# Data Science: Capstone - Fake New Detection Project

Shanna Jardim

01/02/2021

## INTRODUCTION

> With all that is going on in the world, fake news, conspiracy theories, over exaggerating the truth have increased. It is hard to trust mainstream media, social media, any articles. I decided to look into how I can create a Fake New Detection Algorithm.

**Fake new on the Rise** > Misleading content such as fake news and fake reviews, have increasingly become dangerous for online users. Social Media is one of the biggest sources of spreading fake new and poorly written news articles, which have a certain degree of real news but are not entirely accurate. Usually, this kind of news is designed to promote 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 write fake news 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 as there is no real way to tell if the content is fake or true. This has given way to the development of Fact checking websites that use machine learning & human research to detect fake news.

## GOAL OF PROJECT

> In this assignment, I will use a dataset from Kaggle that has already split the articles into Fake or True and train an algorithm to detect if a new article is false or fact by simply using the text of that article.

## DATA SET

I am using the following datasets which have already been split into Fake and True News. https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset

This dataset does not have a lot of columns I am therefore going to do a very simple analysis using only the title and article text to identify whether an article is fake or true adding factors discussed in the document. **Bias's to consider** Number of False articles is already higher than True. The source of articles is unknown. The author is unknown

## EXPLORING THE DATA

```
head(FakeNews)
```

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

```
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 he~ politic~ December~
## 2 U.S. military to accept tra~ WASHINGTON (Reuters) - Transg~ politic~ December~
## 3 Senior U.S. Republican sena~ WASHINGTON (Reuters) - The sp~ 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 prepa~ WEST PALM BEACH, Fla./WASHING~ politic~ December~
```

- **Observations**
- title and text can be joined into one Text field
- Will change the format of Date to year and month
- Fake or True flag must be added when combining data

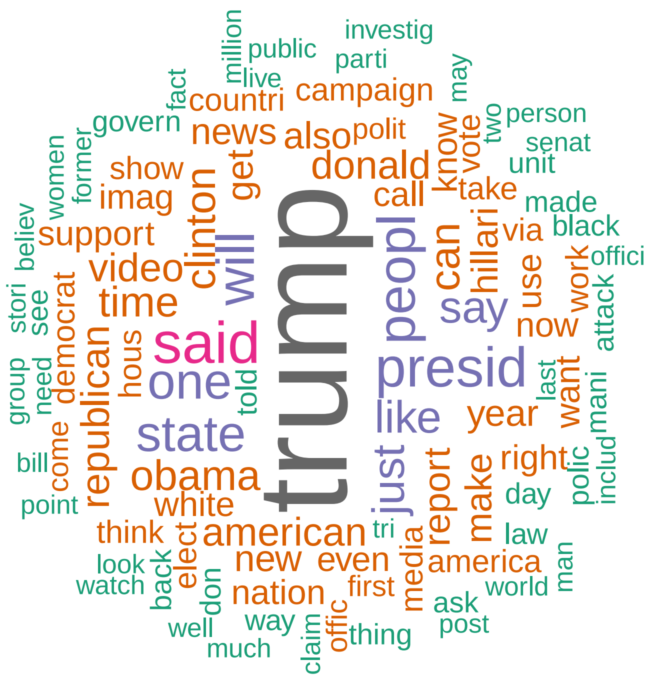
Subjects that appear in True are not in fake new dataset I feel this adds little meaning to the prediction

**Word Analysis**

Text Preprocessing - Finding the most common words in fake & true

**Fake News Words** Text data contains characters, like punctuations, stop words etc, that does not give information and increase the complexity of the analysis. So, in order to simplify our data, we remove all this noise to obtain a clean and analyzable dataset.

