Data Science: Capstone - Fake New Detection Project

A lie gets halfway around the world before the truth has a chance to get its pants on - Winston Churchill

Shanna Jardim

01/04/2021

INTRODUCTION

Misleading content such as fake news, fake reviews, conspiracy theories and over exaggeration of the truth is nothing new. However it has increasingly become dangerous for online users in the last decade. What makes the present-day fake news most alarming is the speed with which it spreads. According to a new survey by Pew Research Center, conducted in association with the John S. and James L. Knight Foundation - 62% of U.S. adults access news from social media.

Social Media is one of the biggest sources of spreading false news and poorly written news articles, which may have a certain degree of truth but are not entirely accurate or are completely made up. Usually, this kind of news is designed to promote or defend a certain agenda or biased opinion. One reason for the increase in misleading information is how easy it is for anyone to write fake reviews or articles on the web and there is no pre-approval process of the writings and spreading false information or beliefs across all social media. This poses a challenge to many readers to decide if the content is fake or true. This has given way to the development of Fact checking websites that use machine learning with natural language processing and research to detect fake news.

GOAL OF PROJECT

The main challenge in this line of research is collecting quality data, i.e., instances of fake and real news articles on a balanced distribution of topics. In this assignment, I will use a dataset from Kaggle that has already separated a collection of articles into Fake or True. I will train an algorithm to detect if a new article is false or true by simply using metrics gathered from the text of the articles.

Natural Language Processing The news article data and headlines are in text format. Building automated machine learning models on text data involves cleaning and converting textual data to machine readable format. Natural-language processing (NLP) is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages, and how to program computers to fruitfully process large amounts of natural language data.

DATA SET

I am using the following datasets:

Source based Fake News Classification Fake and real news dataset

EXPLORING THE DATA

Data Set 1

```
as_tibble(head(FakeNews))
```

```
## # A tibble: 6 x 4
##   title                          text                          subject date
##   <chr>                          <chr>                         <chr>   <chr>
## 1 " Donald Trump Sends Out Emba~ Donald Trump just couldn t w~ News    December~
## 2 " Drunk Bragging Trump Staffe~ House Intelligence Committee~ News    December~
## 3 " Sheriff David Clarke Become~ On Friday, it was revealed t~ News    December~
## 4 " Trump Is So Obsessed He Eve~ On Christmas day, Donald Tru~ News    December~
## 5 " Pope Francis Just Called Ou~ Pope Francis used his annual~ News    December~
## 6 " Racist Alabama Cops Brutali~ The number of cases of cops ~ News    December~
```

```
as_tibble(head(TrueNews))
```

```
## # A tibble: 6 x 4
##   title                       text                           subject  date
##   <chr>                       <chr>                          <chr>    <chr>
## 1 As U.S. budget fight looms~ "WASHINGTON (Reuters) - The h~ politic~ "December~
## 2 U.S. military to accept tr~ "WASHINGTON (Reuters) - Trans~ politic~ "December~
## 3 Senior U.S. Republican sen~ "WASHINGTON (Reuters) - The s~ politic~ "December~
## 4 FBI Russia probe helped by~ "WASHINGTON (Reuters) - Trump~ politic~ "December~
## 5 Trump wants Postal Service~ "SEATTLE/WASHINGTON (Reuters)~ politic~ "December~
## 6 White House, Congress prep~ "WEST PALM BEACH, Fla./WASHIN~ politic~ "December~
```

```
nrow(FakeNews)
```

```
## [1] 23481
```

```
nrow(TrueNews)
```

```
## [1] 21417
```

For the purposes of this project I will only use the title and article text to identify whether an article is fake or true adding factors discussed in this document.

- **Observations**
- there are more Fake news articles than true news
- title and text can be joined into one Text field
- The author is unknown
- The source of articles is unknown.
- subject and date is not needed for this analysis so I will remove it.
- Fake or True flag must be added when combining data

Data Set 2

```
as_tibble(head(News_Articles))
```

```
## # A tibble: 6 x 12
##   author published title text  language site_url main_img_url type  label
##   <fct>  <chr>     <chr> <chr> <fct>    <fct>    <fct>        <fct> <chr>
## 1 Barra~ 2016-10-~ musl~ prin~ english  100perc~ http://bb4s~ bias  Real
## 2 reaso~ 2016-10-~ atto~ atto~ english  100perc~ http://bb4s~ bias  Real
## 3 Barra~ 2016-10-~ brea~ red ~ english  100perc~ http://bb4s~ bias  Real
## 4 Fed Up 2016-11-~ pin ~ emai~ english  100perc~ http://100p~ bias  Real
## 5 Fed Up 2016-11-~ fant~ emai~ english  100perc~ http://100p~ bias  Real
## 6 Barra~ 2016-11-~ hill~ prin~ english  100perc~ http://bb4s~ bias  Real
## # ... with 3 more variables: title_without_stopwords <fct>,
## #   text_without_stopwords <fct>, hasImage <int>
```

```
News_Articles %>% group_by(type) %>% summarize(NoArticles = n())
```

```
## # A tibble: 9 x 2
##   type         NoArticles
##   <fct>             <int>
## 1 ""                    1
## 2 "bias"              436
## 3 "bs"                601
## 4 "conspiracy"        430
## 5 "fake"               15
## 6 "hate"              244
## 7 "junksci"           102
## 8 "satire"            146
## 9 "state"             121
```

- **Observations**
- Although "type" would be a great predicting factor, because Dataset 1 does not contain this information and for the purpose of removing any bias I will remove all other columns except title & text
- Need to Concatenate the datasets
- (Will leave False & True Dataset separate for now)
- Since we have more fake articles from the Dataset 1, I will only add the true articles from Dataset 2 and ensure there is an equal amount of Fake and True articles. This is to ensure there is no bias when predicting.

```
News_Articles <- subset(News_Articles, select = c(title_without_stopwords,text_without_stopwords,label)
News_Articles <- News_Articles %>%
  rename(title = title_without_stopwords,
         text = text_without_stopwords,
         type = label)
```

```
nrow(FakeNews)
```

```
## [1] 22218
```

```
nrow(TrueNews)
```

```
## [1] 22218
```

```
# Removed unused dataframes
rm(TrueNews2, News_Articles)
```

Word Analysis

1. **The following Text Preprocessing logic is used:**

- Add title & text to same field
- Load the text as a corpus (A text corpus is a language resource consisting of a large and structured set of texts)
- Convert the text to lower case
- Remove numbers
- Remove punctuation
- Remove english common stopwords (Discussed later in this project)
- Eliminate extra white spaces

```
## Loading required package: RColorBrewer
```

2. Text stemming - which reduces words to their root form. Stemming is changing the words into their original form and decreasing the number of word types or classes in the data. For example, the words "Running," "Ran," and "Runner" will be reduced to the word "run. Observation: In my research I have noticed that this is not always the most accurate process as the stemming process sometime cuts off a full word.

