601 Individual Assignment

Olalekan Fagbuyi

2023-10-10

```
setwd('C:\\Users\\User\\Desktop\\WLU')
```

1.Business Understanding

This project aims to predict the popularity of online articles using the number of shares as a metric for measuring popularity. Correctly classifying these online articles is important to the company because it implies we will be able to optimize revenue regeneration ($0.75 for the first 1000 shares and $2 after) considering our 12 articles per day limit constraint.

A threshold of 1400 has been set to determine which articles are popular and which are not. Shares Value Range: Number of Instances in Range: < 1400 18490 (46.64%) and >= 1400 21154 (53.35%).

2. Data Understanding

The dataset contain 39644 articles with 61 attributes will be analyzed and then used to build a machine model for making predictions.

Dataset Breakdown: 61 attributes (58 predictive attributes, 2 non-predictive, 1 goal field)

2.1 Importing Libraries and Loading Dataset

```
#importing library
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✔ dplyr     1.1.3     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ lubridate 1.9.2     ✔ tibble    3.2.1
## ✔ purrr     1.0.2     ✔ tidyr     1.3.0

## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
```

```
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors

library(randomForest)

## randomForest 4.7-1.1
## 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

library(corrplot)

## corrplot 0.92 loaded

library(rpart)
library(kernlab)

##
## Attaching package: 'kernlab'
##
## The following object is masked from 'package:purrr':
##
##     cross
##
## The following object is masked from 'package:ggplot2':
##
##     alpha

library(pROC)

## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library(gridExtra)

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

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

```r
#loading dataset
articles <- read.csv("OnlineNewsPopularity.csv")
head(articles)
```

```
##                                                              url timedelta
## 1   http://mashable.com/2013/01/07/amazon-instant-video-browser/       731
## 2    http://mashable.com/2013/01/07/ap-samsung-sponsored-tweets/       731
## 3 http://mashable.com/2013/01/07/apple-40-billion-app-downloads/       731
## 4       http://mashable.com/2013/01/07/astronaut-notre-dame-bcs/       731
## 5               http://mashable.com/2013/01/07/att-u-verse-apps/       731
## 6               http://mashable.com/2013/01/07/beewi-smart-toys/       731
##   n_tokens_title n_tokens_content n_unique_tokens n_non_stop_words
## 1             12              219       0.6635945                1
## 2              9              255       0.6047431                1
## 3              9              211       0.5751295                1
## 4              9              531       0.5037879                1
## 5             13             1072       0.4156456                1
## 6             10              370       0.5598886                1
##   n_non_stop_unique_tokens num_hrefs num_self_hrefs num_imgs num_videos
## 1                0.8153846         4              2        1          0
## 2                0.7919463         3              1        1          0
## 3                0.6638655         3              1        1          0
## 4                0.6656347         9              0        1          0
## 5                0.5408895        19             19       20          0
## 6                0.6981982         2              2        0          0
##   average_token_length num_keywords data_channel_is_lifestyle
## 1             4.680365            5                         0
## 2             4.913725            4                         0
## 3             4.393365            6                         0
## 4             4.404896            7                         0
## 5             4.682836            7                         0
## 6             4.359459            9                         0
##   data_channel_is_entertainment data_channel_is_bus data_channel_is_socmed
## 1                             1                   0                      0
## 2                             0                   1                      0
## 3                             0                   1                      0
## 4                             1                   0                      0
## 5                             0                   0                      0
## 6                             0                   0                      0
##   data_channel_is_tech data_channel_is_world kw_min_min kw_max_min
kw_avg_min
## 1                    0                     0          0          0
0
```

```
## 2                    0                     0          0          0
0
## 3                    0                     0          0          0
0
## 4                    0                     0          0          0
0
## 5                    1                     0          0          0
0
## 6                    1                     0          0          0
0
##   kw_min_max kw_max_max kw_avg_max kw_min_avg kw_max_avg kw_avg_avg
## 1          0          0          0          0          0          0
## 2          0          0          0          0          0          0
## 3          0          0          0          0          0          0
## 4          0          0          0          0          0          0
## 5          0          0          0          0          0          0
## 6          0          0          0          0          0          0
##   self_reference_min_shares self_reference_max_shares
## 1                       496                       496
## 2                         0                         0
## 3                       918                       918
## 4                         0                         0
## 5                       545                     16000
## 6                      8500                      8500
##   self_reference_avg_sharess weekday_is_monday weekday_is_tuesday
## 1                    496.000                 1                  0
## 2                      0.000                 1                  0
## 3                    918.000                 1                  0
## 4                      0.000                 1                  0
## 5                   3151.158                 1                  0
## 6                   8500.000                 1                  0
##   weekday_is_wednesday weekday_is_thursday weekday_is_friday
## 1                    0                   0                 0
## 2                    0                   0                 0
## 3                    0                   0                 0
## 4                    0                   0                 0
## 5                    0                   0                 0
## 6                    0                   0                 0
##   weekday_is_saturday weekday_is_sunday is_weekend     LDA_00     LDA_01
## 1                   0                 0          0 0.50033120 0.37827893
## 2                   0                 0          0 0.79975569 0.05004668
## 3                   0                 0          0 0.21779229 0.03333446
## 4                   0                 0          0 0.02857322 0.41929964
## 5                   0                 0          0 0.02863281 0.02879355
## 6                   0                 0          0 0.02224528 0.30671758
##       LDA_02     LDA_03     LDA_04 global_subjectivity
## 1 0.04000468 0.04126265 0.04012254           0.5216171
## 2 0.05009625 0.05010067 0.05000071           0.3412458
## 3 0.03335142 0.03333354 0.68218829           0.7022222
## 4 0.49465083 0.02890472 0.02857160           0.4298497
```

```
## 5 0.02857518 0.02857168 0.88542678           0.5135021
## 6 0.02223128 0.02222429 0.62658158           0.4374086
##   global_sentiment_polarity global_rate_positive_words
## 1                0.09256198                 0.04566210
## 2                0.14894781                 0.04313725
## 3                0.32333333                 0.05687204
## 4                0.10070467                 0.04143126
## 5                0.28100348                 0.07462687
## 6                0.07118419                 0.02972973
##   global_rate_negative_words rate_positive_words rate_negative_words
## 1                0.013698630           0.7692308           0.2307692
## 2                0.015686275           0.7333333           0.2666667
## 3                0.009478673           0.8571429           0.1428571
## 4                0.020715631           0.6666667           0.3333333
## 5                0.012126866           0.8602151           0.1397849
## 6                0.027027027           0.5238095           0.4761905
##   avg_positive_polarity min_positive_polarity max_positive_polarity
## 1             0.3786364            0.10000000                   0.7
## 2             0.2869146            0.03333333                   0.7
## 3             0.4958333            0.10000000                   1.0
## 4             0.3859652            0.13636364                   0.8
## 5             0.4111274            0.03333333                   1.0
## 6             0.3506100            0.13636364                   0.6
##   avg_negative_polarity min_negative_polarity max_negative_polarity
## 1            -0.3500000                -0.600            -0.2000000
## 2            -0.1187500                -0.125            -0.1000000
## 3            -0.4666667                -0.800            -0.1333333
## 4            -0.3696970                -0.600            -0.1666667
## 5            -0.2201923                -0.500            -0.0500000
## 6            -0.1950000                -0.400            -0.1000000
##   title_subjectivity title_sentiment_polarity abs_title_subjectivity
## 1          0.5000000               -0.1875000             0.00000000
## 2          0.0000000                0.0000000             0.50000000
## 3          0.0000000                0.0000000             0.50000000
## 4          0.0000000                0.0000000             0.50000000
## 5          0.4545455                0.1363636             0.04545455
## 6          0.6428571                0.2142857             0.14285714
##   abs_title_sentiment_polarity shares
## 1                    0.1875000    593
## 2                    0.0000000    711
## 3                    0.0000000   1500
## 4                    0.0000000   1200
## 5                    0.1363636    505
## 6                    0.2142857    855

## 'data.frame':    39644 obs. of  61 variables:
##  $ url                          : chr
"http://mashable.com/2013/01/07/amazon-instant-video-browser/"
"http://mashable.com/2013/01/07/ap-samsung-sponsored-tweets/"
"http://mashable.com/2013/01/07/apple-40-billion-app-downloads/"
```

