Tweeter sentiment analysis during COVID19 pandemic

Ivan Kozyryev

4/30/2020

Overview

The goal of my project was to determine whether the overall sentiment of the tweets with COVID19-related hashtags influenced whether the tweet would be retweeted and how many times. My initial hypothesis was that the more negative the tweet is the more retweets it will get because it has potential to stir emotions in many people. During the course of this project as I started to learn more about tweeter, I became more broadly interested in what parameters of the tweet effect the number of retweets. Specifically, I considered the following tweet variables:

1. number of followers
2. number of friends
3. number of hashtags
4. number of user mentions
5. sentiment of the text

While there 90 different variable associated with each tweet and many possibilities for further manipulation of the features, I chose to focus on the five above for two reasons:

- Based on my intuition about social media it seemed likely that those parameters might matter.
- I wanted to have a simpler model that can have interpretable implications about how one can spread positive sentiments and correct information on social media to a wide audience. Or alternatively, understand how *misinformation* related to the disease is effectively spread on social media.

Exploratory initial analysis

First let's load all the libraries needed for the analysis of Twitter data. I used rtweet library to interface with the tweeter API.

```r
source('Load_libraries.R')
```

As a first step, let's quickly check what topics are trending on Twitter now. I first get the current trends in the US, then sort all the topics by the tweet volume and look at the top 10 trending topics.

```r
US_trends <- get_trends("United States")

US_trends_sorted <- US_trends %>%
  group_by(trend) %>%
  summarize(tweet_vol = mean(tweet_volume)) %>%
  arrange(desc(tweet_vol))

US_trends_sorted$trend[1:10]
```

```
##  [1] "#WayV_BeyondLIVE"
##  [2] "Bad Bunny"
##  [3] "#HappyToHaveGOT7"
##  [4] "\"Bye\""
##  [5] "#<U+B0B4><U+C0B6><U+C774><U+AF43><U+C774><U+B418><U+B294><U+C21C><U+AC04>_<U+C601><U+C7AC><U+C
##  [6] "Drake"
##  [7] "taeil"
##  [8] "Murder Hornets"
##  [9] "#CalderonSiSabia"
## [10] "winwin"
```

Many of the days recently, covid-19 related hashtags were among the top 10 trending topics on Twitter which indicates its importance in the social discussion.

Obtaining the data for tweets

Let's read *original* tweets and see what associated data is available for them. Thus, I have excluded all replies, retweets and quotes. I also only focused on the tweets in English since I am interested in analyzing the sentiment of their text.

```
covid19_st <- search_tweets("#COVID19 -filter:retweets -filter:quote -filter:replies",n = 36000, includ
```

```
## retry on rate limit...
## waiting about 13 minutes...
```

```
colnames(covid19_st)[1:25]
```

```
##  [1] "user_id"              "status_id"            "created_at"
##  [4] "screen_name"          "text"                 "source"
##  [7] "display_text_width"   "reply_to_status_id"   "reply_to_user_id"
## [10] "reply_to_screen_name" "is_quote"             "is_retweet"
## [13] "favorite_count"       "retweet_count"        "quote_count"
## [16] "reply_count"          "hashtags"             "symbols"
## [19] "urls_url"             "urls_t.co"            "urls_expanded_url"
## [22] "media_url"            "media_t.co"           "media_expanded_url"
## [25] "media_type"
```

As we can see, there is a lot of parameters associated with each tweet but for now I am interested in only a few of those. There are 90 different parameters but I only displayed 25 for example above. Number of display columns can be modified to see all the available data types for each tweet.

```
covid_twts <- covid19_st[,c("status_id","screen_name","text","retweet_count","followers_count","friends
```

Pre-processing the text for tweets

In order to perform analysis of the text, I first need to clean up the body of the tweet:

- Separate the text from other tweet parameters
- Remove URLs
- Only keep the upper and lower case characters

```
twt_txt <- covid_twts$text # get the tweet text
head(twt_txt)
```

```
## [1] "Checking in on everyone! How are you all doing?\n\nMy sister's husband died from #COVID19 a few
## [2] "#COVID19: Billionaire Prophet, Jeremiah Fufeyin Receives Worldwide Accolades For Effective Propl
## [3] "#COVID19: Check Out The New Sitting Arrangement Of BRT Buses In Lagos (Photos/Video) https://t.
## [4] "Getting mixed messages about whether the Government will enable the emergency changes to the #M
## [5] "Show starts NOW!! <U+0001F525>\nThe @PUBGMOBILE @gwbps day 2 for EU/MENA kicks of in 5 minutes!
## [6] "Spain's kids got freedom after 44 days of lockdown #Spain #Lockdown #Coronavirus #COVID #Covid_
```

```r
twt_txt_no_url <- rm_twitter_url(twt_txt)
# head(twt_txt_no_url)
twt_chrs <- gsub("[^A-Za-z]"," ", twt_txt_no_url)
head(twt_chrs)
```

```
## [1] "Checking in on everyone  How are you all doing  My sister s husband died from  COVID   a few day
## [2] " COVID    Billionaire Prophet  Jeremiah Fufeyin Receives Worldwide Accolades For Effective Propl
## [3] " COVID    Check Out The New Sitting Arrangement Of BRT Buses In Lagos  Photos Video "
## [4] "Getting mixed messages about whether the Government will enable the emergency changes to the  Mo
## [5] "Show starts NOW      The  PUBGMOBILE  gwbps day   for EU MENA kicks of in   minutes  And at the
## [6] "Spain s kids got freedom after    days of lockdown  Spain  Lockdown  Coronavirus  COVID  Covid
```

As we can see by comparing the same tweets in the "raw" form and after formatting, we are left only with letters that can be further processed and analyzed.

