Fake news detection code output

Niveditha Mangala Venkatesha

24/04/2022

Importing Data of fake new, true new and news articles

```
library(tidyverse)

FakeNews <- read.csv("C:/Users/nived/Desktop/Masters/Statistical
learning/Project/Fake.csv")
FakeNews <- as.data.frame(FakeNews) %>% mutate(title = as.character(title),
text = as.character(text), date = as.character(date), subject =
as.character(subject))

TrueNews <- True <- read.csv("C:/Users/nived/Desktop/Masters/Statistical
learning/Project/True.csv")
TrueNews <- as.data.frame(TrueNews) %>% mutate(title = as.character(title),
text = as.character(text), date = as.character(date), subject =
as.character(subject))

news_articles <- read.csv("C:/Users/nived/Desktop/Masters/Statistical
learning/Project/news_articles.csv")
news_articles <- as.data.frame(news_articles) %>% mutate(title =
as.character(title_without_stopwords), text =
as.character(text_without_stopwords), published = as.character(published),
label = as.character(label))
```

Data Exploration & Preparation Fake and True Data sets

```
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
```

Subject and date are not needed for the analysis so will be removed

```
FakeNews <- subset(FakeNews, select = -c(date,subject) )
TrueNews <- subset(TrueNews, select = -c(date,subject) )
```

News articles Second Data Set

```
as_tibble(head(news_articles))
```

```
## # A tibble: 6 x 12
##   author  published  title   text    language site_url main_img_url  type
label
##   <chr>   <chr>      <chr>   <chr>   <chr>    <chr>    <chr>         <chr>
<chr>
## 1 Barrac~ 2016-10-2~ muslim~ print ~ english  100perc~ http://bb4sp~ bias
Real
## 2 reason~ 2016-10-2~ attorn~ attorn~ english  100perc~ http://bb4sp~ bias
Real
## 3 Barrac~ 2016-10-3~ breaki~ red st~ english  100perc~ http://bb4sp~ bias
Real
## 4 Fed Up  2016-11-0~ pin dr~ email ~ english  100perc~ http://100pe~ bias
Real
```

```
## 5 Fed Up  2016-11-0~ fantas~ email ~ english  100perc~ http://100pe~ bias
Real
## 6 Barrac~ 2016-11-0~ hillar~ print ~ english  100perc~ http://bb4sp~ bias
Real
## # ... with 3 more variables: title_without_stopwords <chr>,
## #   text_without_stopwords <chr>, hasImage <int>
```

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

```
## # A tibble: 9 x 2
##   type         NoArticles
##   <chr>             <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
```

Will only add the true articles from Data set 2 and ensure that there is an equal amount of Fake and True articles. So that there is no bias when predicting. Changing the label of Real to True articles and removing all other columns except title & text.

```
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)

TrueNews2 <- filter(news_articles, type == 'Real')
TrueNews2 <- subset(TrueNews2, select = -c(type) )
```

Concatenating TrueNews2 with TrueNews

```
TrueNews <- rbind(TrueNews, TrueNews2)

FakeNews <- FakeNews[1:22218,]
nrow(FakeNews)
```

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

```
nrow(TrueNews)
```

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

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

Now both fakenews and truenews has same number of data of 22218

Analysis of the words: Finding the most common words in fake & true. Adding title & text to same field

```
library("SnowballC")
library("wordcloud")

## Loading required package: RColorBrewer

library("RColorBrewer")
library("syuzhet")
library("ggplot2")
library(tm)

## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate

FakeNews$text <- with(FakeNews, paste(title, text))
TrueNews$text <- with(TrueNews, paste(title, text))
```

Loading the fake news data as a corpus: Fake news word cloud

```
fake_corpus = VCorpus(VectorSource(FakeNews$text))
```

Converting the text to lower case

```
fake_corpus = tm_map(fake_corpus, content_transformer(tolower))
```

Removing numbers

```
fake_corpus = tm_map(fake_corpus, removeNumbers)
```

Removing punctuation

```
fake_corpus = tm_map(fake_corpus, removePunctuation)
```