Once the text preprocessig is not we can product a Word Cloud and table with the results

Fake News Word Cloud

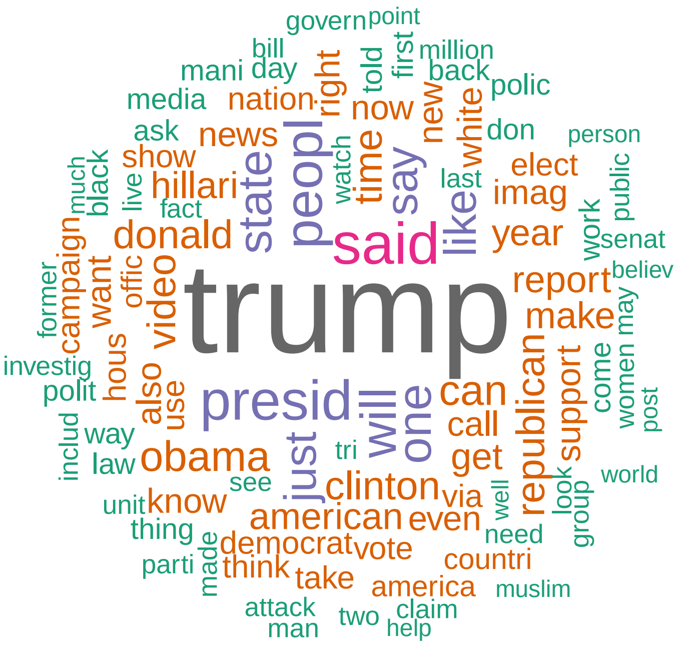


Table 1: Fake News: Top5 most Frequent words

	word	freq	type
trump	trump	77693	fake
said	said	29566	fake
presid	presid	27345	fake
peopl	peopl	25112	fake
will	will	24148	fake

True News Word Cloud

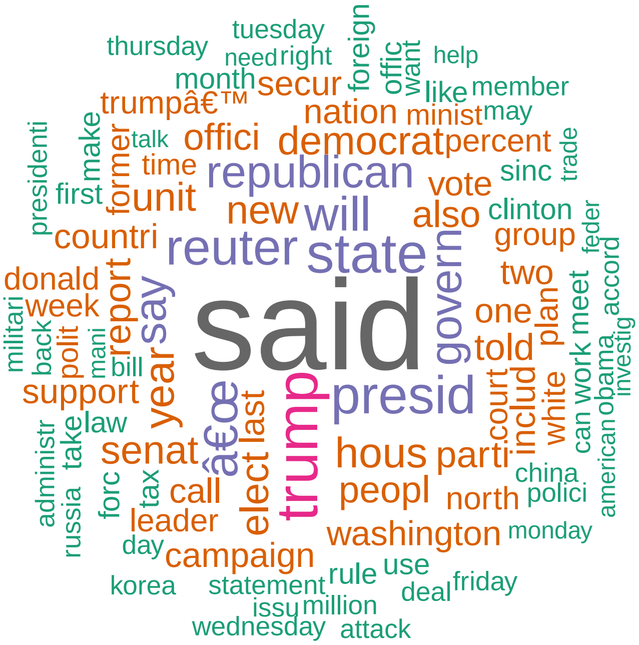


Table 2: True News: Top5 most Frequent words

	word	freq	type
said	said	99933	true
trump	trump	49438	true
state	state	36652	true
presid	presid	28409	true
reuter	reuter	28402	true

TRANSFORMATION OF THE DATA

1. Check for any Null Values
2. Add an ID to the dataset

```
NewsData$ID <-seq_len(nrow(NewsData))
NewsData <- select(NewsData, ID, everything())
```

3. Remove Title as it was added to the text previously

```
NewsData <- subset(NewsData, select = -c(title) )
```

```
library(quanteda)
```

4. Add number of sentences per article

```
NewsData$No_of_sentences <- nsentence(NewsData$text)
```

5. Add Number of characters per article

```
NewsData$TextLength <- nchar(NewsData$text)
summary(NewsData$TextLength)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##      15    1304    2265    2487    3173   49766
```

6. **Punctuation** Punctuation in natural language provides the grammatical context to the sentence. Punctuations such as a commas, exclmation marks, full stop etc. wont add much value in understanding the meaning of the sentence, however before removing all punctuation, questions and statements can be analysed.

Calculating the number of Exclamation marks and question marks may add some interesting insight

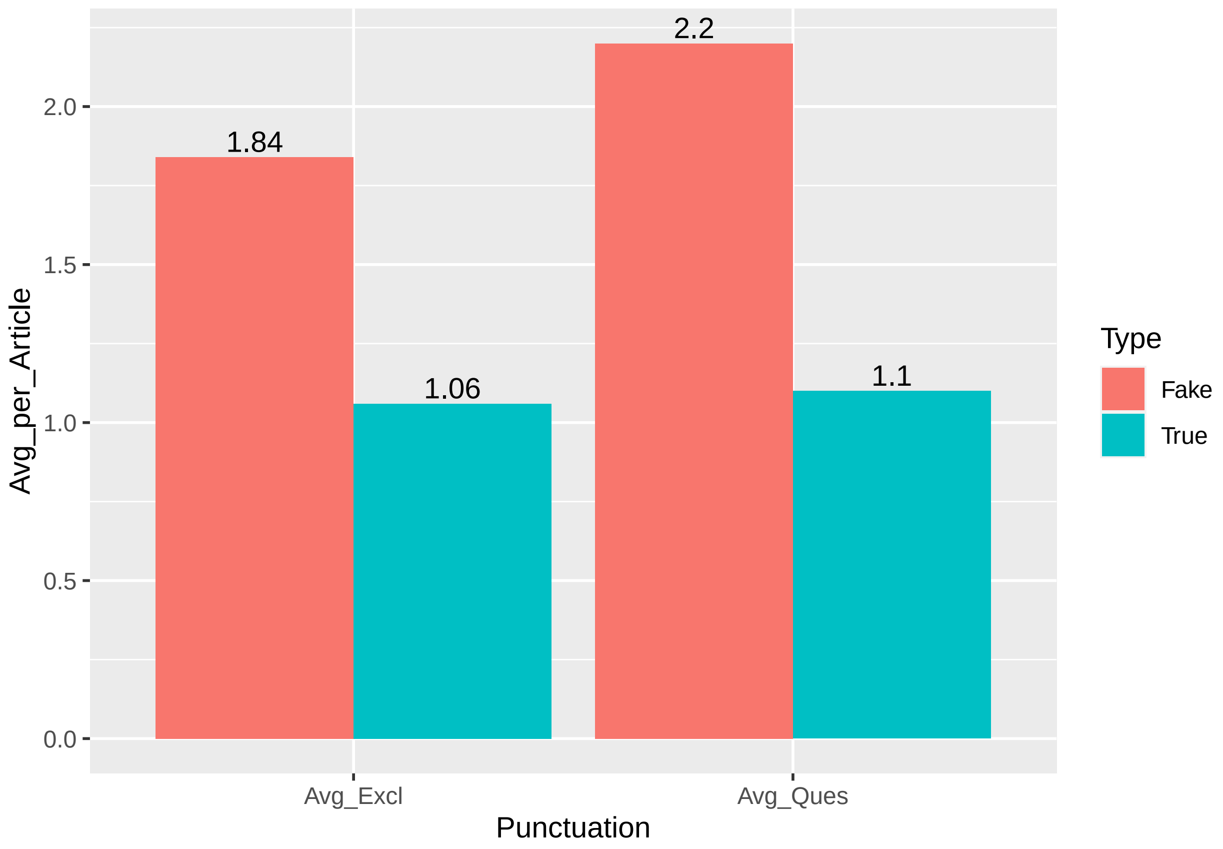
Used the Sapply function to calculate number of exclamation marks

```
NewsData$No_of_excl <- sapply(NewsData$text, 
                 function(x) length(unlist(strsplit(as.character(x), "\\!+"))))
```

