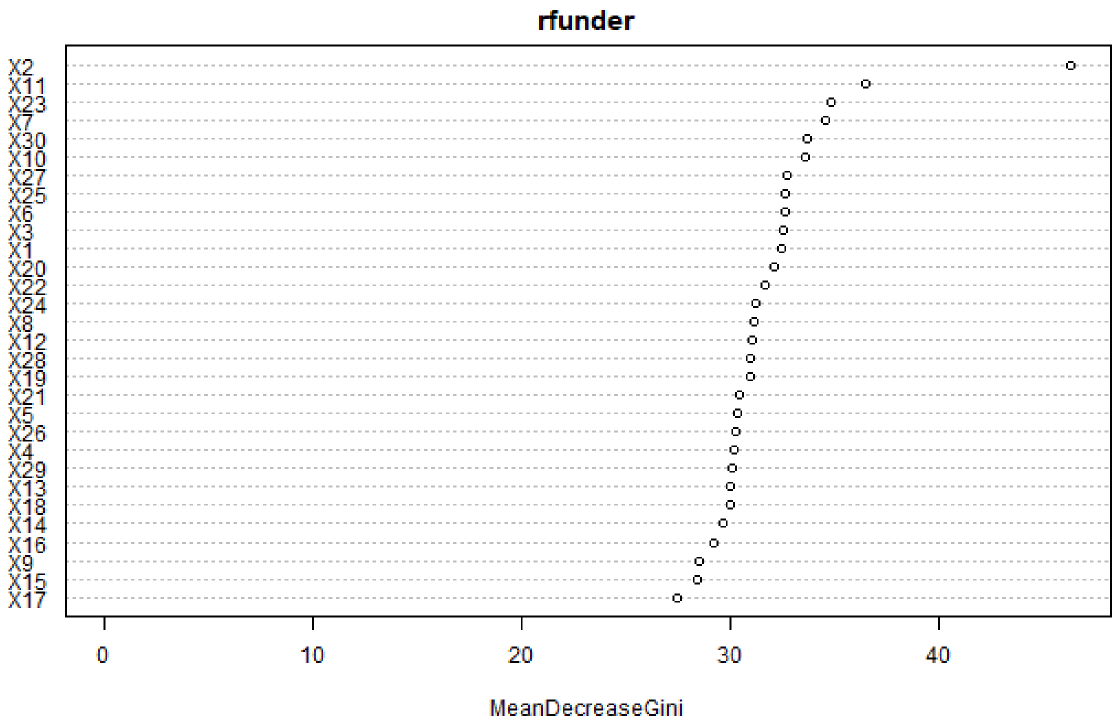


Under-sampling

- **Confusion Matrix**

```
> rfunderpred <- predict(rfunder, newstest)
> confusionMatrix(rfunderpred, newstest$Label, positive = 'REAL')
Confusion Matrix and Statistics

          Reference
Prediction FAKE REAL
      FAKE  322  146
      REAL  234  173

               Accuracy : 0.5657
                 95% CI : (0.5321, 0.5989)
    No Information Rate : 0.6354
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1147

 Mcnemar's Test P-Value : 8.082e-06

            Sensitivity : 0.5423
            Specificity : 0.5791
         Pos Pred Value : 0.4251
         Neg Pred Value : 0.6880
             Prevalence : 0.3646
         Detection Rate : 0.1977
   Detection Prevalence : 0.4651
      Balanced Accuracy : 0.5607

       'Positive' Class : REAL
```

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

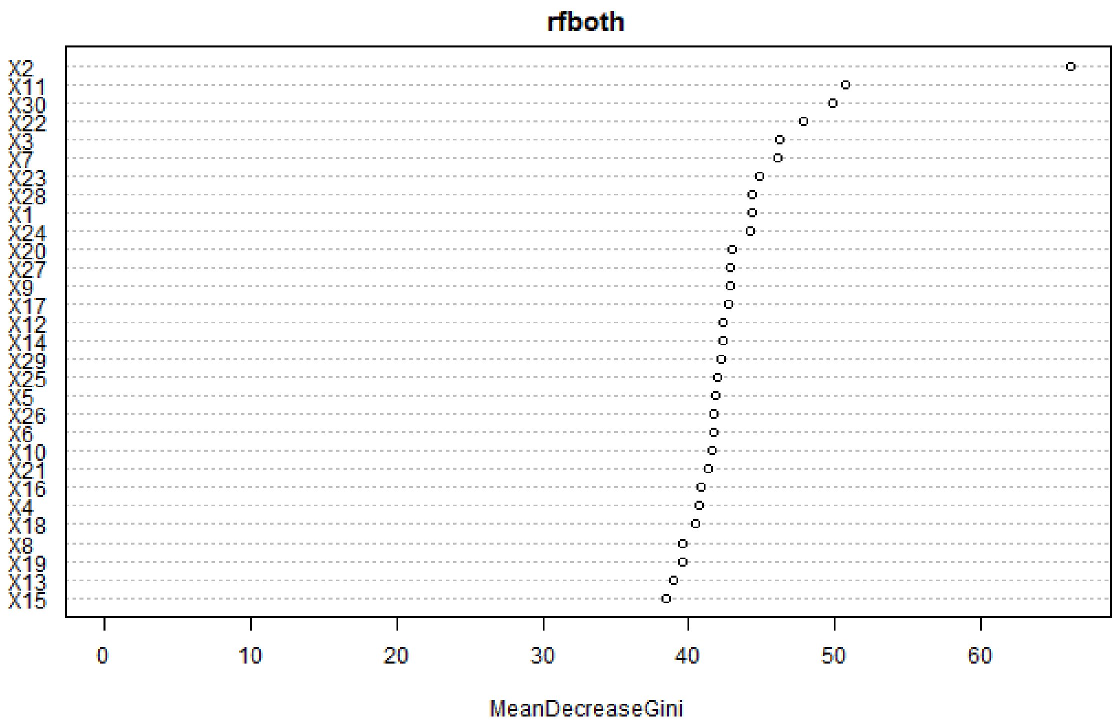


Both Sampling Methods

- **Confusion Matrix**

```
> rfboth <-randomForest(Label~., data=both)
> set.seed(100)
> rfbothpred <- predict(rfboth, newstest)
> confusionMatrix(rfbothpred, newstest$Label, positive = 'REAL')
Confusion Matrix and Statistics

          Reference
Prediction FAKE REAL
      FAKE  363  178
      REAL  193  141

               Accuracy : 0.576
                 95% CI : (0.5425, 0.609)
    No Information Rate : 0.6354
    P-Value [Acc > NIR] : 0.9999

                  Kappa : 0.0939

 Mcnemar's Test P-Value : 0.4673

            Sensitivity : 0.4420
            Specificity : 0.6529
         Pos Pred Value : 0.4222
         Neg Pred Value : 0.6710
             Prevalence : 0.3646
         Detection Rate : 0.1611
   Detection Prevalence : 0.3817
      Balanced Accuracy : 0.5474

       'Positive' Class : REAL
```

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



8. 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>.