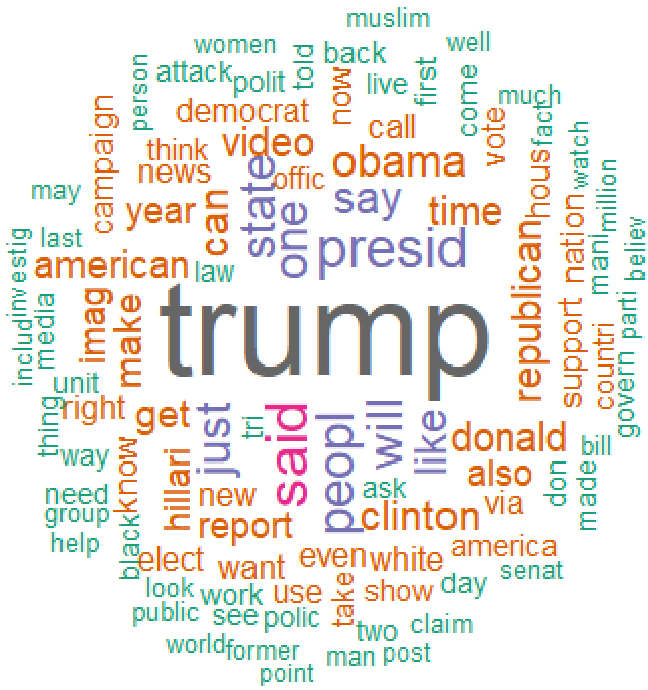
Removing English common stop words

```
fake_corpus = tm_map(fake_corpus, removeWords, stopwords())
```

Eliminating extra white spaces

```
fake_corpus = tm_map(fake_corpus, stripWhitespace)
```

Text stemming - which reduces words to their root form.



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

Loading the true news data as a corpus: True news word cloud

```
true_corpus = VCorpus(VectorSource(TrueNews$text))
```

Converting the text to lower case

```
true_corpus = tm_map(true_corpus, content_transformer(tolower))
```

Removing numbers

```
true_corpus = tm_map(true_corpus, removeNumbers)
```

Removing punctuation

```
true_corpus = tm_map(true_corpus, removePunctuation)
```

Removing Stop words

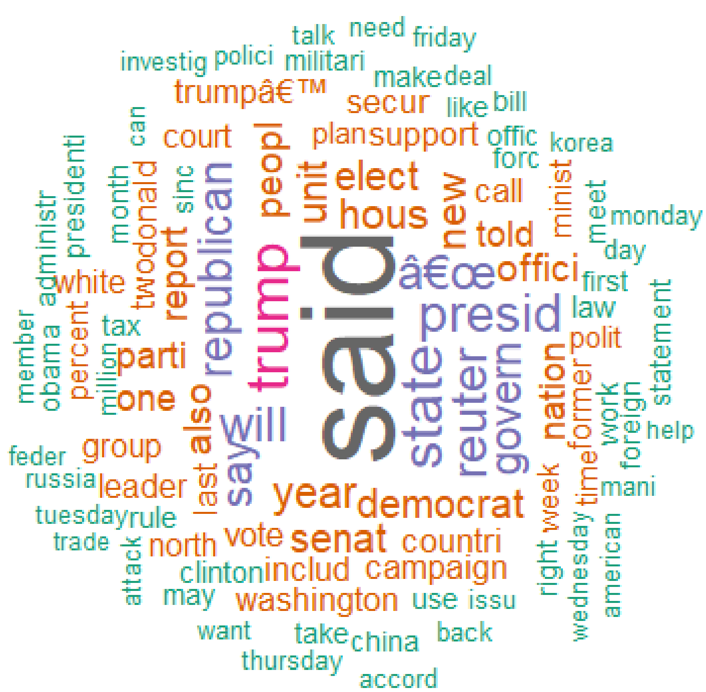
```
true_corpus = tm_map(true_corpus, removeWords, stopwords())
```

Eliminating extra white spaces

```
true_corpus = tm_map(true_corpus, stripWhitespace)
```

Text stemming

```
true_corpus = tm_map(true_corpus, stemDocument)

true_dtm = DocumentTermMatrix(true_corpus)
true_dtm = removeSparseTerms(true_dtm, 0.999)
true_dataset = as.data.frame(as.matrix(true_dtm))

#wordCloud
library(wordcloud)

true_v = sort(colSums(true_dataset),decreasing=TRUE)
myNames_true = names(true_v)
true_words = data.frame(word=myNames_true,freq=true_v,type='true')

wordcloud(words = true_words$word, freq = fake_words$freq, min.freq = 5,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
```




```
Words <- rbind(fake_words, true_words)
```

```
# Display the top 5 most frequent words
head(fake_words, 5)

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

head(true_words, 5)

##          word  freq type
## said     said 99910 true
## trump   trump 49413 true
## state   state 36651 true
## presid presid 28406 true
## reuter reuter 28306 true
```

True News: Top5 most Frequent words

	word	freq	type
said	said	99910	true
trump	trump	49413	true
state	state	36651	true
presid	presid	28406	true
reuter	reuter	28306	true