Used the Sapply function to calculate number of question marks

```
NewsData$No_of_question <- sapply(NewsData$text, 
                    function(x) length(unlist(strsplit(as.character(x), "\\?+"))))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



There is a lot more punctuation used in false new than true, this may be directly linked to the amount of words, sentences or the length of the article. I will explore the correlation between these measures further down.

7. Make all text lower case

```
NewsData$text <- tolower(NewsData$text) #make it lower case
```

8. Add word counts of top word found in Fake News

```
NewsData$No_of_Wordtrump <- str_count(NewsData$text, "trump")
```

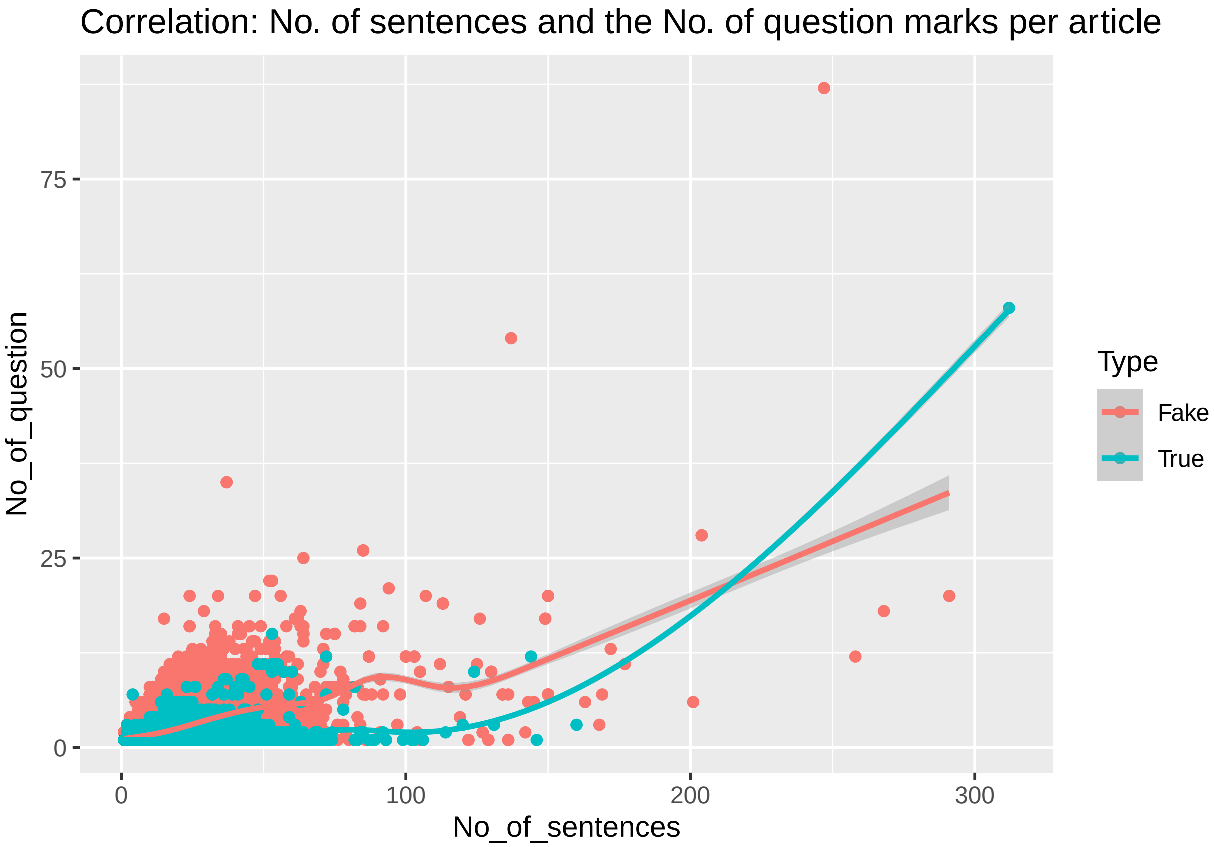
9. Add word counts of top word found in True News