Pre-processing other tweet-related data

The data we read in provides the lists of the hashtags and users mentioned in each tweet, however, I was interested in looking into how the number of hashtags and mentions effects the degree of retweeting which required some initial processing.

First, lets count the number of mentions and hashtags

```r
twts_tidy <- covid_twts %>%
  mutate(text = twt_chrs) %>%
  mutate(mentions = lengths(mentions_user_id)) %>%
  mutate(hashtags = lengths(hashtags)) %>%
  mutate(urls = lengths(urls_url))
```

It is important to notice that most of the tweets don't have any mentions with "NA" in place so we need to properly account for that. I first found which tweets don't have any associated mentions of users and then properly corrected the "mentions" count for those entries.

```r
mentions_NAinds <- which(is.na(covid_twts$mentions_user_id)) # find which mentions are non-existent
twts_tidy$mentions[mentions_NAinds] <- 0 # correct the number of mentions entries for those indeces
```

Unlike for mentions, note that each tweet has at least one hashtag since we specifically searched for #COVID19 tweets to begin with. But we also need to account for the fact that not all tweets have mentioned urls:

```r
urls_NAinds <- which(is.na(covid_twts$urls_url)) # find which urls are non-existent
twts_tidy$urls[urls_NAinds] <- 0 # correct the number of mentions entries for those indeces
```

Just for convinience, let's rename the columns of the tweet data set and remove some of the columns that are redundant or will not be useful for our analysis.

```r
twts_tidy <- twts_tidy %>%
  dplyr::rename(id = status_id, retweets = retweet_count, friends = friends_count, followers = followers

twts_tidy <- twts_tidy %>%
  dplyr::select(-c(screen_name,mentions_user_id,urls_url))

colnames(twts_tidy)
```

```
## [1] "id"        "text"      "retweets"  "followers" "friends"   "hashtags"
## [7] "mentions"  "urls"
```

Text analysis

In order to study the sentiment of individual tweets we need to unnest tokens (words in this case) and remove stop words that don't carry additional information.

```
twts_clean <- twts_tidy %>%
  unnest_tokens(word,text) %>%
  anti_join(get_stopwords(), by = "word")
```

Let's look at the most common words used in tweets associated with #COVID19

```
common_words <- twts_clean %>%
  count(word) %>%
  arrange(desc(n)) %>%
  top_n(n=25, wt = n)

common_words$word
```

```
##  [1] "covid"       "coronavirus" "s"           "cases"       "amp"
##  [6] "new"         "pandemic"    "can"         "people"      "lockdown"
## [11] "t"           "us"          "deaths"      "may"         "help"
## [16] "now"         "total"       "health"      "today"       "day"
## [21] "one"         "get"         "time"        "world"       "india"
```

While many of the top words listed have important meaning (like "us" or "people"), there additional words or letter that don't provide any useful information. In the next step, I remove unique stop words that don't add any meaning.

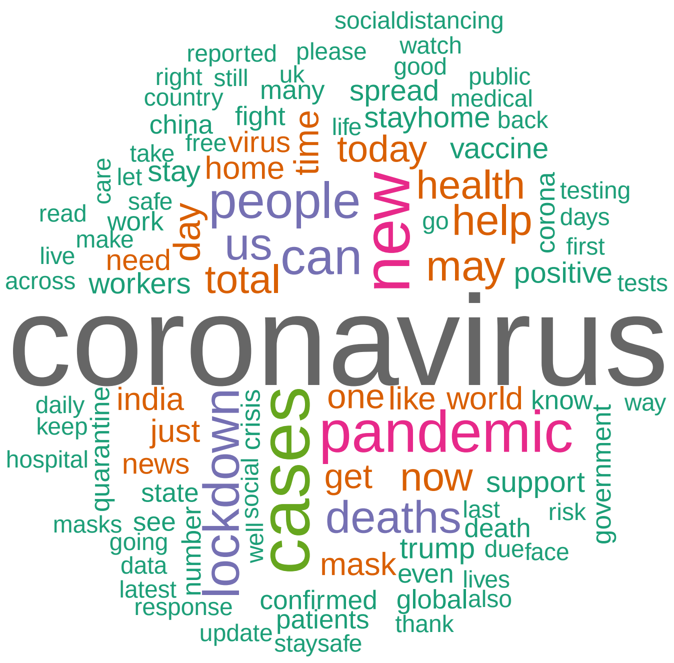
```
uniq_sw <- data.frame(word = c("s","covid","amp","t", "via", "m", "re", "don", "ve", "q", "gt", "o", "p
#
twts_clean <- twts_clean %>%
   anti_join(uniq_sw, by = "word")
```

Notice that I also removed "covid" since all the tweets have it because we specifically searched for #COVID19 and otherwise the wordcloud would be overwhelmed by the "covid" word (hashtag symbol and "19" were removed during the pre-processing step for the text).