Data Transformation

A fake or truth indicator added for the prediction

```
FakeNews$Type <- c('Fake')
TrueNews$Type <- c('True')
```

Joining Fake & True Data to make a full data set.

```
NewsData <- rbind(FakeNews, TrueNews)
```

Checking for any Null Values

```
anyNA(NewsData)

## [1] FALSE
```

True Check which columns contain Null

```
colnames(NewsData)[colSums(is.na(NewsData)) > 0]

## character(0)
```

we are removing the NA values

```
NewsData <- na.omit(NewsData)
nrow(NewsData)

## [1] 44436

ncol(NewsData)

## [1] 3
```

Adding an ID

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

Removing 'Title' as we have already added it to the text field

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

Adding number of sentences per article

```
library(quanteda)

## Package version: 3.2.1
## Unicode version: 13.0
## ICU version: 69.1

## Parallel computing: 8 of 8 threads used.

## See https://quanteda.io for tutorials and examples.

##
## Attaching package: 'quanteda'

## The following object is masked from 'package:tm':
##
##     stopwords

## The following objects are masked from 'package:NLP':
##
##     meta, meta<-

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

Adding Number of characters per article

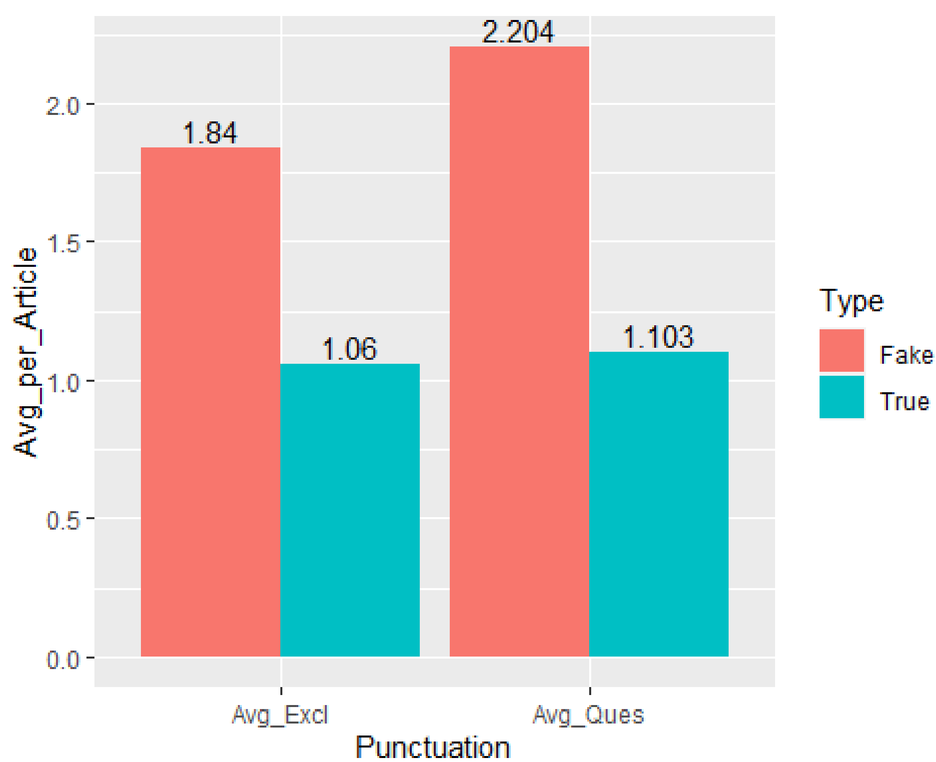
```
NewsData$TextLength <- nchar(NewsData$text)
summary(NewsData$TextLength)

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

TextLength <- NewsData$TextLength
```

Using Sapply function to calculate number of punctuation marks

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

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

Counting of exclamations & question marks in fake and true news avg in Fake and True

```
Punctuation <-NewsData %>% group_by(Type) %>%
summarise(Avg_Excl=round(mean(No_of_excl),3),

Avg_Ques=round(mean(No_of_question),3))

Punctuation <- Punctuation %>% gather("Punctuation", "Avg_per_Article", -
Type)

ggplot(Punctuation, aes(x = Punctuation, y = Avg_per_Article, fill=Type)) +
geom_col(position = "dodge") +   geom_text(aes(label=Avg_per_Article),
position=position_dodge(width=0.9), vjust=-0.25)
```




```
#removing punctuation
NewsData$text<- gsub('[[:punct:]]', '', NewsData$text)
```

Make text lower case