```
NewsData$No_of_Wordsaid <- str_count(NewsData$text, "said")
```

View the all measures by type

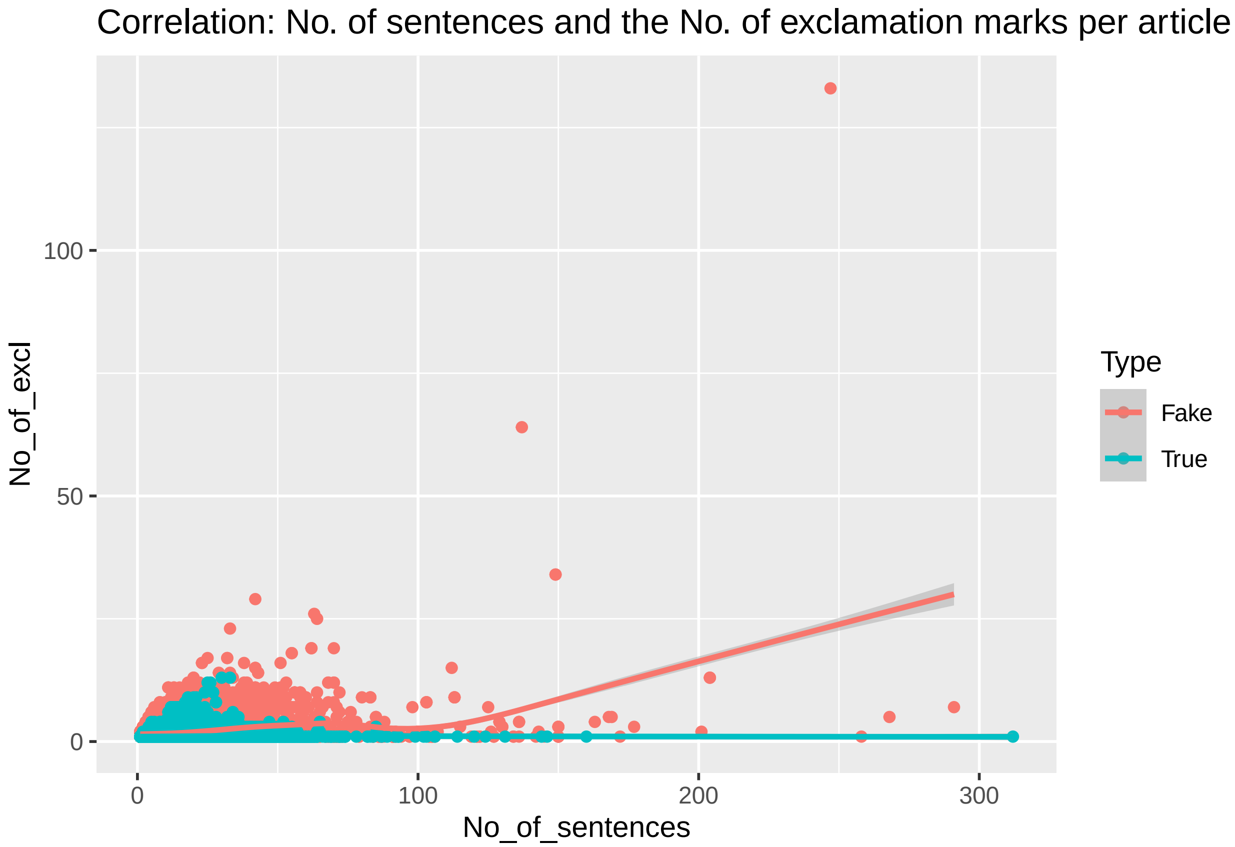
Table 3: Data Measures by article type

Type	Avg_no_Sentences	Avg_TextLength	Max_TextLength	Avg_no_excl	Avg_no_question	Avg_no_trump	Avg_no_said
Fake	15.57296	2519.192	49766	1.839544	2.203844	4.120623	1.451346
True	13.90904	2454.817	30218	1.059636	1.103205	2.819651	4.505671



Correlations

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



The is an evident positive correlation between the number of sentences and the number of question marks however this is not so evident with the number of exclamations.

10. **Remove Stop words** Stops Words (most common words in a language which do not provide much context) can be processed and filtered from the text as they are more common and hold less useful information They should be removed before performing further analysis. Some Stop words act like a connecting part of a sentences, for example, conjunctions "and", "or" and "but", prepositions like "of", "in", "from", "to", and the articles "a", "an", and "the". Such stop words may take up valuable processing time, and hence removing stop words as a part of data preprocessing is a key first step in natural language processing.