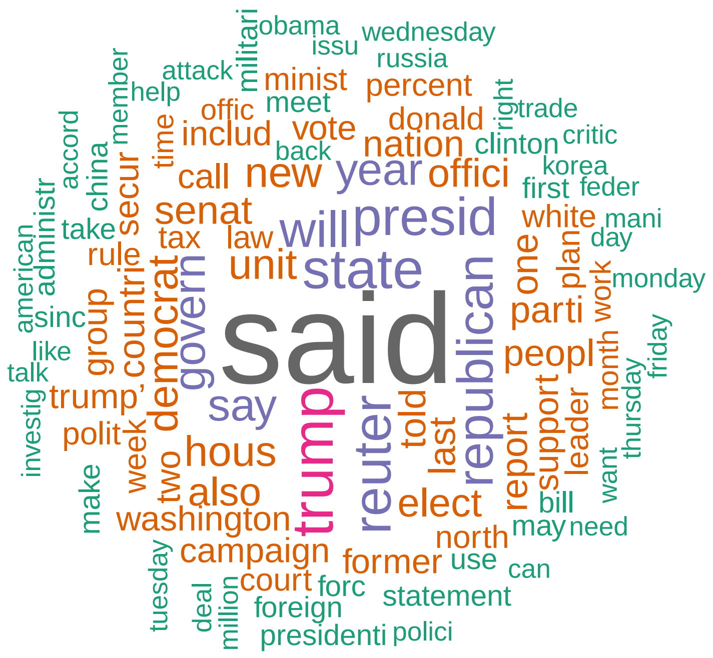
```
# Add title & text to same field
FakeNews$text <- with(FakeNews, paste(title, text))
TrueNews$text <- with(TrueNews, paste(title, text))
# Load the data as a corpus

fake_corpus = VCorpus(VectorSource(FakeNews$text))
# Convert the text to lower case
fake_corpus = tm_map(fake_corpus, content_transformer(tolower))
# Remove numbers
fake_corpus = tm_map(fake_corpus, removeNumbers)
# Remove punctuation
fake_corpus = tm_map(fake_corpus, removePunctuation)
# Remove english common stopwords
fake_corpus = tm_map(fake_corpus, removeWords, stopwords())
# Eliminate extra white spaces
fake_corpus = tm_map(fake_corpus, stripWhitespace)
# Text stemming - which reduces words to their root form.
# In my research I have noticed that this is not always the most accurate process
fake_corpus = tm_map(fake_corpus, stemDocument)


fake_dtm = DocumentTermMatrix(fake_corpus)
fake_dtm = removeSparseTerms(fake_dtm, 0.999)
fake_dataset = as.data.frame(as.matrix(fake_dtm))

#wordCloud
library(wordcloud)

fake_v = sort(colSums(fake_dataset),decreasing=TRUE)
myNames = names(fake_v)
fake_words = data.frame(word=myNames,freq=fake_v,type='fake')

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



### True News Words

```
true_corpus = VCorpus(VectorSource(TrueNews$text))
# Convert the text to lower case
true_corpus = tm_map(true_corpus, content_transformer(tolower))
# Remove numbers
true_corpus = tm_map(true_corpus, removeNumbers)
```

```r
# Remove punctuation
true_corpus = tm_map(true_corpus, removePunctuation)
# Remove Stop words - They are the most commonly occurring words in a language and
# have very little value in terms of gaining useful information.
# They should be removed before performing further analysis.
true_corpus = tm_map(true_corpus, removeWords, stopwords())
# Eliminate extra white spaces
true_corpus = tm_map(true_corpus, stripWhitespace)
# Text stemming - which reduces words to their root form.
# In my research I have noticed that this is not always the most accurate process
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"))
```



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

```
##          word  freq type
## trump   trump 81336 fake
## said     said 31205 fake
## presid presid 29069 fake
## peopl   peopl 26483 fake
## will     will 26179 fake
```