```
NewsData$text <- tolower(NewsData$text)
```

From the word cloud of Fake News - Calculating Number of Times ‘Trump’ Appears

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

From the word cloud of True News - Calculatin Number of Times ‘said’ Appears

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

View the all measures by type

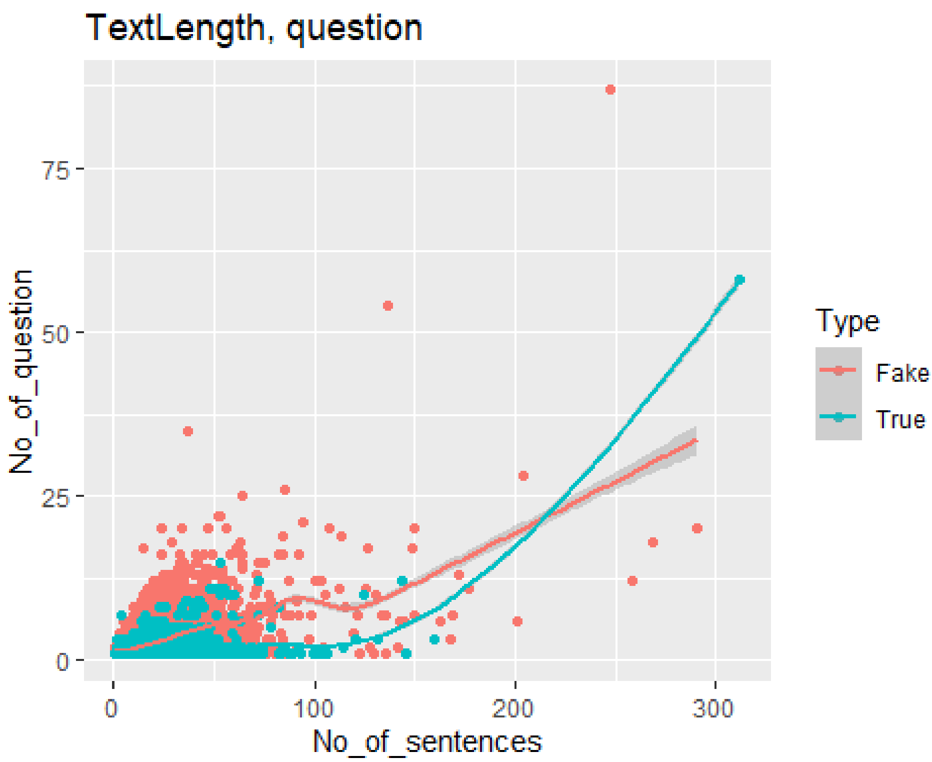
```
Data_measures <- NewsData %>% group_by(Type) %>%
  summarize(No_articles = n(),
            Avg_no_Sentences = mean(No_of_sentences),
            Avg_TextLength = mean(TextLength),
            Avg_no_excl = mean(No_of_excl),
            Avg_no_question = mean(No_of_question),
            Avg_no_trump = mean(No_of_Wordtrump),
            Avg_no_said = mean(No_of_Wordsaid))
```

Data Measures by article type

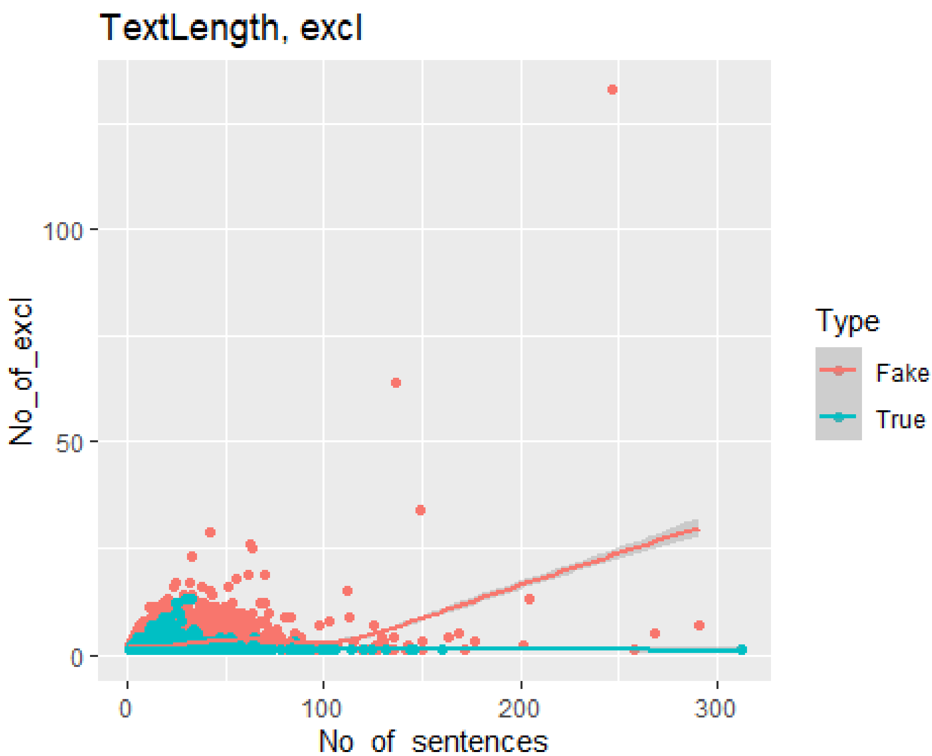
Type	No_articles	Avg_no_Sentences	Avg_TextLength	Avg_no_excl	Avg_no_question	Avg_no_trump	Avg_no_said
Fake	22218	15.57296	2519.192	1.839544	2.203844	4.120623	1.451346
True	22218	13.90904	2454.817	1.059636	1.103205	2.819651	4.505671

Correlations