```
"http://mashable.com/2013/01/07/astronaut-notre-dame-bcs/" ...
##  $ timedelta                    : num  731 731 731 731 731 731 731 731 731
731 ...
##  $ n_tokens_title               : num  12 9 9 9 13 10 8 12 11 10 ...
##  $ n_tokens_content             : num  219 255 211 531 1072 ...
##  $ n_unique_tokens              : num  0.664 0.605 0.575 0.504 0.416 ...
##  $ n_non_stop_words             : num  1 1 1 1 1 ...
##  $ n_non_stop_unique_tokens     : num  0.815 0.792 0.664 0.666 0.541 ...
##  $ num_hrefs                    : num  4 3 3 9 19 2 21 20 2 4 ...
##  $ num_self_hrefs               : num  2 1 1 0 19 2 20 20 0 1 ...
##  $ num_imgs                     : num  1 1 1 1 20 0 20 20 0 1 ...
##  $ num_videos                   : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ average_token_length         : num  4.68 4.91 4.39 4.4 4.68 ...
##  $ num_keywords                 : num  5 4 6 7 7 9 10 9 7 5 ...
##  $ data_channel_is_lifestyle    : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ data_channel_is_entertainment: num  1 0 0 1 0 0 0 0 0 0 ...
##  $ data_channel_is_bus          : num  0 1 1 0 0 0 0 0 0 0 ...
##  $ data_channel_is_socmed       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ data_channel_is_tech         : num  0 0 0 0 1 1 0 1 1 0 ...
##  $ data_channel_is_world        : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ kw_min_min                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_max_min                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_avg_min                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_min_max                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_max_max                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_avg_max                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_min_avg                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_max_avg                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ kw_avg_avg                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ self_reference_min_shares    : num  496 0 918 0 545 8500 545 545 0 0
...
##  $ self_reference_max_shares    : num  496 0 918 0 16000 8500 16000 16000
0 0 ...
##  $ self_reference_avg_sharess   : num  496 0 918 0 3151 ...
##  $ weekday_is_monday            : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ weekday_is_tuesday           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_wednesday         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_thursday          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_friday            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_saturday          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_sunday            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ is_weekend                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ LDA_00                       : num  0.5003 0.7998 0.2178 0.0286 0.0286
...
##  $ LDA_01                       : num  0.3783 0.05 0.0333 0.4193 0.0288
...
##  $ LDA_02                       : num  0.04 0.0501 0.0334 0.4947 0.0286
...
##  $ LDA_03                       : num  0.0413 0.0501 0.0333 0.0289 0.0286
...
```

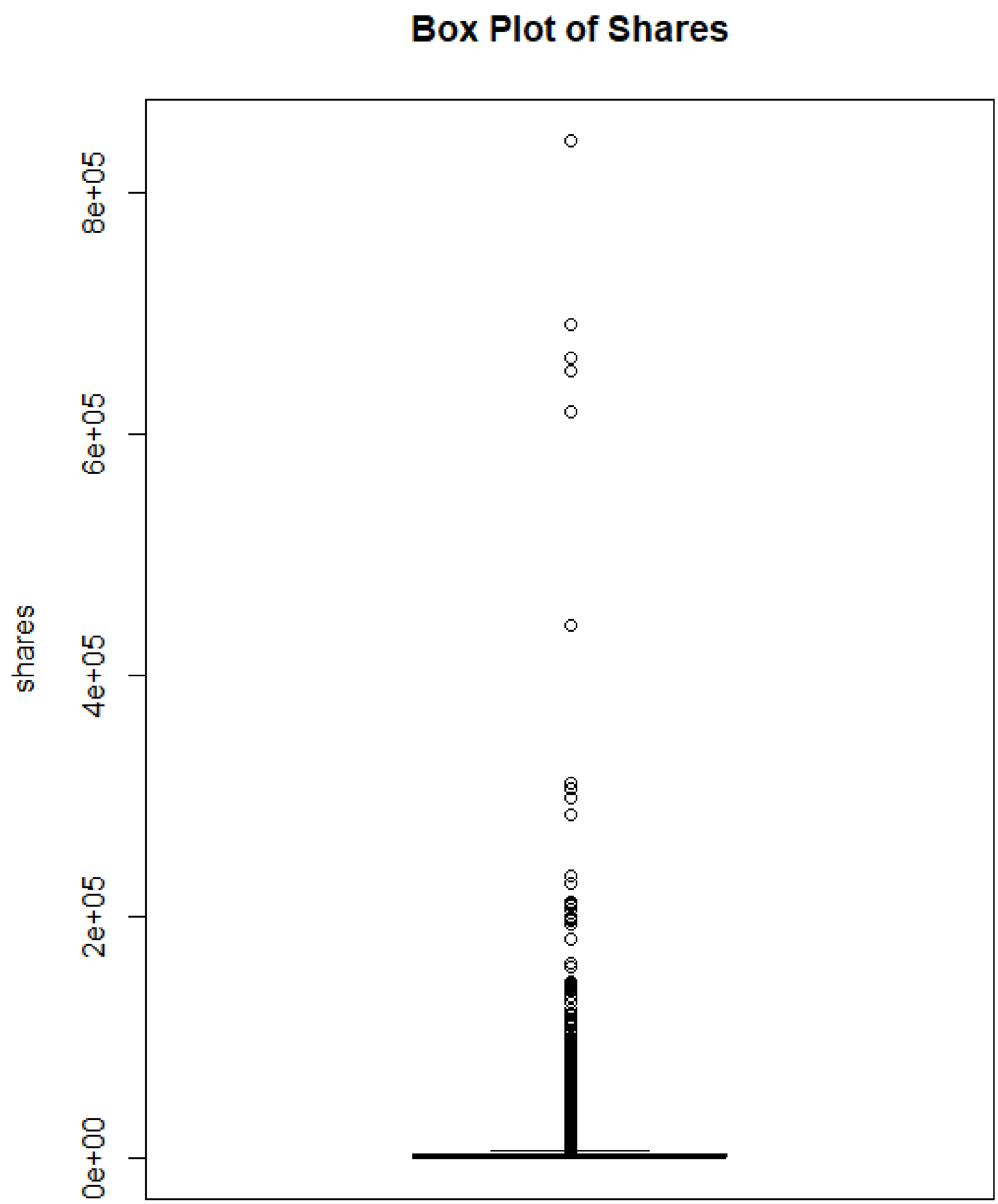
```
##  $ LDA_04                       : num  0.0401 0.05 0.6822 0.0286 0.8854
...
##  $ global_subjectivity          : num  0.522 0.341 0.702 0.43 0.514 ...
##  $ global_sentiment_polarity    : num  0.0926 0.1489 0.3233 0.1007 0.281
...
##  $ global_rate_positive_words   : num  0.0457 0.0431 0.0569 0.0414 0.0746
...
##  $ global_rate_negative_words   : num  0.0137 0.01569 0.00948 0.02072
0.01213 ...
##  $ rate_positive_words          : num  0.769 0.733 0.857 0.667 0.86 ...
##  $ rate_negative_words          : num  0.231 0.267 0.143 0.333 0.14 ...
##  $ avg_positive_polarity        : num  0.379 0.287 0.496 0.386 0.411 ...
##  $ min_positive_polarity        : num  0.1 0.0333 0.1 0.1364 0.0333 ...
##  $ max_positive_polarity        : num  0.7 0.7 1 0.8 1 0.6 1 1 0.8 0.5 ...
##  $ avg_negative_polarity        : num  -0.35 -0.119 -0.467 -0.37 -0.22 ...
##  $ min_negative_polarity        : num  -0.6 -0.125 -0.8 -0.6 -0.5 -0.4 -
0.5 -0.5 -0.125 -0.5 ...
##  $ max_negative_polarity        : num  -0.2 -0.1 -0.133 -0.167 -0.05 ...
##  $ title_subjectivity           : num  0.5 0 0 0 0.455 ...
##  $ title_sentiment_polarity     : num  -0.188 0 0 0 0.136 ...
##  $ abs_title_subjectivity       : num  0 0.5 0.5 0.5 0.0455 ...
##  $ abs_title_sentiment_polarity : num  0.188 0 0 0 0.136 ...
##  $ shares                       : int  593 711 1500 1200 505 855 556 891
3600 710 ...
```

2.2 Determining Popularity Threshold

In order to predict popularity of articles, an average threshold will be determined from the shares column.

The mean would be used if the column is normally distributed with no outliers. However, if there are outliers, the median of the column will be picked as this measure is less sensitive to extreme values.

```r
# Create a box plot for a specific column
boxplot(articles$shares, main = "Box Plot of Shares",
        ylab = "shares")
```




```
#No of outliers in the shares column
outliers <- boxplot.stats(articles$shares)$out
length(outliers)

## [1] 4541
```

The shares column has a considerable amount of outliers (4541 or 11.45% of total data points). The median will be used as an average instead of the mean on this occasion.