```
StopWords <- removeWords(NewsData$text, stopwords("en"))
StopWords <- data.frame(StopWords)
StopWords$ID <-seq_len(nrow(StopWords))
StopWords <- select(StopWords, ID, everything())
NewsData <- left_join(NewsData,StopWords,by="ID")
NewsData <- NewsData %>% rename(NoStop_text = StopWords)
```

After removing the stop words there are many white spaces left between words so I will remove the duplicate white spaces between the words.

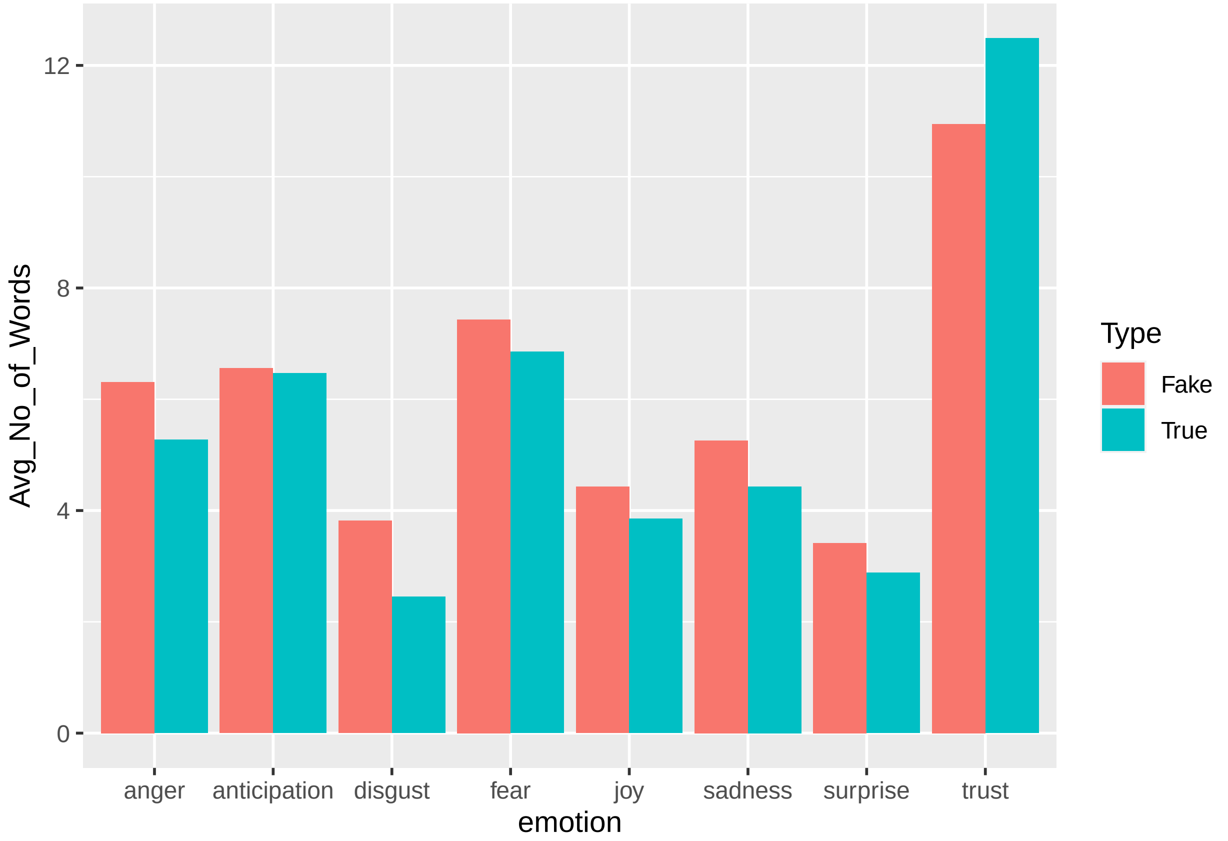
```
NewsData$NoStop_text <- str_replace_all(NewsData$NoStop_text, fixed("  "), " ")
```

11. Add number of words per article after removing Stop words

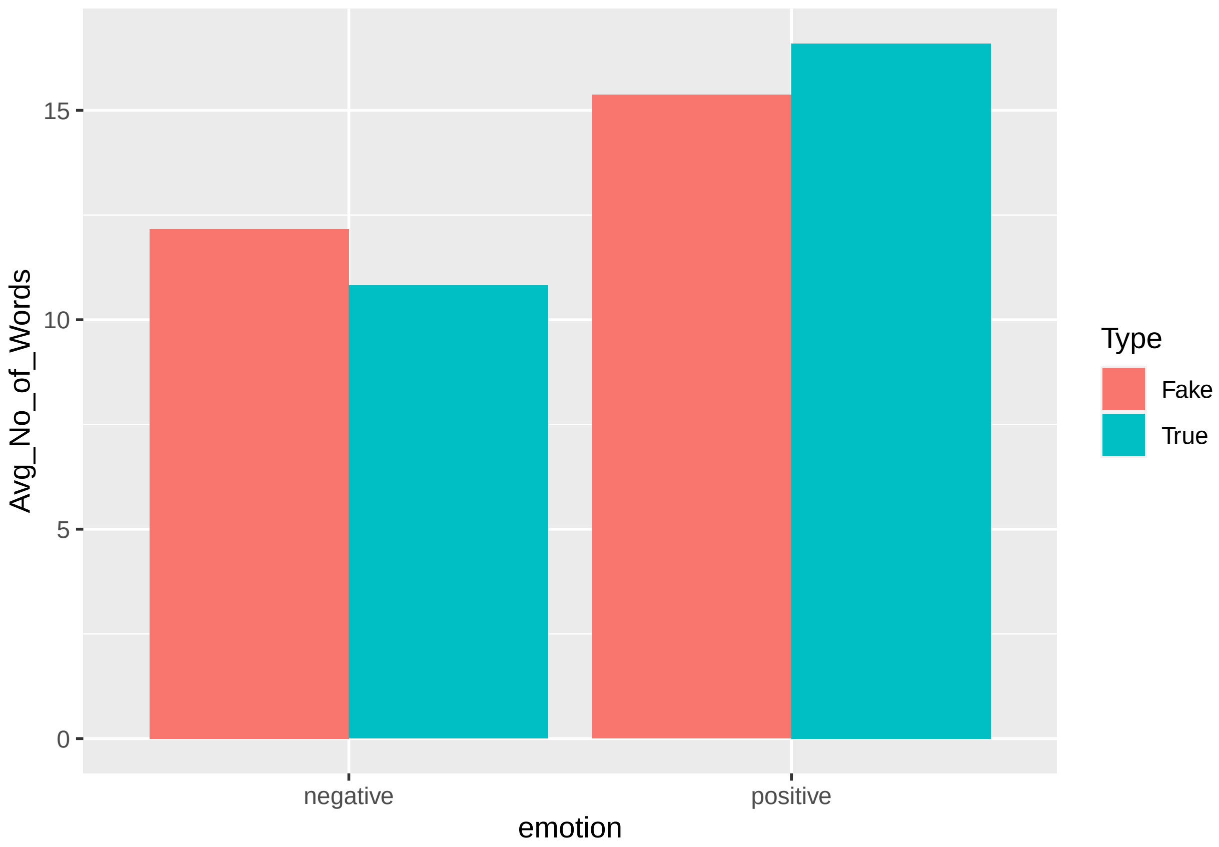
```
NewsData$No_of_words <- sapply(strsplit(NewsData$NoStop_text, " "), length)
```

12. **Sentiment Analysis** Sentiment Analysis: The study of extracted information to identify reactions, attitudes, context and emotions. R offers the get_nrc_sentiment function via the Tidy or Syuzhet packages for analysis of emotion words expressed in text. Both packages implemented Saif Mohammad's NRC Emotion lexicon, comprised of several words for emotion expressions of anger, fear, anticipation, trust, surprise, sadness, joy, and disgust

```
emotion <-get_nrc_sentiment(as.character(NewsData$NoStop_text))
```



Observation
"Trust" is the only sentiment on average that is more in true news than in fake news. I will add this observation to the working dataset



I will use the negative, positive and trust as predicts for this analysis

View the all additional measures by type

Table 4: Data Measures by article type

Type	No_articles	Avg_no_Sentences	Avg_TextLength	Max_TextLength	Avg_no_excl	Avg_no_question
Fake	22218	15.57296	2519.192	49766	1.839544	2.203844
True	22218	13.90904	2454.817	30218	1.059636	1.103205

Table 5: Data Measures by article type

Type	No_articles	Avg_No_trump	Avg_No_said	Avg_No_trust	Avg_No_positive	Avg_No_negative
Fake	22218	4.120623	1.451346	10.94671	15.36970	12.16442
True	22218	2.819651	4.505671	12.48650	16.59911	10.82078

MODEL BUILDING & TRAINING

Now that we have prepared the data we can start building and training the data

Since most machine learning models only accept numerical variables, preprocessing the categorical variables becomes a necessary step. We need to convert these categorical variables to numbers such that the model is able to understand and extract valuable information. Here I will make Fake and True - 1 and 0 respectively

Encoding categorical data

```
NewsData$Type = factor(NewsData$Type,
                           levels= c('Fake','True'),
                           labels= c(1,0))
```

Splitting the dataset into the Training set and Test set