Visualizing the most common words

Let's see which words are the most popular in tweets after we cleanup the text data.

```
pal <- brewer.pal(8,"Dark2")

twts_clean %>%
  count(word) %>%
  with(wordcloud(word,n,random.order = FALSE, max.words = 100, colors = pal))
```



Sentiment analysis of the #COVID19 tweets

From the wordcloud above, we see that some of the most mentioned words are positive in meaning while others are very negative in meaning. We are now ready to perform tweet sentiment analysis to see whether the *overall* tweet sentiment influences the number of retweets (and therefore potentially the influence of a tweet on other users). In the first step I used "afinn" lexicon which assigned a positive or negative score to each word. The reason I decided to use afinn lexicon is that it doesn't purely flag the word as "positive" or "negative" but assigns degrees to how positive or negative each word is..

Now lets assign a quantitative score to tweets to determine whether they are positive or negative. But first we need to determine which words from the afinn database appear in the tweets we are processing.

```r
twts_afinn <- twts_clean %>%
  inner_join(get_sentiments("afinn"), by = "word")
```

Lets assign the sentiment score to each unique tweet

```r
afinn_sentiment <- twts_afinn %>%
  group_by(id,retweets,followers,friends,hashtags,mentions,urls) %>%
  summarize(score = sum(value)) %>%
  arrange(desc(score))
```

Let's look at some of the most positive tweets to get inspiration for your day (and make sure that our analysis makes sense):

```r
covid_twts$text[which(twts_tidy$id == afinn_sentiment$id[1])]
```

```
## [1] "Dear God \nMake our day useful\nOur night restful\nOur home peaceful\nOur future bright \nOur d
```

```
covid_twts$text[which(twts_tidy$id == afinn_sentiment$id[2])]
```

```
## [1] "Hofmann's of Wakefield Award Winning Pie Box is fantastic value for money &amp; amazing quality
```

```
covid_twts$text[which(twts_tidy$id == afinn_sentiment$id[3])]
```

```
## [1] "Another great voice of confidence in our solution:\nHappy to share that following the winning o
```

Those are indeed positive tweets, so it makes sense that they received high ranking.

We can also look at the most negative tweets (according to the afinn lexicon). However, I am going to suppress the output here because it is usually very negative and sometimes not appropriate.

```
#covid_twts$text[which(twts_tidy$id == afinn_sentiment$id[length(afinn_sentiment$id)])]
#covid_twts$text[which(twts_tidy$id == afinn_sentiment$id[length(afinn_sentiment$id)-1])]
#covid_twts$text[which(twts_tidy$id == afinn_sentiment$id[length(afinn_sentiment$id)-2])]
```

Those tweets do have negative sentiments and the ranking according to the afinn lexicon makes intuitive sense.

Regression analysis for binary retweet status

As first step, lets see what factors determine whether the tweet is retweeted at all.

Logistic regression

Change the number of retweets into a binary indicator form: "N"(no retweets)/"A"(any retweet)

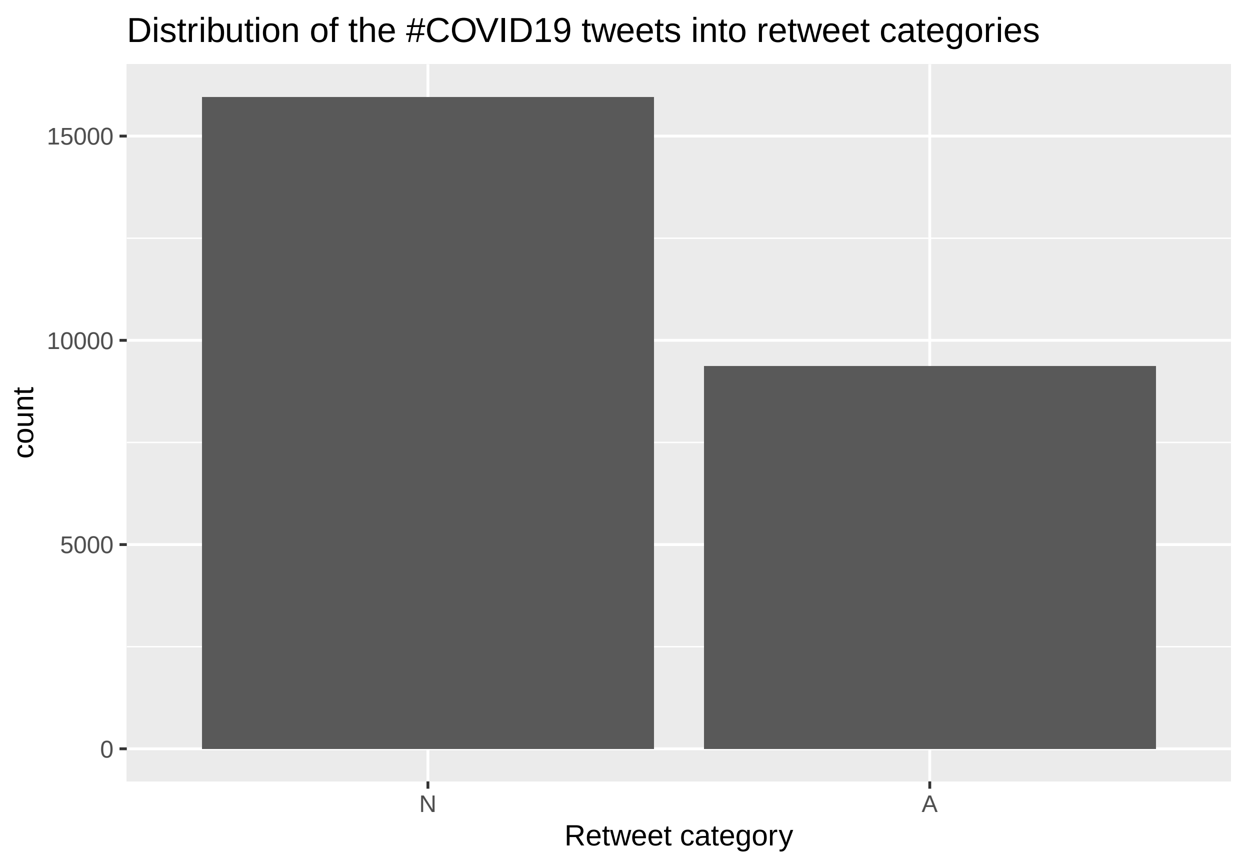
```
afinn_binary <- afinn_sentiment %>%
  mutate(retwtBi = ifelse(retweets > 0, "A", "N"))
```