```
median(articles$shares)

## [1] 1400
```

2.3 Creating popularity column based on Threshold

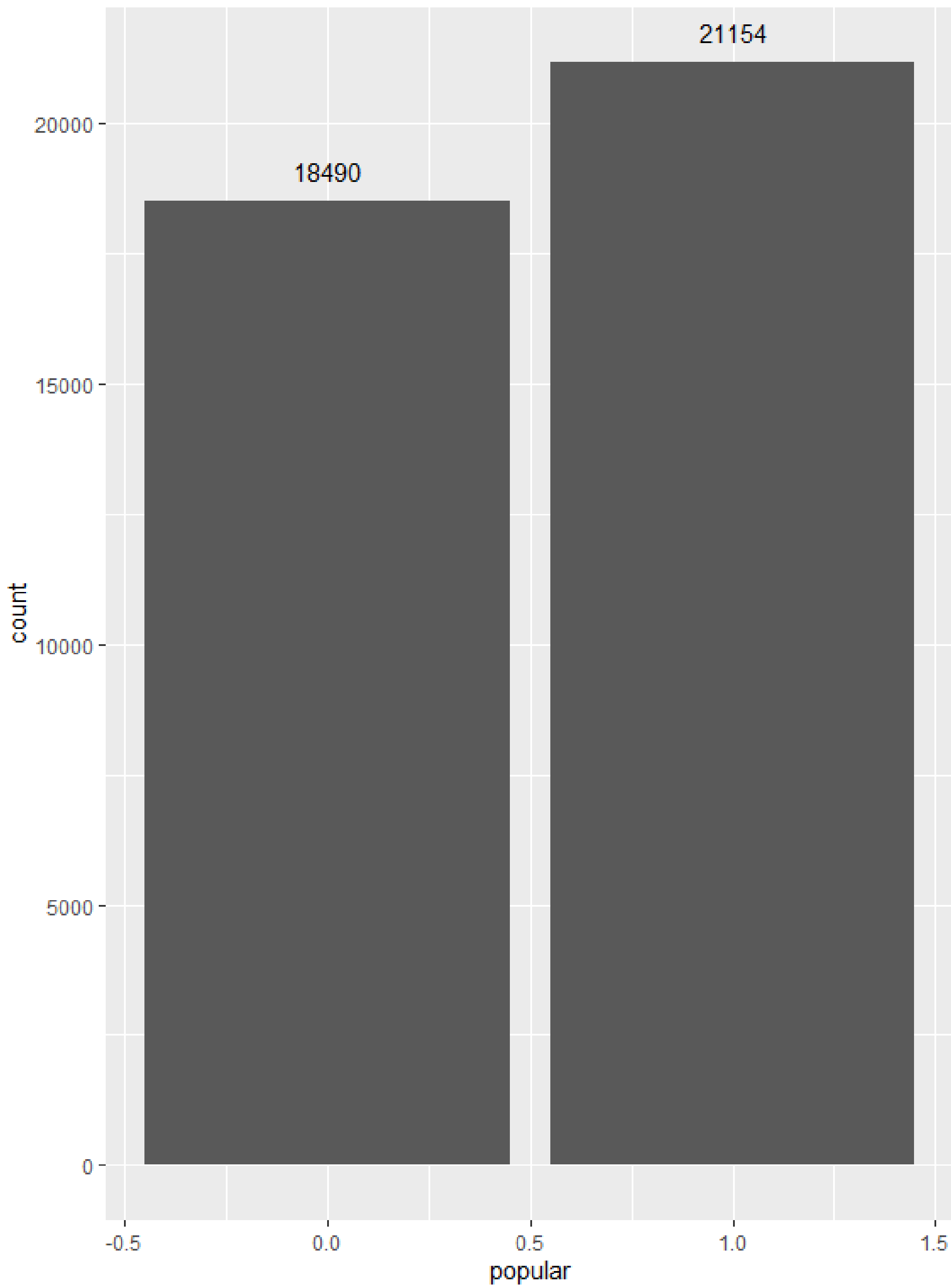
```
articles$popular <- ifelse(articles$shares >= 1400, 1, 0)

#Dropping shares column
articles1 <- subset( articles, select = -c(shares) )
```

3 Data Visualization

Getting a better understanding of the dataset via visualizations

3.1 Popularity Distribution

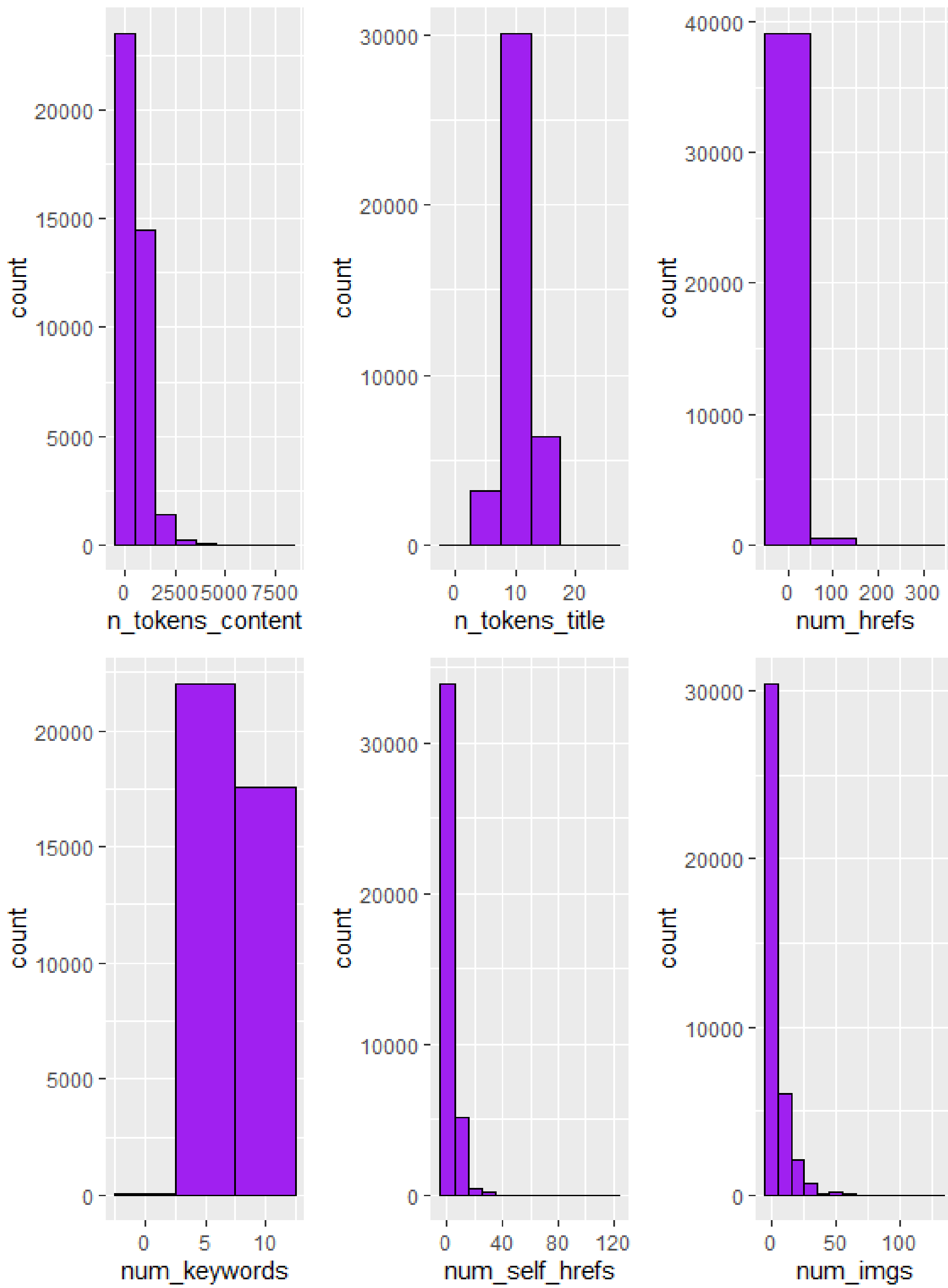
```
ggplot(articles1, aes(x = popular)) +
geom_bar() +
geom_text(stat='count', aes(label=..count..), vjust=-1)

## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2
3.4.0.
## ℹ Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



3.2 - Univariate Analysis - num columns

```
p1 <- ggplot(articles1) + geom_histogram(aes(n_tokens_content),binwidth =
1000, fill ="purple",col ="black")
p2 <- ggplot(articles1) + geom_histogram(aes(n_tokens_title), binwidth =5,
fill ="purple",col ="black")
```