```
library(caTools)
library(caret)
set.seed(123)
```

Feature scaling is neccessary before building the models Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization.

MODEL 1 - Logistic Regression

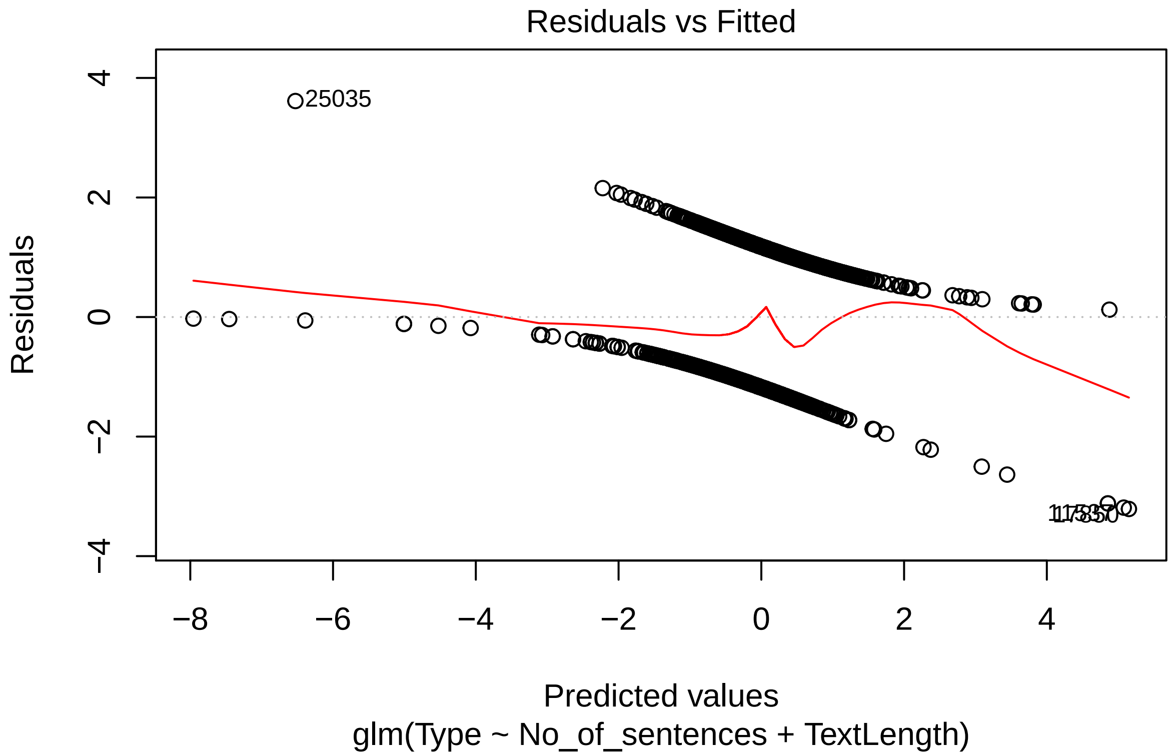
"Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression)" ~ WIKIPEDIA

For the first model, I am going to use the number of Sentences and Text Length only for prediction

```
# Fitting Logistic Regression to the Training set
classifier = glm(formula = Type ~ No_of_sentences + TextLength,
                 family = binomial,
                 data = train_set)
classifier
```

```
##
## Call:  glm(formula = Type ~ No_of_sentences + TextLength, family = binomial,
##     data = train_set)
##
## Coefficients:
##     (Intercept)  No_of_sentences       TextLength
##      -0.0008657       -0.5062275        0.4085171
##
## Degrees of Freedom: 35547 Total (i.e. Null);  35545 Residual
## Null Deviance:       49280
## Residual Deviance: 48810     AIC: 48820
```

```
plot(classifier, 1)
```



Predicting the Test set results

```
prob_pred = predict(classifier, type = 'response', newdata = test_set[2:3])
y_pred = ifelse(prob_pred > 0.5, 1, 0)
y_pred <-as.factor(y_pred)
```

A confusion matrix is a very useful tool for calibrating the output of a model and examining all possible outcomes of your predictions (true positive, true negative, false positive, false negative).

```
# Making the Confusion Matrix
require(caret)
cm<-confusionMatrix(data=y_pred,
                    reference=test_set$Type)
```

```
Accuracy<-cm$overall[1]
Accuracy
```

```
##  Accuracy
## 0.4456571
```

We can see from the graph that this is not a very accurate prediction with an outcome using only No. of Sentences and Text Length, lets see how this changes when adding more measures for prediction.

MODEL 2 - Logistic Regression 2 : No of Words & Sentiment

Using Logistic Regression again with the sentiment of the words.

```
# Fitting Logistic Regression to the Training set
classifier2 = glm(formula = Type ~ No_of_words + trust + negative + positive,
                 family = binomial,
                 data = train_set)
classifier2
```

```
##
## Call:  glm(formula = Type ~ No_of_words + trust + negative + positive,
##     family = binomial, data = train_set)
##
## Coefficients:
## (Intercept)  No_of_words        trust     negative     positive
##  -9.374e-05   -8.805e-01    9.996e-01   -3.854e-01    2.648e-01
##
## Degrees of Freedom: 35547 Total (i.e. Null);  35543 Residual
## Null Deviance:       49280
## Residual Deviance: 46590     AIC: 46600
```

```
# Predicting the Test set results
prob_pred2 = predict(classifier2, type = 'response', newdata = test_set[8:11])
y_pred2 = ifelse(prob_pred2 > 0.5, 1, 0)
y_pred2 <-as.factor(y_pred2)

# Making the Confusion Matrix
require(caret)
cm<-confusionMatrix(data=y_pred2,
                    reference=test_set$Type)
Accuracy<-cm$overall[1]
Accuracy
```

```
##  Accuracy
## 0.3755626
```

Result is less than previous

MODEL 3 - Logistic Regression 3

Using all measures

```
# Fitting Logistic Regression to the Training set
classifier3 = glm(formula = Type ~ .,
                  family = binomial,
                  data = train_set)
classifier3
```

```
##
## Call:  glm(formula = Type ~ ., family = binomial, data = train_set)
##
```

```
## Coefficients:
##     (Intercept)  No_of_sentences       TextLength       No_of_excl
##        -0.63711          0.03791          2.54922         -1.99427
##  No_of_question  No_of_Wordtrump   No_of_Wordsaid      No_of_words
##        -2.33541         -0.30466          1.99965         -4.25069
##           trust         negative         positive
##         0.66937         -0.29141          0.64353
##
## Degrees of Freedom: 35547 Total (i.e. Null);  35537 Residual
## Null Deviance:       49280
## Residual Deviance: 26490     AIC: 26520
```

```
# Predicting the Test set results
prob_pred3 = predict(classifier3, type = 'response', newdata = test_set[-1])

y_pred3 = ifelse(prob_pred3 > 0.5, 1, 0)
y_pred3 <-as.factor(y_pred3)

# Making the Confusion Matrix
require(caret)
cm<-confusionMatrix(data=y_pred3,
                    reference=test_set$Type)
Accuracy<-cm$overall[1]
Accuracy
```

```
##  Accuracy
## 0.1493024
```

Still not a very significant accuracy. Let's use a different method of classification prediction.

MODEL 4 - SUPPORT VECTOR MACHINE

"Support-vector machines are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis" ~ WIKIPEDIA

Using the Support Vector Machine starting with only No_of_sentences and Text Length

```
## Warning: package 'e1071' was built under R version 3.6.3
```

```
# Fitting SVM to the Training set and predicting the test set results
SVM_classifier = svm(formula = Type ~ No_of_sentences + TextLength,
                 data = train_set,
                 type = 'C-classification',
                 kernel = 'linear')