```
library("ggplot2")
#library(gridExtra)

NewsData  %>%  ggplot(aes(No_of_sentences, No_of_question, color=Type)) +
geom_point() + geom_smooth() + scale_x_continuous(labels = scales::comma)+
ggtitle("TextLength, question")

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




```
NewsData  %>%  ggplot(aes(No_of_sentences, No_of_excl, color=Type)) +
geom_point() + geom_smooth() + scale_x_continuous(labels = scales::comma)+
ggtitle("TextLength, excl")

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



Remove Stop words

```
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 will be removing the duplicate white spaces between the words.

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

Adding number of words per article after removing Stop words

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

Sentiment Analysis: The study of extracted information to identify reactions, attitudes, context and emotions.

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

Taking only ID and Fake Column and combine with emotion

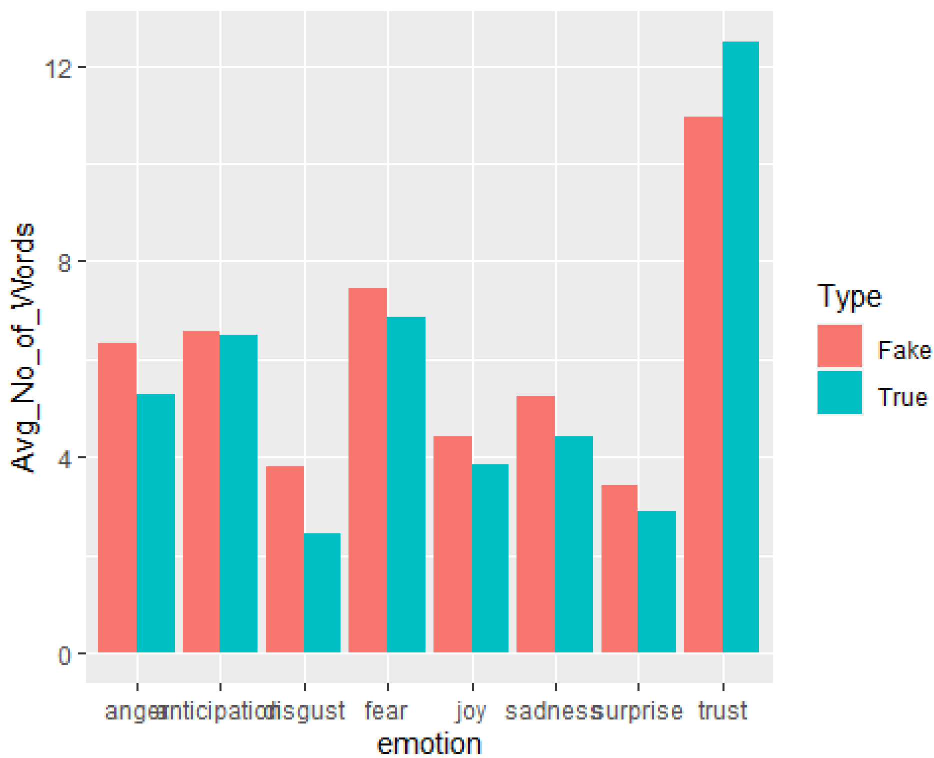
```
IDFAke <- NewsData[c(1,3)]
```

Taking only the emotions - negative & positive will be used in actual News data set