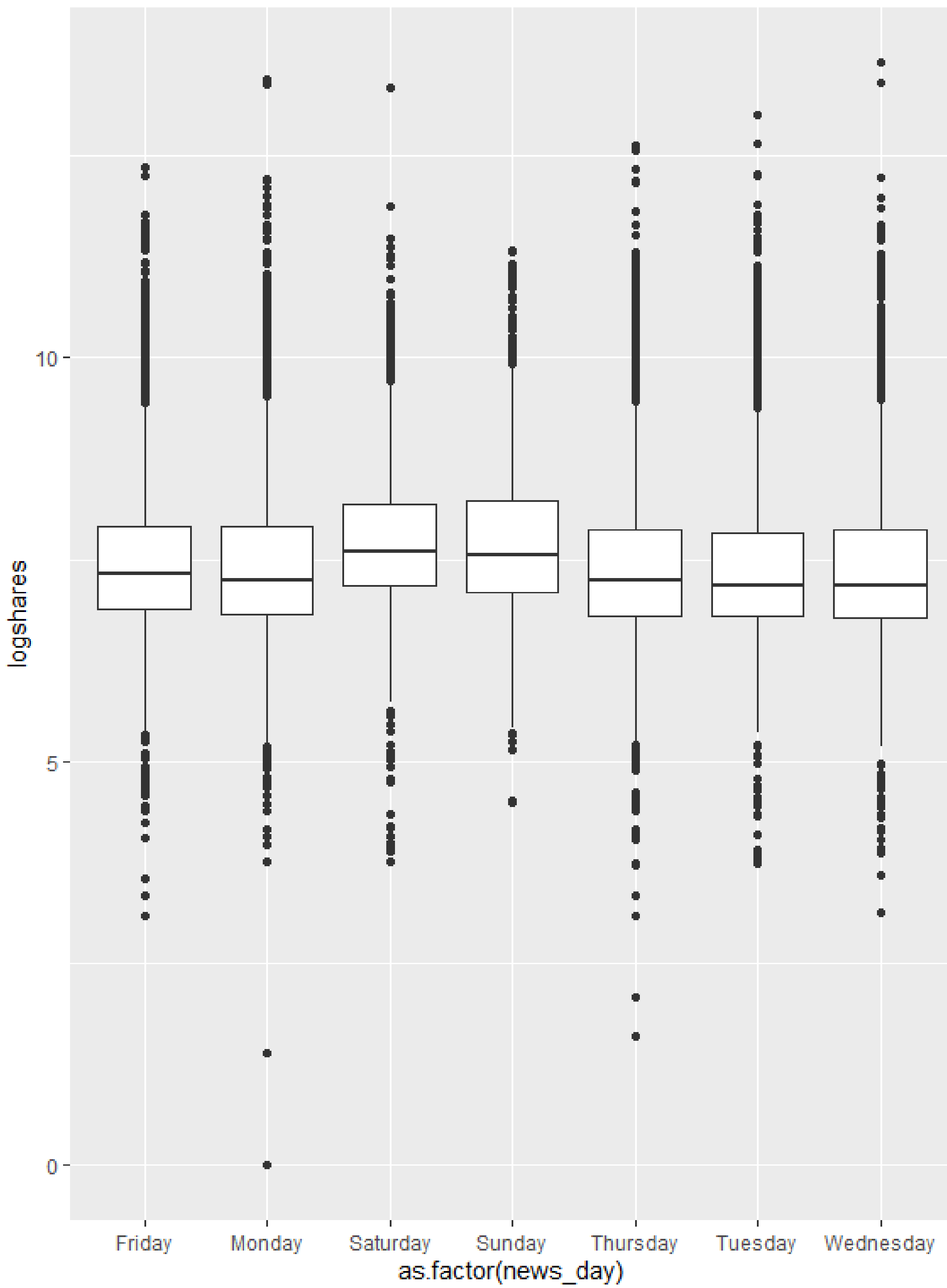
```
p3 <- ggplot(articles1) + geom_histogram(aes(num_hrefs), binwidth = 100, fill
="purple",col ="black")
p4 <- ggplot(articles1) + geom_histogram(aes(num_keywords ), binwidth =5,
fill ="purple",col ="black")
p5 <- ggplot(articles1) + geom_histogram(aes(num_self_hrefs), binwidth = 10,
fill ="purple",col ="black")
p6 <- ggplot(articles1) + geom_histogram (aes(num_imgs), binwidth = 10, fill
="purple",col ="black")

grid.arrange(p1, p2, p3, p4, p5, p6, nrow =2, ncol =3)
```



3.3 Bivariate Analysis - Checking effect of days and news type on shares

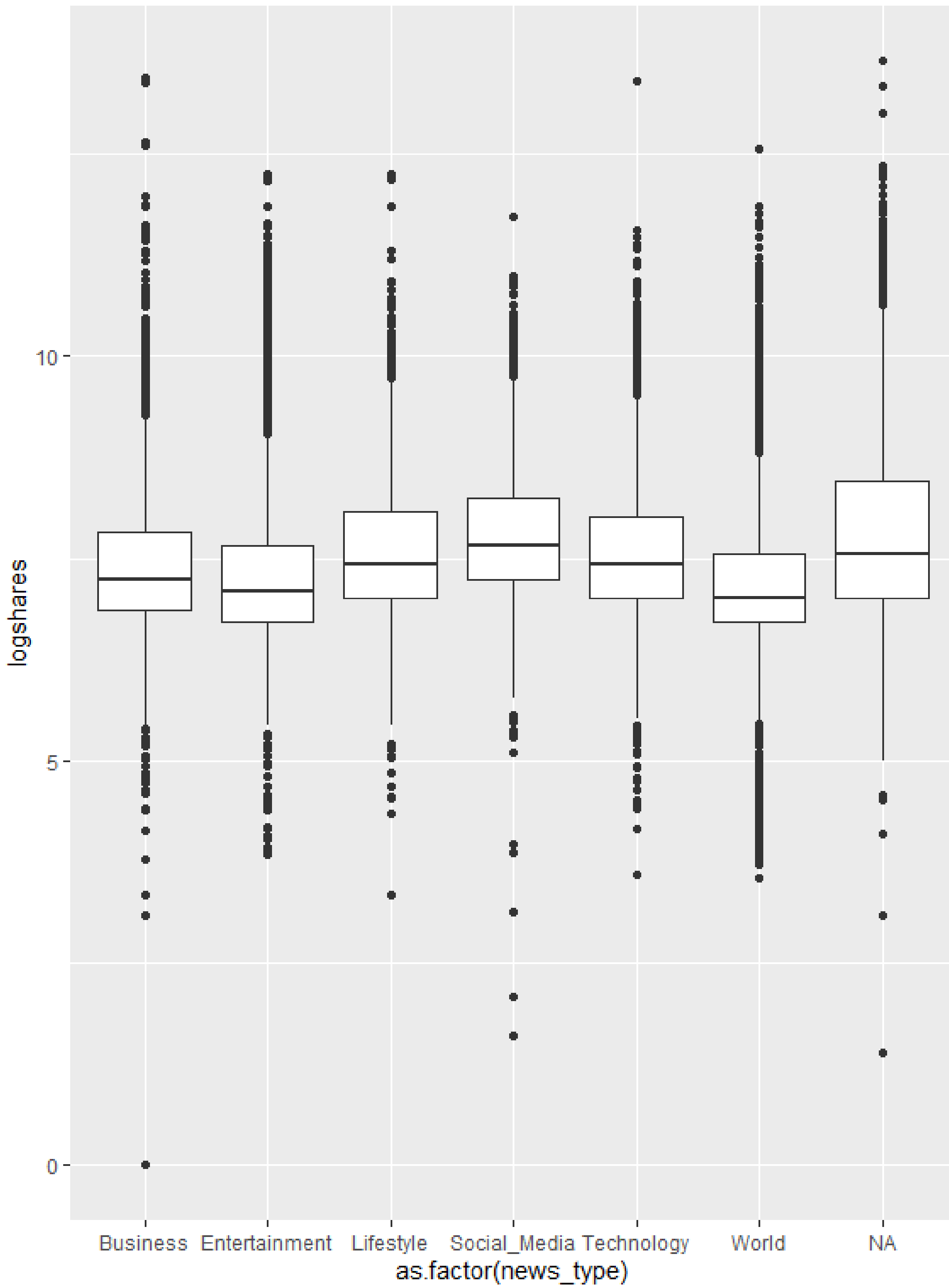
```
#Taking log of shares column to rescale column for visualizations

articles$logshares=log(articles$shares)
```

```r
#Checking if publishing days make a difference
articles$news_day[articles$weekday_is_monday==1] <- "Monday"
articles$news_day[articles$weekday_is_tuesday==1] <- "Tuesday"
articles$news_day[articles$weekday_is_wednesday==1] <- "Wednesday"
articles$news_day[articles$weekday_is_thursday==1] <- "Thursday"
articles$news_day[articles$weekday_is_friday==1] <- "Friday"
articles$news_day[articles$weekday_is_saturday==1] <- "Saturday"
articles$news_day[articles$weekday_is_sunday==1] <- "Sunday"
#Check
p1 <- ggplot(articles, aes(as.factor(news_day), logshares))
p1 + geom_boxplot()
```




```
#Checking if publishing topics make a difference
articles$news_type[articles$data_channel_is_lifestyle==1] <- "Lifestyle"
articles$news_type[articles$data_channel_is_entertainment==1] <-
"Entertainment"
articles$news_type[articles$ data_channel_is_bus==1] <- "Business"
articles$news_type[articles$data_channel_is_socmed==1] <- "Social_Media"
```

```r
articles$news_type[articles$data_channel_is_tech==1] <- "Technology"
articles$news_type[articles$data_channel_is_world==1] <- "World"

p2 <- ggplot(articles, aes(as.factor(news_type), logshares))
p2 + geom_boxplot()
```