```
##          word  freq type
## said     said 99031 true
## trump   trump 47577 true
## state   state 35672 true
## reuter reuter 28366 true
## presid presid 27730 true
```

I am going to use the top 5 words in the analysis. There is a lot more analysis that can be done on the text such as - Word Associations, Term Frequencies etc. But due to time constraints I will move on.

## TRANSFORMATION OF THE DATA

A fake or truth indicator must be added as this is what we are predicting.

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

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

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

Check for any Null Values

```r
anyNA(NewsData)
```

```
## [1] FALSE
```

```r
#True Check which columns contain Null
colnames(NewsData)[colSums(is.na(NewsData)) > 0]
```

```
## character(0)
```

```r
# Considering the text column is the main column we are using for analysis & prediction we will remove
NewsData <- na.omit(NewsData)

nrow(NewsData)
```

```
## [1] 44898
```

Addition of extra information for predictions

```r
# Add an ID
NewsData$ID <-seq_len(nrow(NewsData))
NewsData <- select(NewsData, ID, everything())

# Remove Title, Date, Subject field
NewsData <- subset(NewsData, select = -c(title,date,subject) )

#Add number of sentences per article
library(quanteda)
```

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

```
## Package version: 2.1.2
```

```
## Parallel computing: 2 of 12 threads used.
```

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

```
##
## Attaching package: 'quanteda'
## The following objects are masked from 'package:tm':
##
##     as.DocumentTermMatrix, stopwords
```

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

```
## The following object is masked from 'package:utils':
##
##     View
```

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

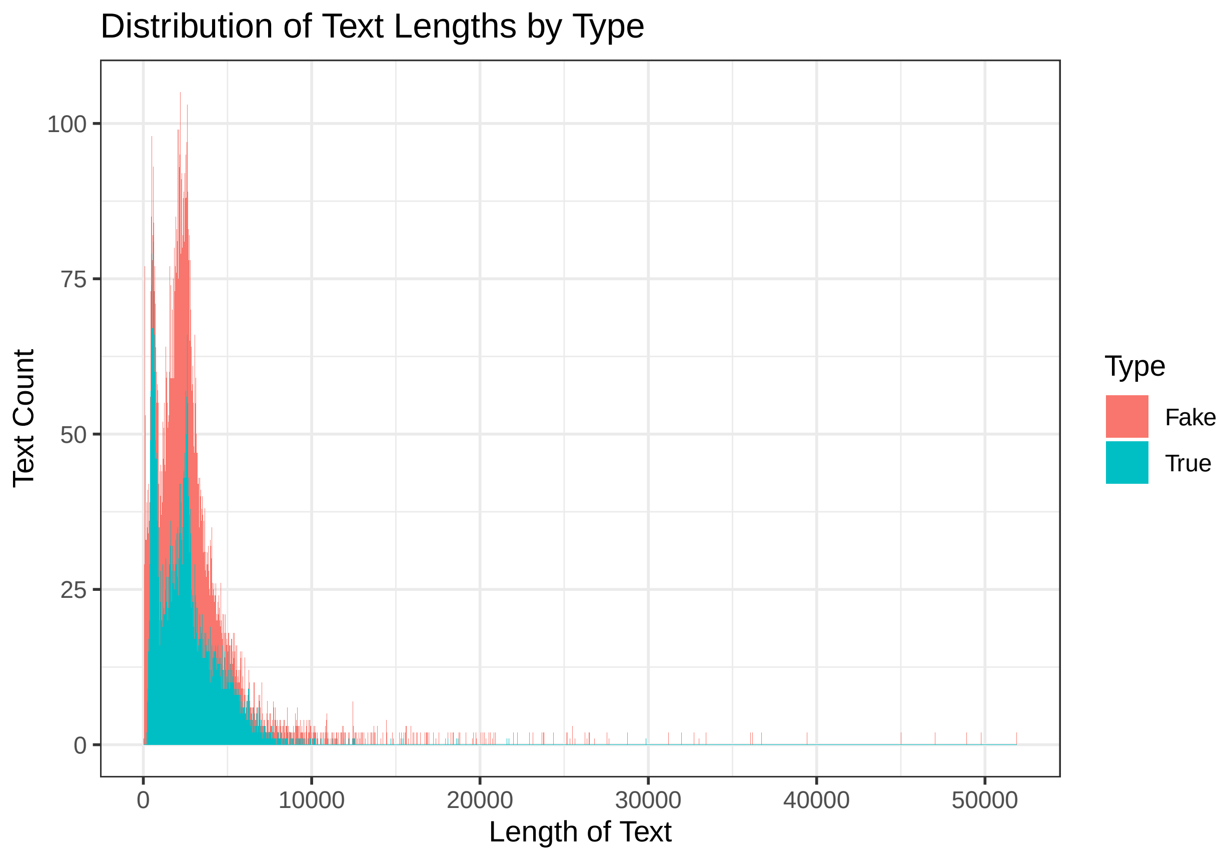
```
#Add Number of character per article
NewsData$TextLength <- nchar(NewsData$text)
summary(NewsData$TextLength)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      32    1316    2268    2549    3188   51892
```