Convert the retweet indicator into a ranked factor: N < A

```
afinn_binary <- afinn_binary %>%
  mutate(retwtBi = factor(retwtBi, levels = c("N", "A")))
```

Let's look at the distribution of the tweets into the "None" and "Any" retweet categories.

```
afinn_binary %>%
  ggplot(aes(retwtBi)) +
  geom_bar() +
  xlab("Retweet category") +
  ggtitle("Distribution of the #COVID19 tweets into retweet categories")
```



As expected, most of the tweets are not reptweeted at all, but a significant portion got retweeted at least ones. In fact we can see how many tweets got retweeted:

```
summaryBi <- afinn_binary %>%
  group_by(retwtBi) %>%
  summarize(n = n())
summaryBi
```

```
## # A tibble: 2 x 2
##   retwtBi     n
##   <fct>   <int>
## 1 N       15961
## 2 A        9364
```

For this given run, looks like `summaryBi$n[2]/sum(summaryBi$n)*100`% of `sum(summaryBi$n)` tweets I am analyzing have been retweeted.

A brief intro to logistic regression

The probability that a tweet is retweeted is $0 < p(A) < 1$.We can model such a probability distribution with a logistic function:

$$\sigma(t) = \frac{1}{1+e^{-t}}$$

which has the following important properties:

- $\sigma(t) \to 0$ as $t \to -\infty$
- $\sigma(t) \to 1$ as $t \to \infty$

- $\sigma(t) = 1/2$ for $t = 0$

Therefore, it seems reasonable to model the probability $p(A)$ as:

$$p = \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_k x_k)}{1 + \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_k x_k)}$$

which is equivalent to the expression for $\sigma(t)$ above. Therefore, the odds are

$$\frac{p}{1-p} = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_k x_k)$$

but more importantantly $logit(p)$ is

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_k x_k$$

is linear in the fitting coefficients β_k. Alternatively, equation above can be represented as:

$\log \frac{p(A)}{p(N)} = \beta_0 + \beta^T x$

where $p(A) + p(N) = 1$.

Implementing logistic regression

Before we can we train the logistic regression model, we need to split the dataset into train and test subsets; here I used the 60/40 split ratio.

```
trBi <- sample(nrow(afinn_binary),round(nrow(afinn_binary)*0.6))
trainSet <- afinn_binary[trBi,]
testSet <- afinn_binary[-trBi,]
```

First, we fit the logit model to the training set to see which parameters are statistically significant:

```
model1LR <- glm(retwtBi ~ followers + friends + score + mentions + hashtags + urls, data = trainSet, fa
summary(model1LR)
```

```
##
## Call:
## glm(formula = retwtBi ~ followers + friends + score + mentions +
##     hashtags + urls, family = "binomial", data = trainSet)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.8767  -0.8500   1.3708   1.6603
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.976e-01  3.313e-02 -24.071  < 2e-16 ***
## followers    1.051e-05  6.251e-07  16.816  < 2e-16 ***
## friends      1.194e-05  2.775e-06   4.302  1.7e-05 ***
## score       -3.423e-03  4.905e-03  -0.698    0.485
## mentions     1.253e-01  1.307e-02   9.588  < 2e-16 ***
## hashtags    -3.205e-03  5.424e-03  -0.591    0.555
## urls        -4.482e-02  3.305e-02  -1.356    0.175
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 20045  on 15194  degrees of freedom
## Residual deviance: 18592  on 15188  degrees of freedom
## AIC: 18606
##
## Number of Fisher Scoring iterations: 10
```

In order to determine how accurate the fitted model is we now apply it to the "test" dataset and compare to the actual values:

```
prob1LR <- predict(model1LR,testSet, type = "response")
# prob1LR
cl1LR <- ifelse(prob1LR > 0.5, "A", "N")
```