4. Data Preparation

4.1 Checking for missing values and duplicates

```
#missing values check
sapply(articles1,function(x) sum(is.na(x)))
```

```
##                            url                      timedelta
##                              0                              0
##                 n_tokens_title               n_tokens_content
##                              0                              0
##                n_unique_tokens               n_non_stop_words
##                              0                              0
##       n_non_stop_unique_tokens                      num_hrefs
##                              0                              0
##                 num_self_hrefs                       num_imgs
##                              0                              0
##                     num_videos           average_token_length
##                              0                              0
##                   num_keywords      data_channel_is_lifestyle
##                              0                              0
##  data_channel_is_entertainment            data_channel_is_bus
##                              0                              0
##         data_channel_is_socmed           data_channel_is_tech
##                              0                              0
##          data_channel_is_world                     kw_min_min
##                              0                              0
##                     kw_max_min                     kw_avg_min
##                              0                              0
##                     kw_min_max                     kw_max_max
##                              0                              0
##                     kw_avg_max                     kw_min_avg
##                              0                              0
##                     kw_max_avg                     kw_avg_avg
##                              0                              0
##      self_reference_min_shares      self_reference_max_shares
##                              0                              0
##     self_reference_avg_sharess              weekday_is_monday
##                              0                              0
##             weekday_is_tuesday           weekday_is_wednesday
##                              0                              0
##            weekday_is_thursday              weekday_is_friday
##                              0                              0
##            weekday_is_saturday              weekday_is_sunday
##                              0                              0
##                     is_weekend                         LDA_00
##                              0                              0
##                         LDA_01                         LDA_02
##                              0                              0
##                         LDA_03                         LDA_04
##                              0                              0
```

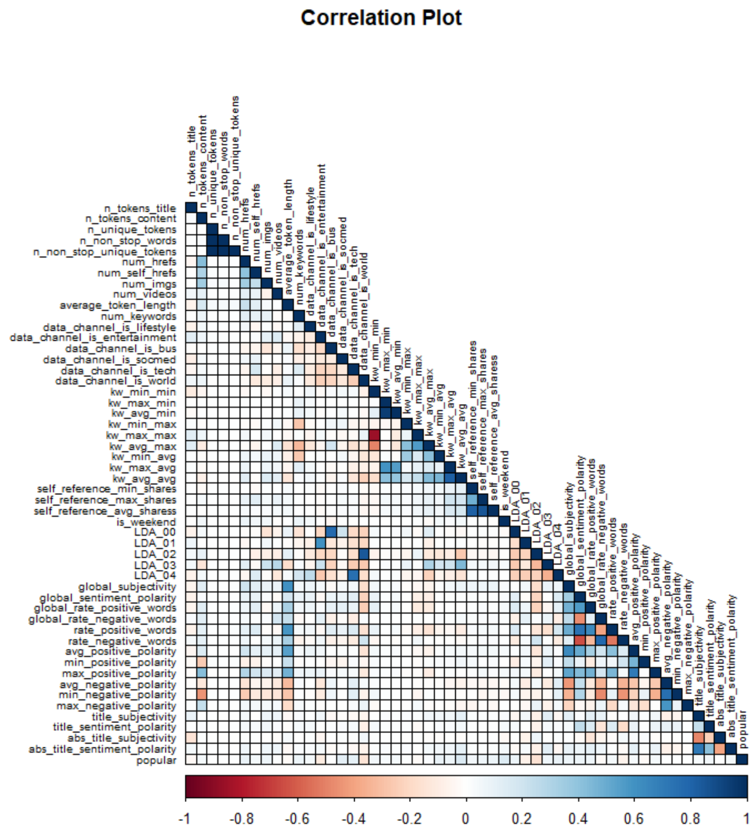
```
##          global_subjectivity     global_sentiment_polarity 
##                            0                             0 
##   global_rate_positive_words    global_rate_negative_words 
##                            0                             0 
##          rate_positive_words           rate_negative_words 
##                            0                             0 
##        avg_positive_polarity         min_positive_polarity 
##                            0                             0 
##        max_positive_polarity         avg_negative_polarity 
##                            0                             0 
##        min_negative_polarity         max_negative_polarity 
##                            0                             0 
##           title_subjectivity      title_sentiment_polarity 
##                            0                             0 
##       abs_title_subjectivity  abs_title_sentiment_polarity 
##                            0                             0 
##                      popular 
##                            0

#duplicates check
articles1[duplicated(articles1)]

## data frame with 0 columns and 39644 rows
```

4.2 Dropping redundant features

```
#removing all day leaving only is_weekend to avoid repetition
articles2 <- subset( articles1, select = -c(weekday_is_monday,
weekday_is_tuesday, weekday_is_wednesday,
                                            weekday_is_thursday,
weekday_is_friday, weekday_is_saturday,
                                            weekday_is_sunday) )

#removing other non_informative features
articles3 <- subset( articles2, select = -c(url, timedelta ) )
```

4.3 Checking for and removing highly correlated features

```
Cor <- round(cor(articles3),2)

corrplot(Cor, type="lower",method ="color", title = "Correlation Plot",
         mar=c(0,1,1,1), tl.cex= 0.65, outline= T, tl.col= rgb(0, 0, 0))
```




```
#Setting correlation cutoff
highlyCorrelated <- findCorrelation(Cor, cutoff = 0.7)
highlyCorCol <- colnames(articles2)[highlyCorrelated]
highlyCorCol

##  [1] "global_rate_positive_words" "kw_min_avg"
##  [3] "data_channel_is_socmed"     "max_positive_polarity"
##  [5] "global_rate_negative_words" "LDA_02"
##  [7] "self_reference_avg_sharess" "title_sentiment_polarity"
##  [9] "data_channel_is_tech"       "kw_avg_avg"
## [11] "data_channel_is_world"      "kw_max_avg"
## [13] "n_tokens_title"             "n_tokens_content"

#removing multicollinear variables
articles3 <- articles3[, -which(colnames(articles2) %in% highlyCorCol)]
dim(articles3)

## [1] 39644    38
```

5 Modelling

3 Models will be used to evaluate this classification task. First the dataset will be split into Training and Test set

```r
#To achieve reproducible model; set the random seed number
set.seed(100)

# Data is split into training and test set in a 80:20 ratio
TrainingIndex <- createDataPartition(articles3$popular, p=0.75, list = FALSE)

TrainingSet <- articles3[TrainingIndex,]# Training Set
TestingSet <- articles3[-TrainingIndex,]# Test Set
```

5.1 Logistic Regression

```r
model1 <- glm(popular~.,family=binomial(link='logit'),data = TrainingSet,
maxit = 1000 )
summary(model1)

##
## Call:
## glm(formula = popular ~ ., family = binomial(link = "logit"),
##     data = TrainingSet, maxit = 1000)
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    9.441e-02  1.416e-01   0.667 0.504846
## n_tokens_title                -3.682e-03  6.043e-03  -0.609 0.542319
## n_tokens_content               1.615e-04  3.800e-05   4.251 2.13e-05 ***
## n_non_stop_unique_tokens       5.274e-03  8.359e-03   0.631 0.528135
## num_hrefs                      1.293e-02  1.510e-03   8.560  < 2e-16 ***
## num_self_hrefs                -2.596e-02  3.797e-03  -6.837 8.07e-12 ***
## num_imgs                       6.194e-03  1.780e-03   3.479 0.000503 ***
## num_videos                     3.558e-03  3.357e-03   1.060 0.289194
## average_token_length          -1.398e-01  2.183e-02  -6.407 1.48e-10 ***
## num_keywords                   6.578e-02  7.818e-03   8.414  < 2e-16 ***
## data_channel_is_lifestyle     -1.263e-02  7.105e-02  -0.178 0.858949
## data_channel_is_entertainment -5.017e-01  4.655e-02 -10.777  < 2e-16 ***
## data_channel_is_bus           -1.784e-01  5.897e-02  -3.025 0.002482 **
## data_channel_is_socmed         1.013e+00  6.869e-02  14.741  < 2e-16 ***
## data_channel_is_tech           2.904e-01  5.612e-02   5.174 2.29e-07 ***
## kw_avg_min                    -3.526e-05  2.969e-05  -1.187 0.235070
## kw_min_max                    -1.149e-06  2.451e-07  -4.689 2.75e-06 ***
## kw_max_max                    -6.363e-07  8.125e-08  -7.831 4.84e-15 ***
## kw_avg_max                     5.268e-07  1.646e-07   3.200 0.001377 **
## kw_min_avg                     1.450e-04  1.272e-05  11.398  < 2e-16 ***
## kw_max_avg                     1.919e-05  3.615e-06   5.310 1.10e-07 ***
## self_reference_avg_sharess     7.165e-06  9.636e-07   7.436 1.04e-13 ***
## is_weekend                     8.783e-01  3.933e-02  22.332  < 2e-16 ***
## LDA_01                        -4.471e-01  9.266e-02  -4.825 1.40e-06 ***
## LDA_02                        -1.206e+00  8.228e-02 -14.660  < 2e-16 ***
## LDA_03                        -2.668e-01  8.340e-02  -3.200 0.001377 **
## global_subjectivity            1.220e+00  1.744e-01   6.996 2.64e-12 ***
## global_sentiment_polarity     -7.266e-02  3.429e-01  -0.212 0.832190
## global_rate_positive_words    -1.477e+00  1.263e+00  -1.169 0.242345
```

```
## global_rate_negative_words     2.327e-02  2.026e+00   0.011 0.990834
## avg_positive_polarity          2.177e-02  2.850e-01   0.076 0.939109
## min_positive_polarity         -9.397e-01  2.388e-01  -3.935 8.32e-05 ***
## max_positive_polarity         -5.861e-02  8.966e-02  -0.654 0.513273
## avg_negative_polarity          2.611e-01  1.786e-01   1.462 0.143692
## max_negative_polarity         -1.652e-01  1.917e-01  -0.861 0.388985
## title_subjectivity             1.163e-01  4.526e-02   2.569 0.010212 *
## title_sentiment_polarity       2.063e-01  5.016e-02   4.113 3.91e-05 ***
## abs_title_subjectivity         1.951e-01  7.761e-02   2.514 0.011950 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 41090  on 29732  degrees of freedom
## Residual deviance: 37792  on 29695  degrees of freedom
## AIC: 37868
##
## Number of Fisher Scoring iterations: 5
```

```
#calculating errors
#test error
cut <- 0.5

yhat = (predict(model1,TrainingSet,type="response")>cut)
tr.err = mean(TrainingSet$popular != yhat)
tr.err
```

```
## [1] 0.3586251
```