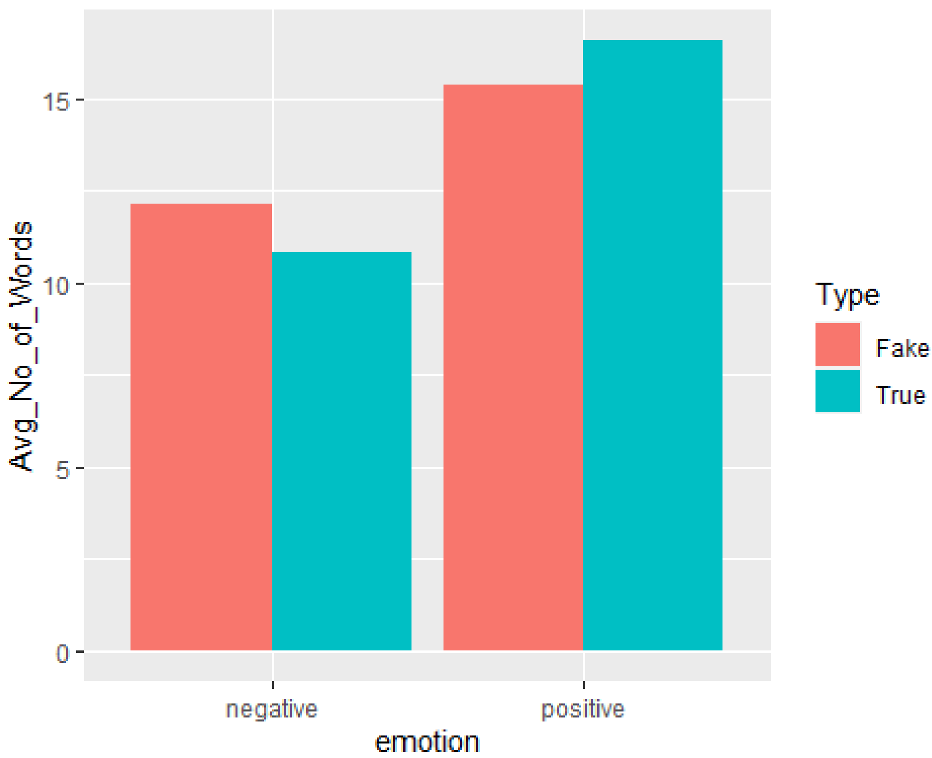
```
emotionDF <- cbind(NewsData[c(3)],emotion[c(1,2,3,4,5,6,7,8)])
emotionDF2 <- cbind(NewsData[c(3)],emotion[c(9,10)])

emotionGraph <- emotionDF %>% group_by(Type) %>%
  summarize_all((mean))
emotionGraph2 <- emotionDF2 %>% group_by(Type) %>%
  summarize_all((mean))

emotionGraph <- emotionGraph %>% gather("emotion", "Avg_No_of_Words", -Type)
emotionGraph2 <- emotionGraph2 %>% gather("emotion", "Avg_No_of_Words", -
Type)
```

Creating graph of Emotions Fake vs True Words

```
ggplot(emotionGraph, aes(x = emotion, y = Avg_No_of_Words, fill=Type)) +
geom_col(position = "dodge")
```




```
ggplot(emotionGraph2, aes(x = emotion, y = Avg_No_of_Words, fill=Type)) +
geom_col(position = "dodge")
```



Taking only negative and positive and trust for the analysis

```
emotionNegPos <- emotion[c(8,9,10)]
emotionNegPos$ID <- seq_len(nrow(emotion))
emotionNegPos <- select(emotionNegPos, ID, everything())

NewsData<-left_join(NewsData,emotionNegPos)

## Joining, by = "ID"
```

View the all additional measures by type

```
Data_measures <- NewsData %>% group_by(Type) %>% summarize(No_articles = n(),
Avg_no_Sentences = mean(No_of_sentences), Avg_TextLength = mean(TextLength),
Max_TextLength = max(TextLength), Avg_no_excl = mean(No_of_excl),
Avg_no_question = mean(No_of_question), )

Data_measures2 <- NewsData %>% group_by(Type) %>% summarize(No_articles =
n(), Avg_No_trump = mean(No_of_Wordtrump), Avg_No_said =
mean(No_of_Wordsaid), Avg_No_trust = mean(trust), Avg_No_positive =
mean(positive), Avg_No_negative = mean(negative))

Dataset <- NewsData

knitr::kable(Data_measures, caption = "Data Measures by article type")
```

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

```
knitr::kable(Data_measures2, caption = "Data Measures by article type")
```

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

Building model and training

```
anyNA(NewsData)

## [1] FALSE

NewsData <- subset(Dataset, select = -c(ID,text, NoStop_text))
```