Confusion matrix will provide a lot of valuable information on the model peformance. I set the positive class to "A" as we are interested in predicting whether the tweet with receive any retweets

```
confusionMatrix(factor(cl1LR, levels = c("N","A")),testSet$retwtBi, positive = "A")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    A
##          N 6248 3061
##          A  158  663
##
##                Accuracy : 0.6822
##                  95% CI : (0.6731, 0.6913)
##     No Information Rate : 0.6324
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1833
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.17803
##             Specificity : 0.97534
##          Pos Pred Value : 0.80755
##          Neg Pred Value : 0.67118
##              Prevalence : 0.36762
##          Detection Rate : 0.06545
##    Detection Prevalence : 0.08105
##       Balanced Accuracy : 0.57668
##
##        'Positive' Class : A
##
```

From the summary table of logistic regression model (model1LR) we can see that followers and mentions variables have the highest statistical significance in determining whether the tweet will be retweeted at least once or not. Let's run the second model now only with those parameters

```
model2LR <- glm(retwtBi ~ followers + mentions, data = trainSet, family = "binomial")
summary(model2LR)
```

```
##
```

```
## Call:
## glm(formula = retwtBi ~ followers + mentions, family = "binomial",
##     data = trainSet)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.8720  -0.8569   1.3701   1.5367
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.140e-01  2.037e-02 -39.966   <2e-16 ***
## followers    1.133e-05  6.247e-07  18.145   <2e-16 ***
## mentions     1.245e-01  1.297e-02   9.595   <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: 20045  on 15194  degrees of freedom
## Residual deviance: 18610  on 15192  degrees of freedom
## AIC: 18616
##
## Number of Fisher Scoring iterations: 10
```

We can see now that all of the variables are statistically significant as expected.

```
prob2LR <- predict(model2LR,testSet, type = "response") # predicted probability
# pred2LR
cl2LR <- ifelse(prob2LR > 0.5, "A", "N") # predicted binary classification
confusionMatrix(factor(cl2LR, levels = c("N","A")),testSet$retwtBi, positive = "A")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    A
##          N 6269 3073
##          A  137  651
##
##                Accuracy : 0.6831
##                  95% CI : (0.674, 0.6922)
##     No Information Rate : 0.6324
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1838
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.17481
##             Specificity : 0.97861
##          Pos Pred Value : 0.82614
##          Neg Pred Value : 0.67106
##              Prevalence : 0.36762
##          Detection Rate : 0.06426
##    Detection Prevalence : 0.07779
##       Balanced Accuracy : 0.57671
```

```
##
##        'Positive' Class : A
##
```

We can see that accuracy has in fact slightly improved.

One of the advantages of the ligistic regression is that it's coefficients lend to intuitive interpretation based on the log(odds) discussion above.

```
exp(coef(model2LR))
```

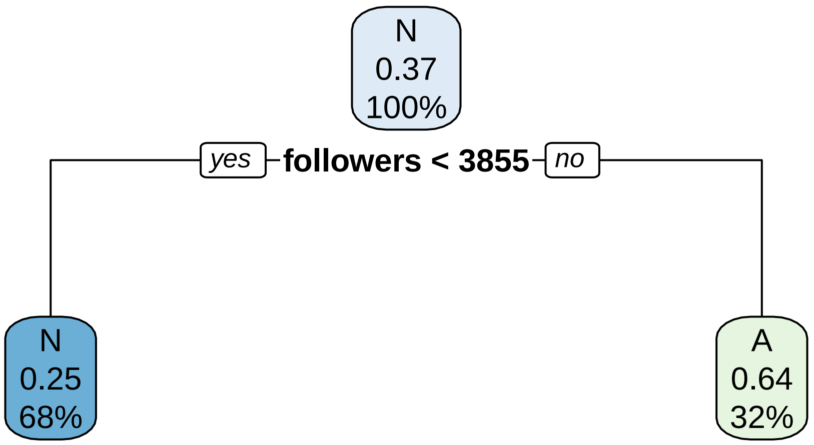
```
## (Intercept)   followers    mentions
##   0.4430992   1.0000113   1.1325668
```

We can see that mentions of other users have the largest effect on log-odds. This is something that the author of the tweet can control thus potentially increasing the odd of the tweet being retweeted

Decision tree for binary retweet indicator

The logit function assumed a linear relationship between the log(odd) and fitting coefficients. Thus it could be instructive to look at some nonlinear models. Let's use the same variables but fit the decision tree instead of logit regression

```
model1DT <- rpart(retwtBi ~ followers + friends + score + mentions + hashtags, data = trainSet, method =
rpart.plot(model1DT)
```



```
cl1DT <- predict(model1DT,testSet, type = "class")
confusionMatrix(cl1DT,testSet$retwtBi, positive = "A")
```

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

```
##
##           Reference
## Prediction    N    A
##          N 5281 1692
##          A 1125 2032
##
##                Accuracy : 0.7219
##                  95% CI : (0.7131, 0.7306)
##     No Information Rate : 0.6324
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3822
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5456
##             Specificity : 0.8244
##          Pos Pred Value : 0.6436
##          Neg Pred Value : 0.7573
##              Prevalence : 0.3676
##          Detection Rate : 0.2006
##    Detection Prevalence : 0.3116
##       Balanced Accuracy : 0.6850
##
##        'Positive' Class : A
##
```

As we can see from the confusion matrix, a single decision tree performs better than the logistic regression on such important parameters as accuracy and sensitivity (recall). Therefore, there is a lot for potential for random forest approach to improve the predicting power of our model.