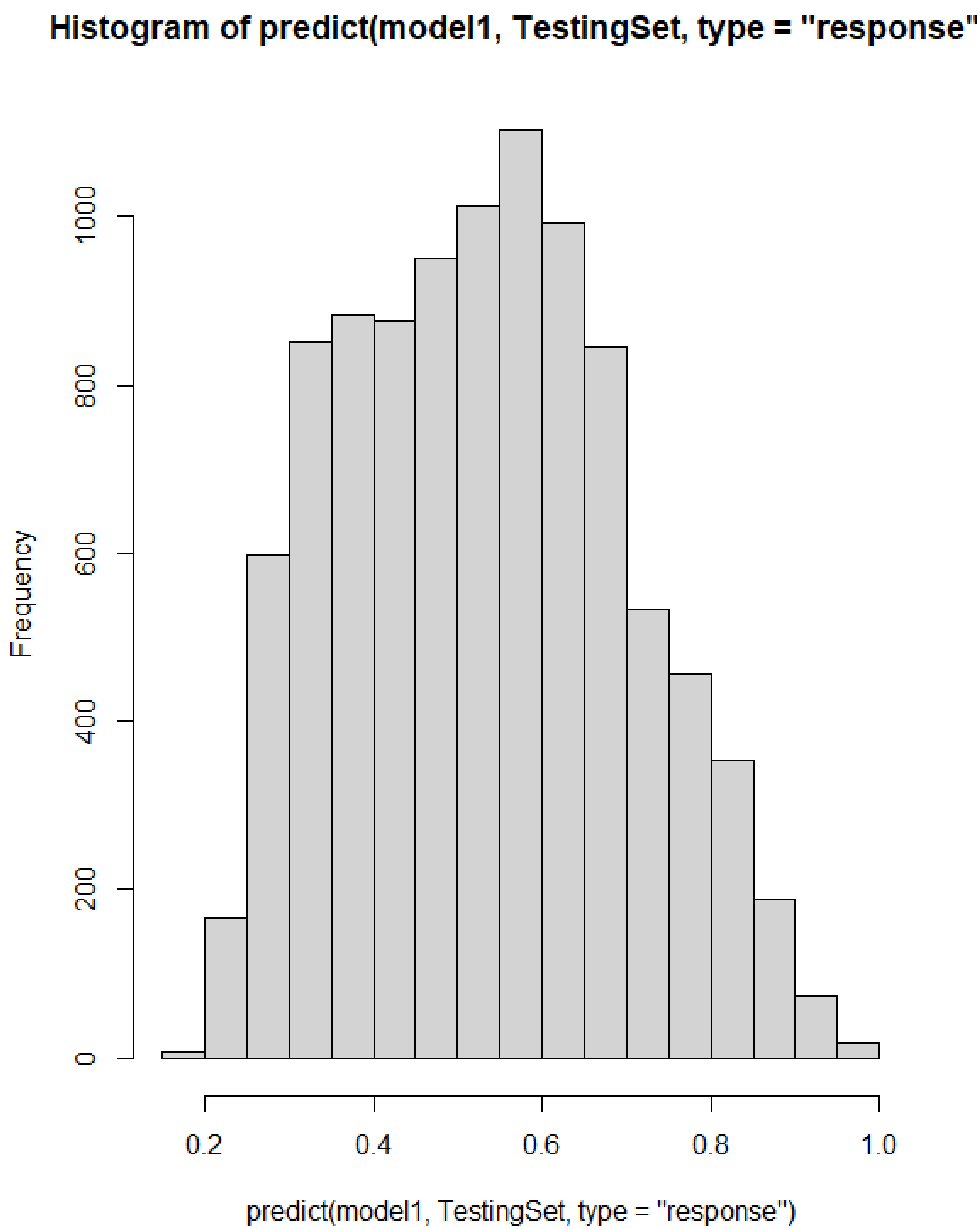
```
# calculate the testing error in a similar manner to the training error
yhat = (predict(model1,TestingSet,type="response")>cut)
te.err = mean(TestingSet$popular != yhat)
print(te.err)
```

```
## [1] 0.3507214
```

```
#print(predict(cls,test,type="response")>cut)

# calculation of Naive predictor error rate where cut = 1

# so the Naive predictor will simply predict all customers stay...
# so it will make errors on each customer that leaves the bank

trN.err <- mean(!TrainingSet$popular)
teN.err <- mean(!TestingSet$popular)

print(paste("Naive train error",trN.err))
```

```
## [1] "Naive train error 0.467157703561699"
```

```r
print(paste("Naive test error",teN.err))

## [1] "Naive test error 0.4641307637978"

hist(predict(model1,TestingSet,type="response"))
```



```r
# Prediction on TestingSet using Logistic Regression
prediction <- predict(model1, TestingSet, type ="response")
head(prediction)

##         3         8         9        12        17        18
## 0.5665468 0.6467499 0.5943744 0.5725975 0.5169225 0.5029694

#Assigning probabilities - If prediction exceeds threshold of 0.5, 1 else 0
prediction <- ifelse(prediction >0.5,1,0)
head(prediction)

##  3  8  9 12 17 18
##  1  1  1  1  1  1

#Computing confusion matrix values
confusionMatrix(factor(TestingSet$popular),factor(prediction), mode
='everything', positive ="0")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2729 1871
##          1 1605 3706
##
##                Accuracy : 0.6493
##                  95% CI : (0.6398, 0.6587)
##     No Information Rate : 0.5627
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2922
##
##  Mcnemar's Test P-Value : 6.965e-06
##
##             Sensitivity : 0.6297
##             Specificity : 0.6645
##          Pos Pred Value : 0.5933
##          Neg Pred Value : 0.6978
##               Precision : 0.5933
##                  Recall : 0.6297
##                      F1 : 0.6109
##              Prevalence : 0.4373
##          Detection Rate : 0.2754
##    Detection Prevalence : 0.4641
##       Balanced Accuracy : 0.6471
##
##        'Positive' Class : 0
##
```

5.2 Decision Trees

3 cp partitions 0.01, 0.001 and 0.00001 will be used for prediction.

5.2.1 Tree 1 cp = 0.01

```
tree1 <- rpart(popular~., method = 'class', data = TrainingSet, control =
rpart.control(cp = 0.01))

#using tree1 for predicting test set
test_prediction1 <-predict(tree1, TestingSet, type = 'class')
head(test_prediction1)

##  3  8  9 12 17 18
##  0  1  1  0  0  0
## Levels: 0 1

# converting TestingSet$popular to factor
popular_factor <-  as.factor(TestingSet$popular)


#confusion matrix for tree1
cfm1 <- confusionMatrix(test_prediction1, popular_factor)
cfm1

## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0    1
##          0 2285 1289
##          1 2315 4022
##
##                Accuracy : 0.6364
##                  95% CI : (0.6268, 0.6458)
##     No Information Rate : 0.5359
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2579
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4967
##             Specificity : 0.7573
##          Pos Pred Value : 0.6393
##          Neg Pred Value : 0.6347
##              Prevalence : 0.4641
##          Detection Rate : 0.2306
##    Detection Prevalence : 0.3606
##       Balanced Accuracy : 0.6270
##
##        'Positive' Class : 0
##
```

```r
#error rate for tree1
error_rate1 <- 1 - cfm1$overall["Accuracy"]
error_rate1

##  Accuracy
## 0.3636364
```

5.2.2 Tree 2 cp = 0.001

```r
tree2 <- rpart(popular~., method = 'class', data = TrainingSet, control =
rpart.control(cp = 0.001))

#using tree2 for predicting test set
test_prediction2 <-predict(tree2, TestingSet, type = 'class')
head(test_prediction2)

##  3  8  9 12 17 18
##  0  1  1  0  1  0
## Levels: 0 1

#confusion matrix for tree1
cfm2 <- confusionMatrix(test_prediction2, popular_factor)
cfm2

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2722 1579
##          1 1878 3732
##
##                Accuracy : 0.6512
##                  95% CI : (0.6417, 0.6606)
##     No Information Rate : 0.5359
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2957
##
##  Mcnemar's Test P-Value : 4.013e-07
##
##             Sensitivity : 0.5917
##             Specificity : 0.7027
##          Pos Pred Value : 0.6329
##          Neg Pred Value : 0.6652
##              Prevalence : 0.4641
##          Detection Rate : 0.2746
##    Detection Prevalence : 0.4340
##       Balanced Accuracy : 0.6472
##
##        'Positive' Class : 0
##
```

```r
#error rate for tree1
error_rate1 <- 1 - cfm2$overall["Accuracy"]
error_rate1

##  Accuracy
## 0.3488044
```