# Predicting the Test set results
y_pred = predict(SVM_classifier,  newdata = test_set[2:3])

# Making the Confusion Matrix
require(caret)
cm<-confusionMatrix(data=y_pred,
                    reference=test_set$Type)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    0
##          1 1234  759
##          0 3210 3685
##
##                Accuracy : 0.5534
##                  95% CI : (0.543, 0.5638)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1069
##
##  Mcnemar’s Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.2777
##             Specificity : 0.8292
##          Pos Pred Value : 0.6192
##          Neg Pred Value : 0.5344
##              Prevalence : 0.5000
##          Detection Rate : 0.1388
##    Detection Prevalence : 0.2242
##       Balanced Accuracy : 0.5534
##
##        ’Positive’ Class : 1
##
```

```
Accuracy <-round(cm$overall[1],2)
Accuracy
```

```
## Accuracy
##     0.55
```

An Accuracy for Support Vector Machine learning method is higher than all models of Logistic Regression

MODEL 5 - SUPPORT VECTOR MACHINE

In this Algorithm I am going to use all the fields to predict if an article is True or Fake

```
# Fitting SVM to the Training set and predicting the test set results
library(e1071)
SVM2_classifier = svm(formula = Type ~ .,
                     data = train_set,
                     type = 'C-classification',
                     kernel = 'linear')

# Predicting the Test set results
y_pred = predict(SVM2_classifier,  newdata = test_set[-1])

# Making the Confusion Matrix
require(caret)
```

```
cm<-confusionMatrix(data=y_pred,
                    reference=test_set$Type)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    0
##          1 3549  414
##          0  895 4030
##
##                Accuracy : 0.8527
##                  95% CI : (0.8452, 0.86)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7054
##
##  Mcnemar’s Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7986
##             Specificity : 0.9068
##          Pos Pred Value : 0.8955
##          Neg Pred Value : 0.8183
##              Prevalence : 0.5000
##          Detection Rate : 0.3993
##    Detection Prevalence : 0.4459
##       Balanced Accuracy : 0.8527
##
##        ’Positive’ Class : 1
##
```

```
Accuracy<-round(cm$overall[1],2)
Accuracy
```

```
## Accuracy
##     0.85
```

This is a great deal higher for Support Vector Machine learning method using all measures is higher than all models.

MODEL 6 - DECISION TREE

“Decision tree is a graph to represent choices and their results in form of a tree. The nodes in the graph represent an event or choice and the edges of the graph represent the decision rules or conditions. It is mostly used in Machine Learning and Data Mining applications using R.” ~ TUTORIALSPOINT

I will rerun the dataset split into the Training set and Test set without Feature Scaling, in order to plot the Decision tree graph

```
split = sample.split(NewsData$Type, SplitRatio = 0.8)
train_set = subset(NewsData, split == TRUE)
test_set = subset(NewsData, split == FALSE)
```

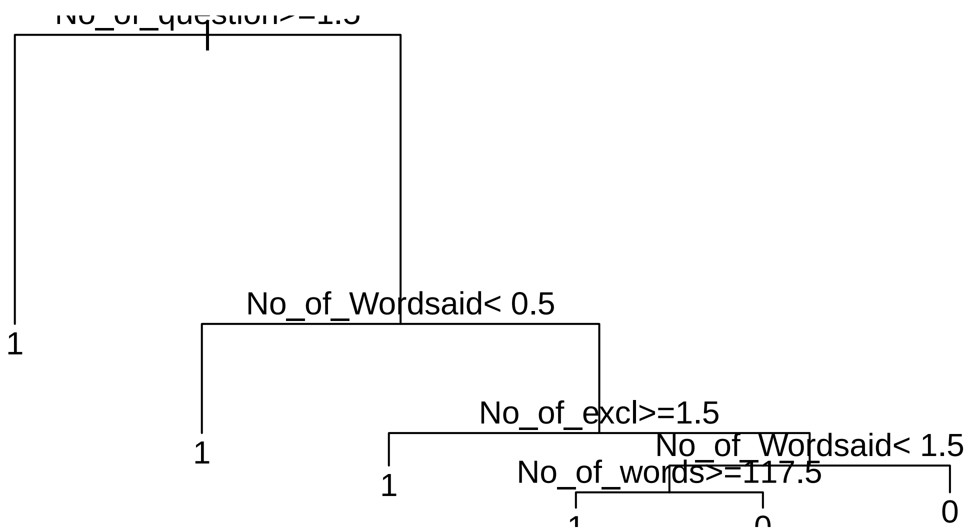
```
# Fitting SVM to the Training set and predicting the test set results
DT_classifier = rpart(formula = Type ~. ,
                   data = train_set)

# Predicting the Test set results
y_pred = predict(DT_classifier, newdata = test_set[-1],type = 'class')

# Making the Confusion Matrix
require(caret)
cm<-confusionMatrix(data=y_pred,
                    reference=test_set$Type)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    0
##          1 3978  963
##          0  466 3481
##
##                Accuracy : 0.8392
##                  95% CI : (0.8314, 0.8468)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6784
##
##  Mcnemar’s Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8951
##             Specificity : 0.7833
##          Pos Pred Value : 0.8051
##          Neg Pred Value : 0.8819
##              Prevalence : 0.5000
##          Detection Rate : 0.4476
##    Detection Prevalence : 0.5559
##       Balanced Accuracy : 0.8392
##
##        ’Positive’ Class : 1
##
```

```
plot(DT_classifier)
text(DT_classifier)
```



```
Accuracy<-round(cm$overall[1],2)
Accuracy
```

```
## Accuracy
##     0.84
```

Let us try one more classification Algorithm .

MODEL 7 - RANDOM FOREST

Building further on the decision trees is the Random Forest. It is a powerful ensembling machine learning algorithm which works by creating multiple decision trees and then combining the output generated by each of the decision trees.

```
## Warning: package 'randomForest' was built under R version 3.6.3
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
# Fitting SVM to the Training set and predicting the test set results
RF_classifier = randomForest(x = train_set[-1],
                             y = train_set$Type,
                             mtry = 6,
                             ntree = 500,
                             localImp = TRUE)
RF_classifier
```

```
##
## Call:
##  randomForest(x = train_set[-1], y = train_set$Type, ntree = 500,      mtry = 6, localImp = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 7.86%
## Confusion matrix:
##       1     0 class.error
## 1 16510  1264  0.07111511
## 0  1529 16245  0.08602453
```

By default, number of trees is 500 and number of variables tried at each split is 2 in this case. When I increased the mtry from 2 to 6, the error rate reduced from 8.25% to 8.17%. (No a huge difference)

```
# Predicting the Test set results
y_pred = predict(RF_classifier, newdata = test_set[-1])