Let's fit the random forest model to the full data:

```
model1RF <- randomForest(retwtBi ~ followers + friends + score + mentions + hashtags + urls, data = trai
```

We can now calculate predicted probabilities and classes:

```
prob1RFboth <- predict(model1RF,testSet, type = "prob") # predicted probabilities
prob1RF = prob1RFboth[,"A"]
# prob1RF
cl1RF <- predict(model1RF,testSet, type = "class") # predicted binary classification
# cl1RF
confusionMatrix(cl1RF,testSet$retwtBi, positive = "A")
```

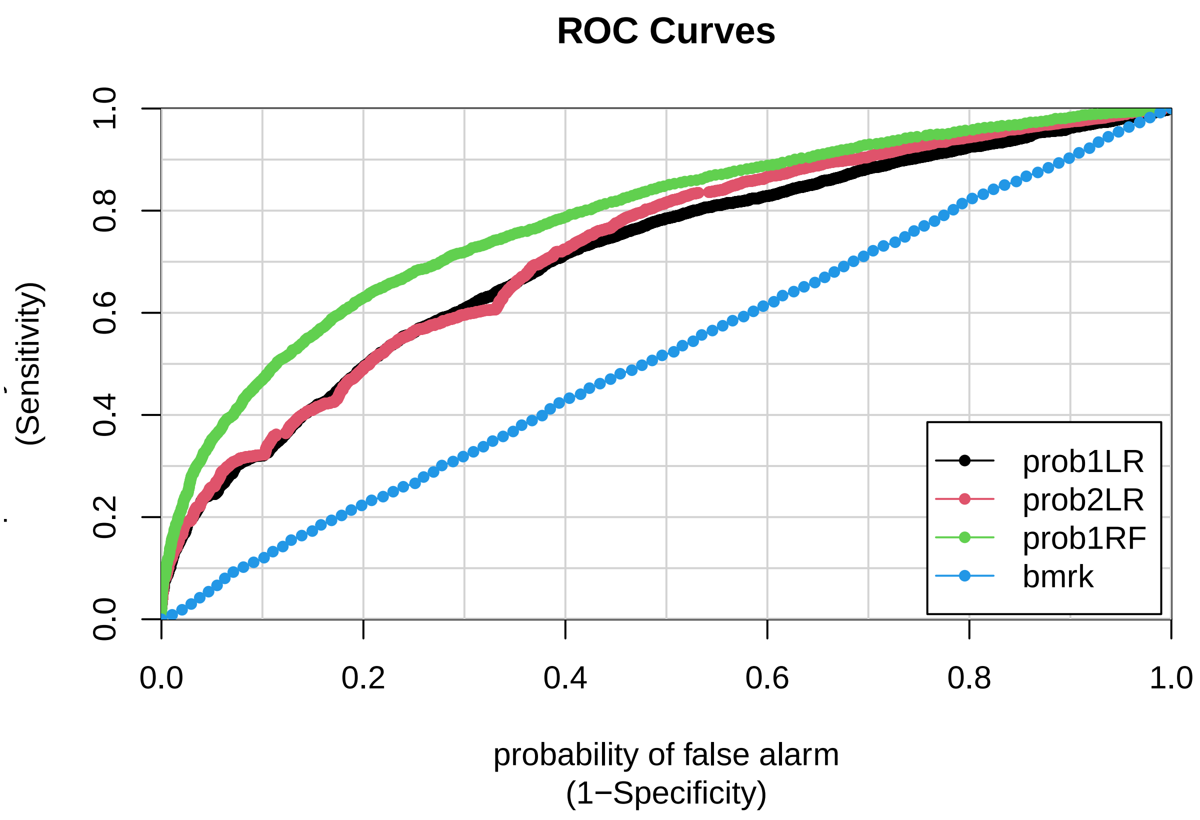
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    A
##          N 5635 1821
##          A  771 1903
##
##                Accuracy : 0.7441
##                  95% CI : (0.7355, 0.7526)
##     No Information Rate : 0.6324
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.4152
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5110
##             Specificity : 0.8796
##          Pos Pred Value : 0.7117
##          Neg Pred Value : 0.7558
##              Prevalence : 0.3676
##          Detection Rate : 0.1879
##    Detection Prevalence : 0.2640
##       Balanced Accuracy : 0.6953
##
##        'Positive' Class : A
##
```

In order to compare how different fitted model perform I will look at the ROC curves:

let's plot the ROC curve

```
bmrk = seq(0,1,by=0.01)
caTools::colAUC(cbind(prob1LR,prob2LR,prob1RF, bmrk), testSet$retwtBi,plotROC = T)
```



```
##            prob1LR   prob2LR   prob1RF      bmrk
## N vs. A 0.7098922 0.7234251 0.7813448 0.5166254
```

Area under the ROC curve can be used to compare various models in order to pick the optimal one and we conclude that random forest model performs the best on this metric as well. I also plotted the benchmark line for reference.

From the area under the ROC curve we can see that Logistic Regression model 2 (LR2) (fitted only on followers and mentions) performs better compared to LR1 model (fitted on all data). We can use cross-validation to try fitting different logistic regression models and seeing which one performs the best:

```r
set.seed(42)
# use cross-validation from the caret library
cv_model1 <- train(
  retwtBi ~ followers,
  data = trainSet,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

cv_model2 <- train(
  retwtBi ~ mentions,
  data = trainSet,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

cv_model3 <- train(
  retwtBi ~ hashtags,
  data = trainSet,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

cv_model4 <- train(
  retwtBi ~ score,
  data = trainSet,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

cv_model5 <- train(
  retwtBi ~ followers + mentions + hashtags,
  data = trainSet,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)
```

Let’s comapre the accuracy performance of different models now:

```r
summary(
  resamples(
    list(
      cv1 = cv_model1,
      cv2 = cv_model2,
      cv3 = cv_model3,
      cv4 = cv_model4,
      cv5 = cv_model5
    )
  )
)$statistics$Accuracy
```

```
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## cv1 0.6754444 0.6787360 0.6809211 0.6808819 0.6825134 0.6892693    0
## cv2 0.6254115 0.6285575 0.6302632 0.6300101 0.6315183 0.6346280    0
## cv3 0.6287031 0.6287031 0.6288252 0.6288252 0.6289474 0.6289474    0
## cv4 0.6287031 0.6287031 0.6288252 0.6288252 0.6289474 0.6289474    0
## cv5 0.6756579 0.6778014 0.6823568 0.6827901 0.6866776 0.6912442    0
```

We can see that the model with all three statistically significant parameters from fitting model LR2 has the best accuracy. However, using only the number of followers in the logistic regression model provides a decent accuracy as well, which indicates that this is the most important parameter for predicting whether the tweet is retweeted or not.

Summary of the Binary Retweet Analysis

We have used Logistic Regression to identify the most important tweet variables in determining whether it is retweeted or not. Such factors as number of followers, mentions of other users and number of hashtags turned out to be statiscatilly important. We further observed that number of **followers** is the most important mentric in predicting the propensity of the tweet to be retweeted.

We can try adding more information about the tweet to begin with (remember there are 90 different variables associated with a single tweet we can get) however, personally I am interested in seeing which tweets influence many people: i.e. what causes tweet to become "viral" and be retweeted hundreds or thousands of times and potentially have a significant influence on the perception of many people about COVID19. Thus, let's actually divide tweets which have been retweet into multiple sub-categories.

Multivariate Logit

First I changed the retweets into 4 categories (CAT) that define how "popular" the tweet is:

- "N (none)":0
- "S (small)":1-10
- "M (medium)":11-100
- "L (large)":>100

While other possible divisions exist (for example I included "viral" category initially with >1,000 retweets), I wanted to make sure that there are enough samples in each category for statistically significant results from the analysis.

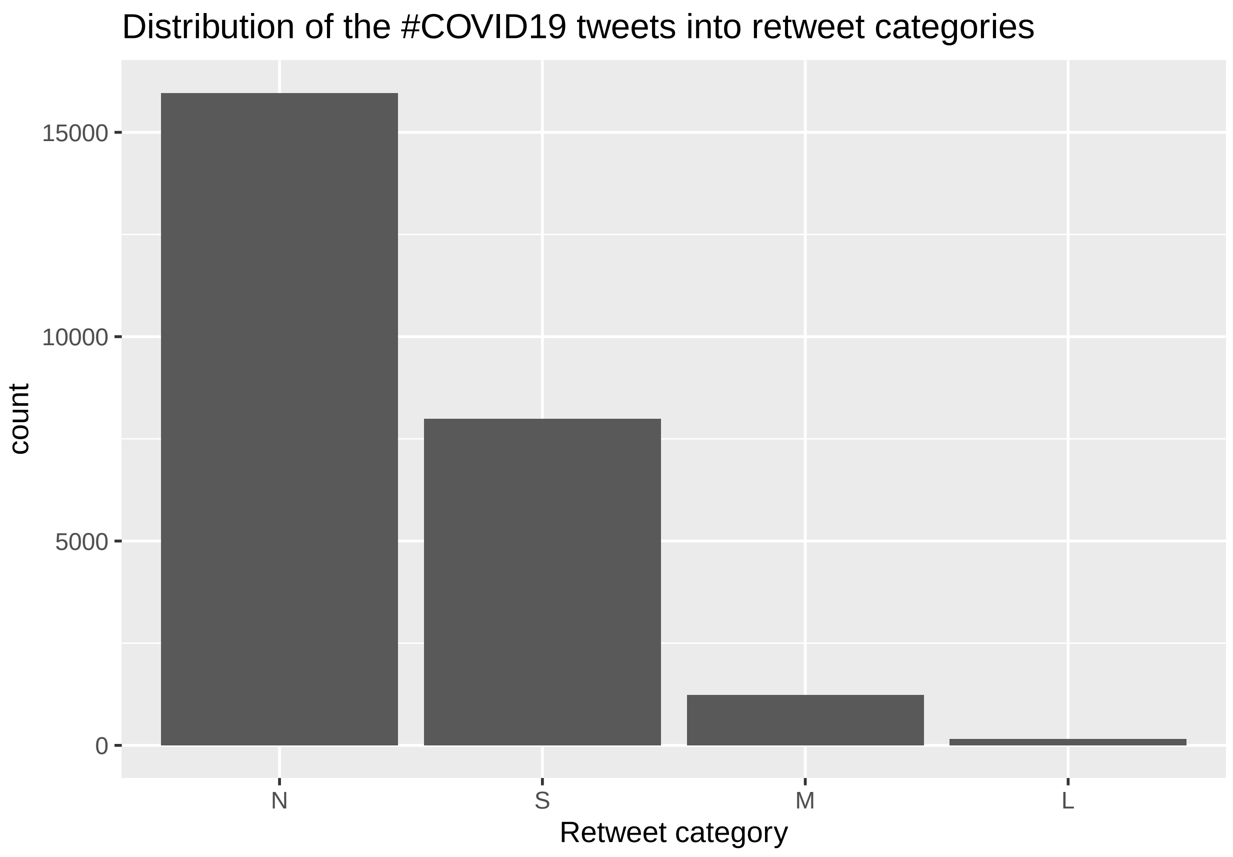
```
# afinn_five <- afinn_sentiment %>%
   # mutate(retwtCAT = ifelse(retweets == 0, "N",ifelse(retweets %in% 1:10, "S", ifelse(retweets %in% 1