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)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

set.seed(123)

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

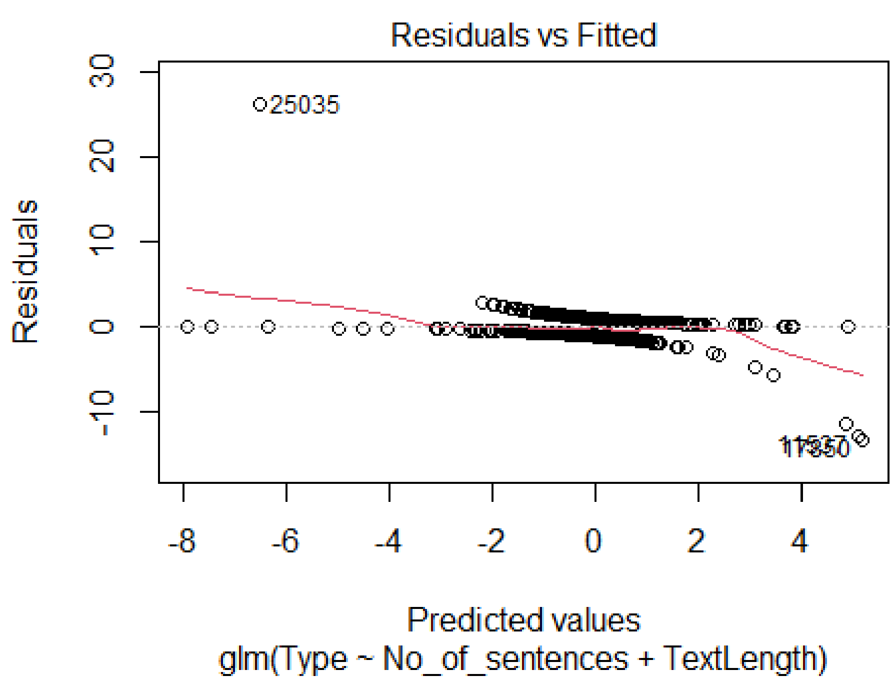
Scaling

```
train_set[,2:11] = scale(train_set[,2:11])
test_set[,2:11] = scale(test_set[,2:11])
```

MODEL 1 - Logistic Regression

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)

# Making the Confusion Matrix
require(caret)

cm = table(test_set[, 1], y_pred > 0.5)

## Warning in Ops.factor(y_pred, 0.5): '>' not meaningful for factors

cm = table(test_set[, 1], y_pred )
cm<-confusionMatrix(test_set[, 1] ,y_pred )

## Warning in confusionMatrix.default(test_set[, 1], y_pred): Levels are not
in the
## same order for reference and data. Refactoring data to match.

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

## Warning in confusionMatrix.default(data = test_set$Type, reference =
y_pred):
## Levels are not in the same order for reference and data. Refactoring data
to
## match.

test = factor(test_set$Type)
cm

## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0    1
##          0 1583 2861
##          1 2066 2378
##
##                Accuracy : 0.4457
##                  95% CI : (0.4353, 0.4561)
##     No Information Rate : 0.5894
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.1087
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.4338
##             Specificity : 0.4539
##          Pos Pred Value : 0.3562
##          Neg Pred Value : 0.5351
##              Prevalence : 0.4106
```

```
##          Detection Rate : 0.1781
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.4439
##
##        'Positive' Class : 0
##
```

```
table(y_pred, test_set[["Type"]])
```

```
##
## y_pred    1    0
##      0 2066 1583
##      1 2378 2861
```

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