# Making the Confusion Matrix
require(caret)
cm<-confusionMatrix(data=y_pred, reference=test_set$Type)
```

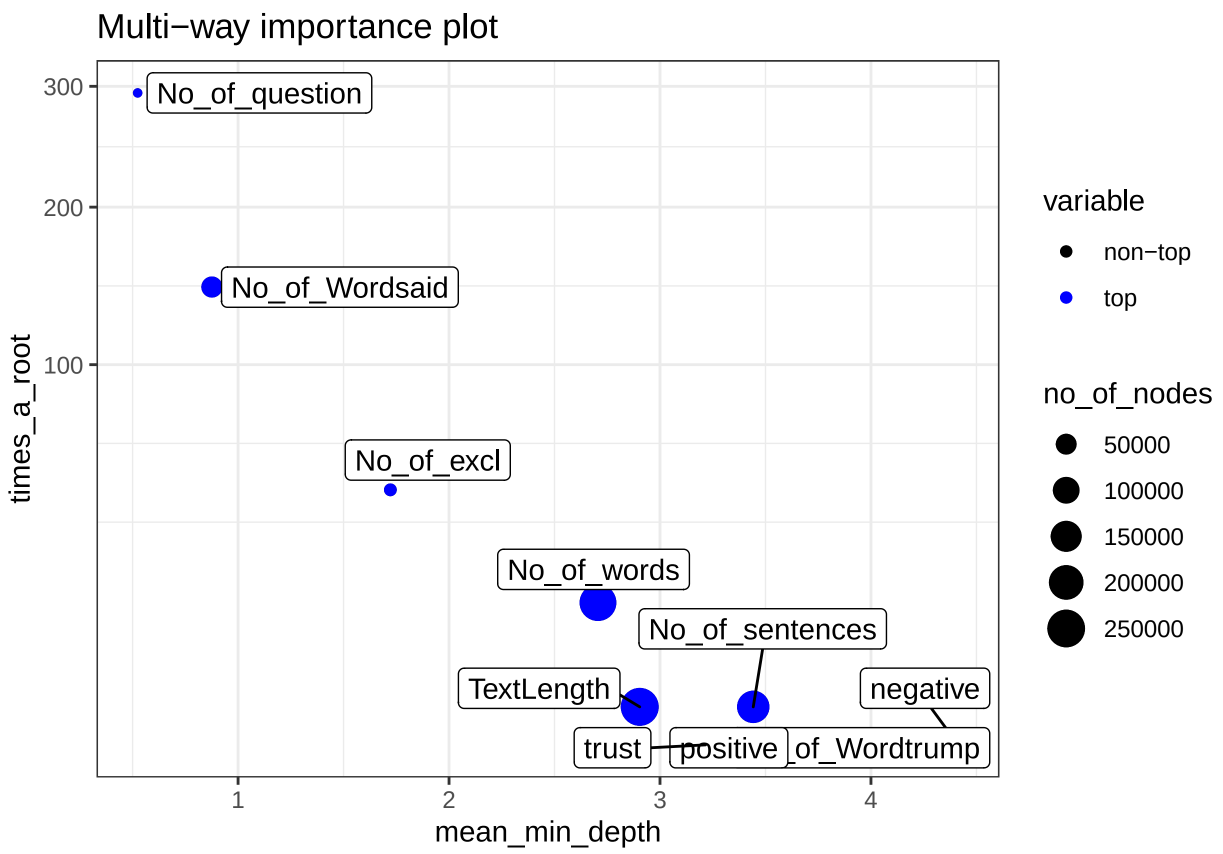
```
## Warning: package 'randomForestExplainer' was built under R version 3.6.3
```

Table 6: Importance Frame

variable	mean_min_depth	no_of_nodes	accuracy_decrease	gini_decrease	no_of_trees	times_a_root	p_value
negative	4.414	170820	0.0343248	805.2883	500	0	0
No_of_excl	1.722	14603	0.0675462	1790.6689	500	45	1
No_of_question	0.524	11279	0.0908328	3737.2324	500	294	1
No_of_sentences	3.442	168580	0.0976596	1289.6824	500	1	0
No_of_words	2.706	234166	0.1401019	1964.7937	500	14	0
No_of_Wordsaid	0.876	51890	0.1635570	4103.0284	500	145	1
No_of_Wordtrump	3.644	108789	0.0265986	710.0359	500	0	1

variable	mean_min_depth	no_of_nodes	accuracy_decrease	gini_decrease	no_of_trees	times_a_root	p_value
positive	3.448	165650	0.0448842	790.2916	500	0	0
TextLength	2.904	253706	0.1107518	1768.7050	500	1	0
trust	3.216	153245	0.0450081	811.7647	500	0	0

```
plot_multi_way_importance(importance_frame, size_measure = "no_of_nodes")
```



Observe the marked negative relation between times_a_root and mean_min_depth. Also, the superiority of “No_of_question” & “No_of_excl” as well as “No_of_Wordsaid” is clear in all three dimensions plotted. The times_a_root would fall into the category of measuring the “relative power” of a variable compared to its “competitors”. The times_a_root statistic measures the number of times a variable is “at the top” of a decision tree, i.e., how likely it is to be chosen first in the process of selecting split criteria. The no_of_node measures the number of times the variable is chosen at all as a splitting criterion among all of the sub sampled. In this situation and we can see that both of these variables have a comparable explanatory ability.

```
Accuracy<-cm$overall[1]
Accuracy
```

```
## Accuracy
## 0.909991
```

The above comparison shows the true power of ensembling and the importance of using Random Forest over Decision Trees. Though Random Forest comes up with its own limitations -eg. number of factor levels a categorical variable can have - it still is one of the best models that can be used for classification. It is easy to use and tune as compared to some of the other complex models that I have not tried in this report. Random Forest still provides a good level of accuracy..

CONCLUSION

Although the model yielded a pretty good accuracy of 90.9%, this is a limited sample dataset. I have only used a set amount of measures for the text. To further this study and analysis, it would be beneficial to gather data from different sources and on various topics as well as adding alternative dimensions such as: Author, News Channel, Website, Topic, Country or Region, etc. This can add a lot more value to a fake checking detection. Word Associations, Term Frequencies and phases as well as stemming can also be used to further the analysis on fake news.

Thank you for taking the time to read this report.

REFERENCES

https://datascienceplus.com/parsing-text-for-emotion-terms-analysis-visualization-using-r/ #:~:text=R%20offers%20the%2

https://medium.com/swlh/exploring-sentiment-analysis-a6b53b026131

https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html

https://www.csie.ntu.edu.tw/~cjlin/papers/libmf/mf_adaptive_pakdd.pdf

https://scholar.smu.edu/cgi/viewcontent.cgi?article=1036&context=datasciencereview

https://cran.r-project.org/web/packages/randomForestExplainer/vignettes/randomForestExplainer.html

https://www.tutorialspoint.com/r/r_decision_tree.htm#:~:text=Decision%20tree%20is%20a%20graph, Data%20Mining%20applications%20using%20R.