5.2.3 Tree 3, cp = 0.00001

```r
tree3 <- rpart(popular~., method = 'class', data = TrainingSet, control =
rpart.control(cp = 0.00001))

#using tree1 for predicting test set
test_prediction3 <-predict(tree3, TestingSet, type = 'class')
head(test_prediction3)

##  3  8  9 12 17 18
##  0  1  0  1  0  0
## Levels: 0 1

#confusion matrix for tree1
cfm3 <- confusionMatrix(test_prediction3, popular_factor)
cfm3

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2559 2062
##          1 2041 3249
##
##                Accuracy : 0.586
##                  95% CI : (0.5762, 0.5957)
##     No Information Rate : 0.5359
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.168
##
##  Mcnemar's Test P-Value : 0.7549
##
##             Sensitivity : 0.5563
##             Specificity : 0.6117
##          Pos Pred Value : 0.5538
##          Neg Pred Value : 0.6142
##              Prevalence : 0.4641
##          Detection Rate : 0.2582
##    Detection Prevalence : 0.4662
##       Balanced Accuracy : 0.5840
##
##        'Positive' Class : 0
##
```

```r
#error rate for tree1
error_rate3 <- 1 - cfm3$overall["Accuracy"]
error_rate3

##  Accuracy
## 0.4139845
```

5.3 Model 3 - Random Forest

```r
#First step in running rf is converting target variable to factor
TrainingSet$popular <- as.factor(TrainingSet$popular)
```

5.3.1 - Random Forest ntree = 100

```r
# Assuming your data frame is called 'df' and the target variable is 'target'
rf_model <- randomForest(popular~ ., data = TrainingSet, ntree = 100)
rf_model

##
## Call:
##  randomForest(formula = popular ~ ., data = TrainingSet, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 34.47%
## Confusion matrix:
##      0     1 class.error
## 0 8246  5644   0.4063355
## 1 4606 11237   0.2907278

rf_predictions <- predict(rf_model, TestingSet)
head(rf_predictions)

##  3  8  9 12 17 18
##  0  0  1  0  0  0
## Levels: 0 1

#confusion matrix for rf_model
cf_rf <- confusionMatrix(rf_predictions, popular_factor)
cf_rf

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2731 1486
##          1 1869 3825
##
##                Accuracy : 0.6615
##                  95% CI : (0.6521, 0.6708)
##     No Information Rate : 0.5359
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.3157
##
##  Mcnemar's Test P-Value : 4.252e-11
##
##             Sensitivity : 0.5937
##             Specificity : 0.7202
##          Pos Pred Value : 0.6476
##          Neg Pred Value : 0.6718
##              Prevalence : 0.4641
##          Detection Rate : 0.2756
##    Detection Prevalence : 0.4255
##       Balanced Accuracy : 0.6569
##
##        'Positive' Class : 0
##
```

```
#error rate for tree1
rf_error_rate <- 1 - cf_rf$overall["Accuracy"]
rf_error_rate
```

```
##  Accuracy
## 0.3385128
```

5.3.2 Random Forest ntree = 500

```
rf_model2 <- randomForest(popular~ ., data = TrainingSet, ntree = 500,
importance = TRUE)
rf_model2
```

```
##
## Call:
##  randomForest(formula = popular ~ ., data = TrainingSet, ntree = 500,
importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 33.54%
## Confusion matrix:
##      0     1 class.error
## 0 8254  5636   0.4057595
## 1 4336 11507   0.2736855
```

```
rf2_predictions <- predict(rf_model2, TestingSet)
head(rf2_predictions)
```

```
##  3  8  9 12 17 18
##  0  0  0  0  0  0
## Levels: 0 1
```

```r
#confusion matrix for rf_model2
cf_rf2 <- confusionMatrix(rf2_predictions, popular_factor)
cf_rf2

## Confusion Matrix and Statistics
##
##            Reference
## Prediction    0    1
##           0 2761 1455
##           1 1839 3856
##
##                Accuracy : 0.6676
##                  95% CI : (0.6583, 0.6769)
##     No Information Rate : 0.5359
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3281
##
##  Mcnemar's Test P-Value : 2.502e-11
##
##             Sensitivity : 0.6002
##             Specificity : 0.7260
##          Pos Pred Value : 0.6549
##          Neg Pred Value : 0.6771
##              Prevalence : 0.4641
##          Detection Rate : 0.2786
##    Detection Prevalence : 0.4254
##       Balanced Accuracy : 0.6631
##
##        'Positive' Class : 0
##

#error rate for rf2
rf2_error_rate <- 1 - cf_rf2$overall["Accuracy"]
rf2_error_rate

## Accuracy
## 0.332358
```

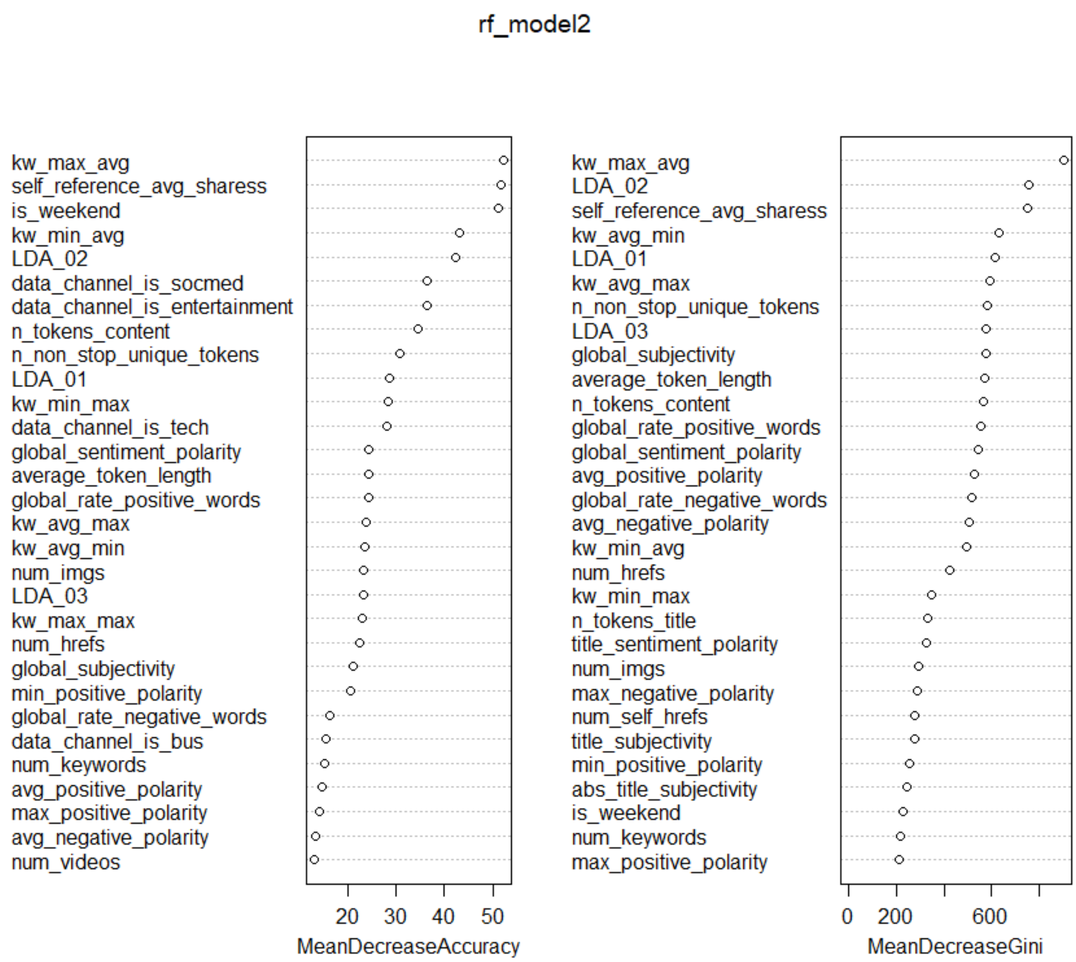
5.4 Calculating Feature Importance using Random Forest

```r
importance_values <- importance(rf_model2)
importance_values

##                                    0           1 MeanDecreaseAccuracy
## n_tokens_title              4.363174  3.33854834             5.529538
## n_tokens_content           20.794649 18.34319220            34.415810
## n_non_stop_unique_tokens   20.778322 13.74512270            30.688867
## num_hrefs                  11.719759 15.51879073            22.494046
## num_self_hrefs             10.780497  4.40122692            12.167521
## num_imgs                   26.219943  2.01084468            23.400708
## num_videos                 17.506425 -1.19419626            12.954985
```

```
## average_token_length          28.241977  -0.95029873                24.282404
## num_keywords                  11.891724   5.83443767                15.098458
## data_channel_is_lifestyle     13.092010 -10.31218676                 4.701038
## data_channel_is_entertainment 33.641036  16.09649599                36.376499
## data_channel_is_bus           15.998257  -1.67418147                15.362631
## data_channel_is_socmed        17.842549  35.25916542                36.405930
## data_channel_is_tech          21.259648  16.63668362                28.211243
## kw_avg_min                    17.212975  13.35841256                23.555205
## kw_min_max                     3.061054  22.65037524                28.356639
## kw_max_max                    11.697241  16.47082765                22.876947
## kw_avg_max                    23.317106   2.21467488                23.915655
## kw_min_avg                    27.633594  19.41837586                43.094963
## kw_max_avg                    37.565745  34.61000478                52.372030
## self_reference_avg_sharess    34.550917  33.74759922                51.720217
## is_weekend                    52.116303  24.22507614                51.283820
## LDA_01                        18.900924  15.88545286                28.738013
## LDA_02                        36.369859   9.95840729                42.441231
## LDA_03                        21.964375   5.08035386                23.209032
## global_subjectivity           12.352345  14.63540257                21.231503
## global_sentiment_polarity     19.630304   9.49759093                24.411006
## global_rate_positive_words    17.801403  11.77461801                24.273712
## global_rate_negative_words    12.073887   7.14262865                16.323985
## avg_positive_polarity          4.585040  13.41228729                14.774362
## min_positive_polarity          5.047227  17.74365088                20.518451
## max_positive_polarity          6.508848  10.54839435                14.078436
## avg_negative_polarity          9.085544   7.46072631                13.266155
## max_negative_polarity          8.573425   5.53425953                11.396534
## title_subjectivity            10.017977   3.68968268                10.705089
## title_sentiment_polarity      11.321232   0.95380545                 8.744615
## abs_title_subjectivity         8.840689  -0.07520549                 6.539739
##                               MeanDecreaseGini
## n_tokens_title                       332.22283
## n_tokens_content                     564.58310
## n_non_stop_unique_tokens             583.52040
## num_hrefs                            422.74092
## num_self_hrefs                       279.25186
## num_imgs                             292.07579
## num_videos                           166.74953
## average_token_length                 570.51392
## num_keywords                         217.50046
## data_channel_is_lifestyle             33.67307
## data_channel_is_entertainment        165.38118
## data_channel_is_bus                   54.26601
## data_channel_is_socmed               102.69582
## data_channel_is_tech                 118.90734
## kw_avg_min                           634.55194
## kw_min_max                           347.37658
## kw_max_max                           147.74721
## kw_avg_max                           595.58835
## kw_min_avg                           496.82662
```

```
## kw_max_avg                          903.40462
## self_reference_avg_sharess          752.87178
## is_weekend                          228.62720
## LDA_01                              617.16050
## LDA_02                              755.81093
## LDA_03                              577.19293
## global_subjectivity                 574.89423
## global_sentiment_polarity           546.93224
## global_rate_positive_words          554.40838
## global_rate_negative_words          514.56847
## avg_positive_polarity               526.78724
## min_positive_polarity               256.15986
## max_positive_polarity               211.59787
## avg_negative_polarity               507.31990
## max_negative_polarity               287.07209
## title_subjectivity                  276.33602
## title_sentiment_polarity            325.06909
## abs_title_subjectivity              244.63308
```

```r
varImpPlot(rf_model2)
```