```
TextLength <- NewsData$TextLength
TextLength[1:50]
```

```
##  [1] 2972 1967 3687 2852 2416 1823 2249 2311 2874 1725 2071 1719 2328 1670 2868
## [16] 2181 3054 2046 2712 2880 3024 2108 1995 1687 1945 2377 2105 2901 2188 2820
## [31] 2190 2600 2685 3119 1852 1740 2328 2608 4130 3351 2709 1767 2021 2383 1892
## [46] 2146 2417 1926 2187 1799
```

```
ggplot(NewsData, aes(x = TextLength, fill = Type)) +
  theme_bw() +
  geom_histogram(binwidth = 5) +
  labs(y = "Text Count", x = "Length of Text",
       title = "Distribution of Text Lengths by Type")
```



```
# Sapply function to calculate number of exclamation marks
NewsData$excl <- sapply(NewsData$text, function(x) length(unlist(strsplit(as.character(x), "\\!+"))))

# Sapply function to calculate number of question marks
NewsData$question <- sapply(NewsData$text, function(x) length(unlist(strsplit(as.character(x), "\\?+")))

##Count of exclamations & question marks in fake and true news avg in Fake and True
NewsData %>% group_by(Type) %>% summarise(Sum_Excl=sum(excl),
                                          Avg_Excl=mean(excl),
                                          Sum_Ques=sum(question),
                                          Avg_Ques=mean(question))
```

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

```
## # A tibble: 2 x 5
##   Type  Sum_Excl Avg_Excl Sum_Ques Avg_Ques
##   <chr>    <int>    <dbl>    <int>    <dbl>
## 1 Fake     42448     1.81    52214     2.22
## 2 True     22732     1.06    23683     1.11
```

```
# Make text lower case
NewsData$text <- tolower(NewsData$text) #make it lower case
NewsData$text<- gsub('[[:punct:]]', '', NewsData$text) #remove punctuation
```

**Removing Stop words from total Dataset** Remove Stop words - They are the most commonly occurring words in a language and have very little value in terms of gaining useful information. They should be removed before performing further analysis.