```
## Accuracy
##     0.45
```

We can see from the graph that this is not a very accurate prediction with an outcome using only 2 fields No. of Sentences and Text Length, with the accuracy of 45%, lets see how this changes when adding more measures for prediction.

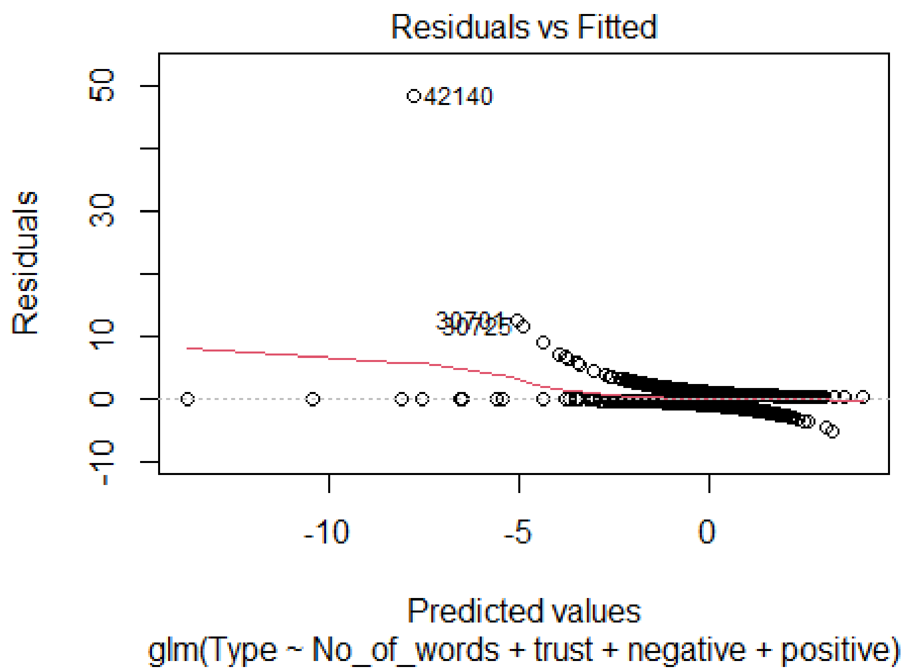
MODEL 2 - Logistic Regression 2 : No of Words & Sentiment

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
```

```
plot(classifier2, 1)
```



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, so let's use a different method of classification prediction.

MODEL 3 - SUPPORT VECTOR MACHINE

Fitting SVM to the Training set and predicting the test set results

Using the Support Vector Machine starting with only No of sentences and Text Length

```
library(e1071)

SVM_classifier = svm(formula = Type ~ No_of_sentences + TextLength, data =
```

```r
train_set, type = 'C-classification', kernel = 'linear')
SVM_classifier
```

```
##
## Call:
## svm(formula = Type ~ No_of_sentences + TextLength, data = train_set,
##     type = "C-classification", kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  34163
```

Predicting the Test set results

```r
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
```

Accuracy is at 55% which is higher than all models of Logistic Regression

MODEL 4 - SUPPORT VECTOR MACHINE 2

In this Algorithm we are using all the fields to predict if an article is True or Fake, fitting SVM to the Training set and predicting the test set results

```
SVM2_classifier = svm(formula = Type ~ .,
                      data = train_set,
                      type = 'C-classification',
                      kernel = 'linear')
SVM2_classifier

##
## Call:
## svm(formula = Type ~ ., data = train_set, type = "C-classification",
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  13763
```

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
```

Accuracy is 85% which is higher for Support Vector Machine learning method using all measures is higher than all models

MODEL 5 - DECISION TREE

Fitting Decision Tree to the Training set

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

library(rpart)
DT_classifier = rpart(formula = Type ~., data = train_set)
DT_classifier
```

```
## n= 35548
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 35548 17774 1 (0.50000000 0.50000000)
##    2) No_of_question>=1.5 10106  1168 1 (0.88442509 0.11557491) *
##    3) No_of_question< 1.5 25442  8836 0 (0.34729974 0.65270026)
##      6) No_of_Wordsaid< 0.5 5782  1424 1 (0.75371844 0.24628156) *
##      7) No_of_Wordsaid>=0.5 19660  4478 0 (0.22777213 0.77222787)
```

```
##       14) No_of_excl>=1.5 1762   444 1 (0.74801362 0.25198638) *
##       15) No_of_excl< 1.5 17898  3160 0 (0.17655604 0.82344396)
##         30) No_of_Wordsaid< 1.5 3537  1288 0 (0.36415041 0.63584959)
##           60) No_of_words>=117.5 2048   817 1 (0.60107422 0.39892578) *
##           61) No_of_words< 117.5 1489    57 0 (0.03828073 0.96171927) *
##         31) No_of_Wordsaid>=1.5 14361  1872 0 (0.13035304 0.86964696) *
```

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
##

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

## Accuracy
##     0.84
```

Accuracy is 84% for the decision tree model

MODEL 6 - RANDOM FOREST

Splitting the dataset into the Training set and Test set

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

split = sample.split(NewsData$Type, SplitRatio = 0.8)
train_set = subset(NewsData, split == TRUE)
test_set = subset(NewsData, split == FALSE)
# Feature Scaling
train_set[,2:11] = scale(train_set[,2:11])
test_set[,2:11] = scale(test_set[,2:11])
```

Fitting random forest to the Training set and predicting the test set results

```
library(randomForest)

## 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

set.seed(123)

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

# 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)

Accuracy<-cm$overall[1]
Accuracy

## Accuracy
## 0.909991
```

Accuracy is 90% for random forest model