6 Model Evaluation

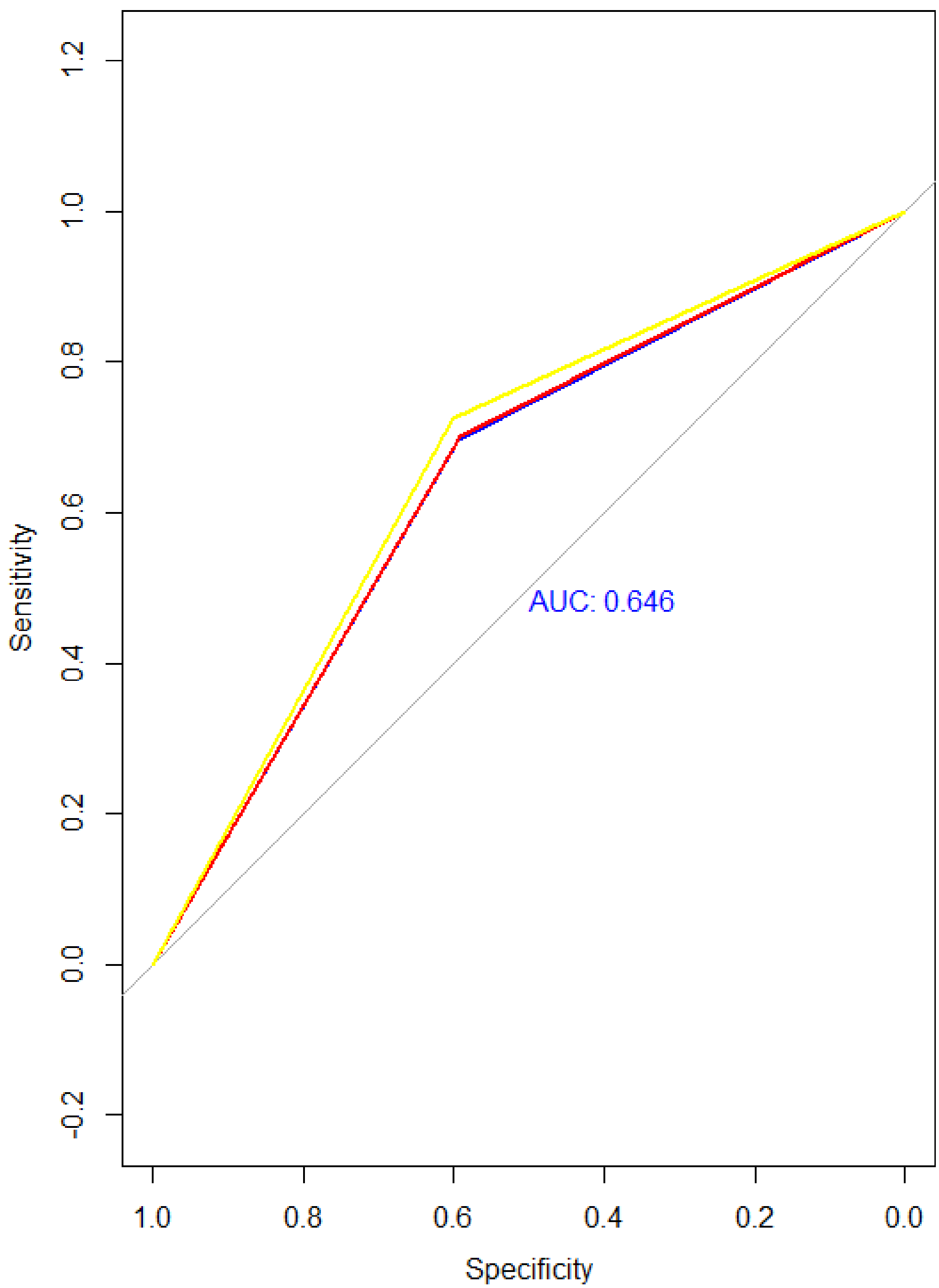
Models are evaluated using accuracy from confusion matrix, testing error and also AUC score.

6.1 Table of Results

```
# Create a new table with some sample data
Model_Comparison <- data.frame(
  Model = c("Logistic Regression", "Decison Trees", "Random Forest"),
  Accuracy = c(0.644, 0.645, 0.668),
  TestingError = c(0.356, 0.355, 0.332),
  Sensitivity = c(0.625, 0.585, 0.595),
  Specificity = c(0.659, 0.702, 0.732))

# Display the new table
print(Model_Comparison)

##                 Model Accuracy TestingError Sensitivity Specificity
## 1 Logistic Regression    0.644        0.356       0.625       0.659
## 2       Decison Trees    0.645        0.355       0.585       0.702
## 3       Random Forest    0.668        0.332       0.595       0.732
```

###6.2 Plotting ROC Curves

```
#converting prediction scores data type before plotting curves
test_prediction2 <- as.numeric(test_prediction2)
rf2_predictions <- as.numeric(rf2_predictions)

#creating the ROC function
glm_roc_curve <- roc(TestingSet$popular, prediction)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

tree_roc_curve <- roc(TestingSet$popular, test_prediction2)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

rf_roc_curve <- roc(TestingSet$popular, rf2_predictions)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

# Plotting ROC Curves
plot(glm_roc_curve, col = "blue", print.auc = TRUE)
plot(tree_roc_curve, col = "red", add = TRUE)
plot(rf_roc_curve, col = "yellow", add = TRUE)
```




```
#calculating AUC curves of models# Calculate ROC and AUC using pROC
glm_score <- print(paste('glm roc_roc_curve score is',auc(glm_roc_curve)))

## [1] "glm roc_roc_curve score is 0.645528947303791"

tree_score <- print(paste('tree_roc_curve score is',auc(tree_roc_curve)))
```

```
## [1] "tree_roc_curve score is 0.647215827691502"

rf2_score <- print(paste('rf_roc_curve score is', auc(rf_roc_curve)))

## [1] "rf_roc_curve score is 0.663128842517171"
```

Proceeding with Random Forest because it has the highest accuracy 66.8% and AUC score of 0.663