```r
#Add number of words per article after removing Stop words
NewsData$No_of_words <- sapply(strsplit(NewsData$text, " "), length)
```

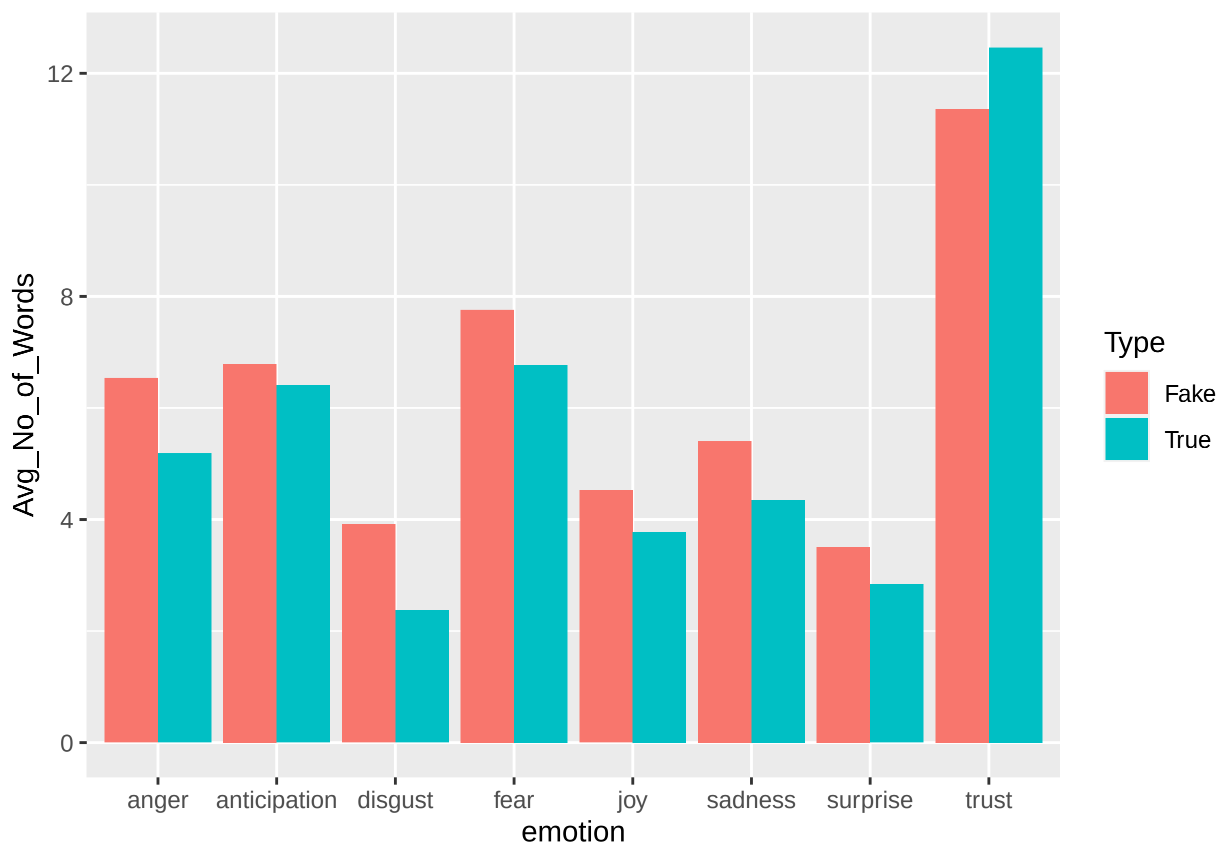
**Sentiment Analysis** Sentiment Analysis:The study of extracted information to identify reactions, attitudes, context and emotions. R offers the get_nrc_sentiment function via the Tidy or Syuzhet packages for analysis of emotion words expressed in text. Both packages implemented Saif Mohammad's NRC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

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

The last 2 columns show Negative & Postive words. Will add this to the dataset.

```r
#Take only ID and Fake Column and combine with emotion
IDFAke <- NewsData[c(1,3)]
#Take only the emotions - negative & postive will the used in actual News dataset
emotionDF <- cbind(NewsData[c(3)],emotion[c(1,2,3,4,5,6,7,8)])

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

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

```r
#Create graph of Emotions Fake vs True Words
ggplot(emotionGraph, aes(x = emotion, y = Avg_No_of_Words, fill=Type)) +
  geom_col(position = "dodge")
```



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

```
#taking only negative , positive and trust column 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"
```

```
Dataset <- NewsData
```

## MODEL BUILDING & TRAINING

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

```
# Check again for any NA data
anyNA(NewsData)
```

```
## [1] FALSE
```

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

```r
#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
# install.packages('caTools')
library(caTools)
```

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

```r
library(caret)
```

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

```
## Loading required package: lattice
```

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

```
##
## Attaching package: 'caret'
```

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

```r
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[,2:9] = scale(train_set[,2:9])
test_set[,2:9] = scale(test_set[,2:9])
```

**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 number of Sentences and Text Length only for prediction

```r
# 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
##        -0.09792          0.69951         -0.77074
##
## Degrees of Freedom: 35918 Total (i.e. Null);  35916 Residual
## Null Deviance:       49720
## Residual Deviance: 49020     AIC: 49030
```

```
summary(classifier)
```

```
##
## Call:
## glm(formula = Type ~ No_of_sentences + TextLength, family = binomial,
##     data = train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0621  -1.1401  -0.9016   1.2031   2.2315
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -0.09792    0.01068  -9.167   <2e-16 ***
## No_of_sentences  0.69951    0.03001  23.309   <2e-16 ***
## TextLength      -0.77074    0.03266 -23.597   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 49718  on 35918  degrees of freedom
## Residual deviance: 49022  on 35916  degrees of freedom
## AIC: 49028
##
## Number of Fisher Scoring iterations: 4
```

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

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

```
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    0
##          1 1496 1178
##          0 3200 3105
##
##                Accuracy : 0.5124
##                  95% CI : (0.502, 0.5228)
##     No Information Rate : 0.523
##     P-Value [Acc > NIR] : 0.9782
##
##                   Kappa : 0.0426
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.3186
##             Specificity : 0.7250
##          Pos Pred Value : 0.5595
##          Neg Pred Value : 0.4925
##              Prevalence : 0.5230
##          Detection Rate : 0.1666
##    Detection Prevalence : 0.2978
##       Balanced Accuracy : 0.5218
##
##        'Positive' Class : 1
##
```

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

```
## Accuracy
##     0.51
```

An Accuracy of 51% is quite good considering the simplicity of the measures I am using for prediction

**MODEL 2 - Logistic Regression 2**

Also using Logistic Regression but now the sentiment of the words.

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

```
##
## Call:  glm(formula = Type ~ trust + negative + positive, family = binomial,
##     data = train_set)
##
## Coefficients:
```

```
## (Intercept)        trust     negative     positive
##    -0.09952      0.96292     -0.78828     -0.24032
##
## Degrees of Freedom: 35918 Total (i.e. Null);  35915 Residual
## Null Deviance:       49720
## Residual Deviance: 47460     AIC: 47460
```

```
summary(classifier2)
```

```
##
## Call:
## glm(formula = Type ~ trust + negative + positive, family = binomial,
##     data = train_set)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.469  -1.105  -0.743   1.155   2.415
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.09952    0.01092  -9.115  < 2e-16 ***
## trust        0.96292    0.03367  28.597  < 2e-16 ***
## negative    -0.78828    0.02032 -38.792  < 2e-16 ***
## positive    -0.24032    0.03417  -7.032 2.03e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 49718  on 35918  degrees of freedom
## Residual deviance: 47455  on 35915  degrees of freedom
## AIC: 47463
##
## Number of Fisher Scoring iterations: 4
```

```
# Predicting the Test set results
prob_pred2 = predict(classifier2, type = 'response', newdata = test_set[7:9])
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)
```

```
## Warning in confusionMatrix.default(data = y_pred2, reference = test_set$Type):
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
cm
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    1    0
##          1 1380 2233
##          0 3316 2050
##
##                Accuracy : 0.382
##                  95% CI : (0.3719, 0.3921)
##     No Information Rate : 0.523
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.225
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.2939
##             Specificity : 0.4786
##          Pos Pred Value : 0.3820
##          Neg Pred Value : 0.3820
##              Prevalence : 0.5230
##          Detection Rate : 0.1537
##    Detection Prevalence : 0.4024
##       Balanced Accuracy : 0.3863
##
##        'Positive' Class : 1
##
```

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

```
## Accuracy
##     0.38
```

Less than previous

**MODEL 3 - Logistic Regression 3** Using only negative & postive

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

```
##
## Call:  glm(formula = Type ~ negative + positive, family = binomial,
##     data = train_set)
##
## Coefficients:
## (Intercept)     negative     positive
##    -0.09816     -0.69140      0.58898
##
## Degrees of Freedom: 35918 Total (i.e. Null);  35916 Residual
## Null Deviance:       49720
## Residual Deviance: 48310     AIC: 48320
```

```
summary(classifier3)
```

```
##
## Call:
## glm(formula = Type ~ negative + positive, family = binomial,
##     data = train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2074  -1.1290  -0.8163   1.1695   2.3130
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.09816    0.01078  -9.103   <2e-16 ***
## negative    -0.69140    0.01956 -35.347   <2e-16 ***
## positive     0.58898    0.01880  31.322   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 49718  on 35918  degrees of freedom
## Residual deviance: 48314  on 35916  degrees of freedom
## AIC: 48320
##
## Number of Fisher Scoring iterations: 4
```

```
# Predicting the Test set results
prob_pred3 = predict(classifier3, type = 'response', newdata = test_set[8:9])

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)
```

```
## Warning in confusionMatrix.default(data = y_pred3, reference = test_set$Type):
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    0
##          1 1521 2130
##          0 3175 2153
##
##                Accuracy : 0.4092
##                  95% CI : (0.399, 0.4194)
##     No Information Rate : 0.523
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.1716
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.3239
##             Specificity : 0.5027
##          Pos Pred Value : 0.4166
##          Neg Pred Value : 0.4041
##              Prevalence : 0.5230
##          Detection Rate : 0.1694
##    Detection Prevalence : 0.4066
##       Balanced Accuracy : 0.4133
##
##        'Positive' Class : 1
##
```

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

```
## Accuracy
##     0.41
```

**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

```r
# Fitting SVM to the Training set and predicting the test set results
library(e1071)
```

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

```r
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 3911 3546
##          0  785  737
##
##                Accuracy : 0.5177
##                  95% CI : (0.5073, 0.528)
##     No Information Rate : 0.523
##     P-Value [Acc > NIR] : 0.8473
##
##                   Kappa : 0.0051
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8328
##             Specificity : 0.1721
##          Pos Pred Value : 0.5245
##          Neg Pred Value : 0.4842
##              Prevalence : 0.5230
##          Detection Rate : 0.4356
##    Detection Prevalence : 0.8305
##       Balanced Accuracy : 0.5025
##
##        'Positive' Class : 1
##
```

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

```
## Accuracy
##     0.52
```

The Accuracy is higher than all models of Logistic Regression

**MODEL 5 - SUPPORT VECTOR MACHINE** In this final Algorithm I am going to use all the fields to predict

```
# 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 3622  321
##          0 1074 3962
##
##                Accuracy : 0.8446
##                  95% CI : (0.837, 0.8521)
##     No Information Rate : 0.523
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.691
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7713
##             Specificity : 0.9251
##          Pos Pred Value : 0.9186
##          Neg Pred Value : 0.7867
##              Prevalence : 0.5230
##          Detection Rate : 0.4034
##    Detection Prevalence : 0.4391
##       Balanced Accuracy : 0.8482
##
##        'Positive' Class : 1
##
```

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

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

Very happy with an accuracy of 84%

## CONCLUSION

Although the model yielded a great accuracy of 84%, This is a very simple dataset and I have only used simple measure of the text themselves. For future working I would add alot more data from different sources. I also believe adding alternative categories such as: Author, News Channel, Website, Topic, Country/ Region, can add a lot of value to a fake checking detection.

Thank you for taking the time to read my 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*