# In order to do the random forest regression we cannot have any empty categories; so delete the Viral
afinn_four <- afinn_sentiment %>%
  mutate(retwtCAT = ifelse(retweets == 0, "N",ifelse(retweets %in% 1:10, "S", ifelse(retweets %in% 11:1
```

Lets convert the categories for retweets into fanked factors since there is an inherent order to them this order will be used with the multivariate logistic regression or can be used for ordinal logistic regression.

```r
# afinn_RK <- afinn_five %>%
#    mutate(retwtCAT = factor(retwtCAT,levels = c("N","S","M","L","V")))
# In case "L" category is empty use only 4 buckets
afinn_RK <- afinn_four %>%
 mutate(retwtCAT = factor(retwtCAT,levels = c("N","S","M","L")))
```

Let's see how many tweets per category there are in our sample:

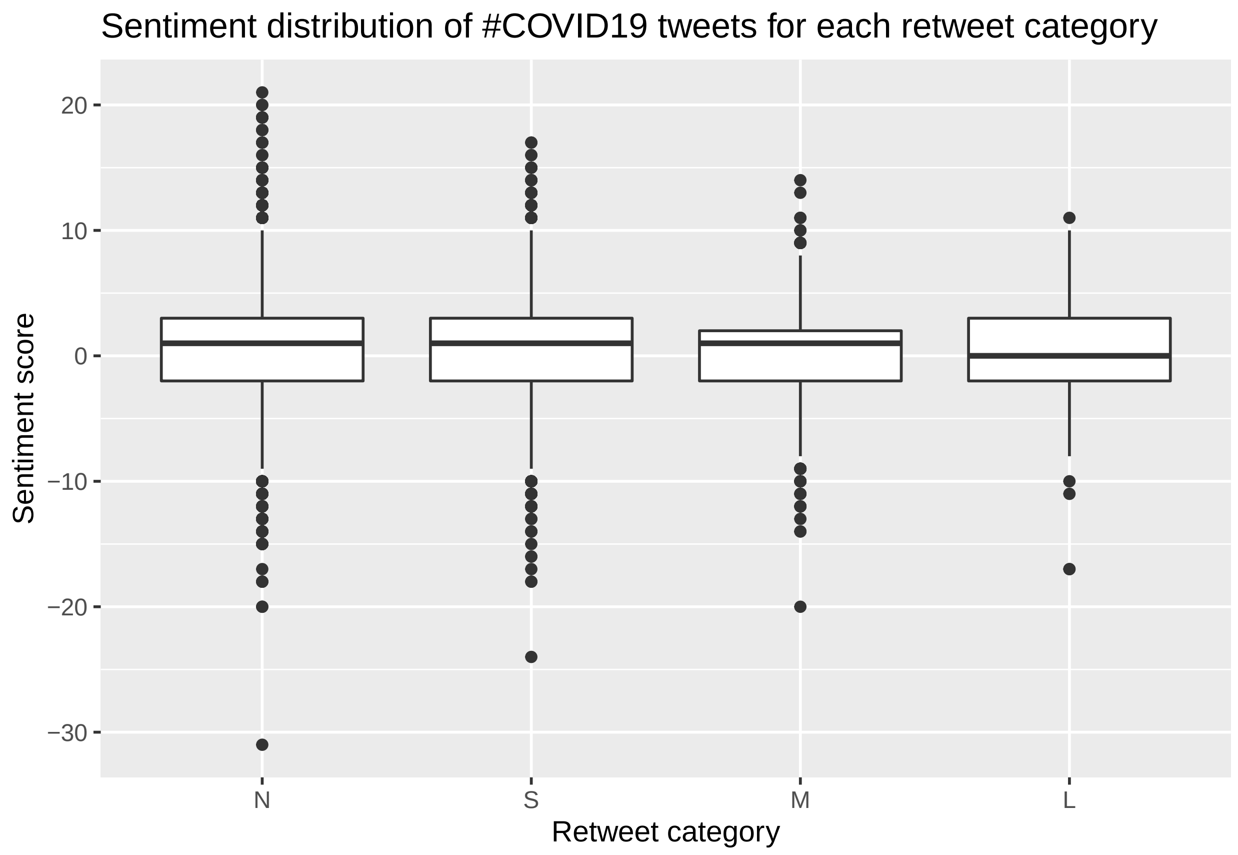
```r
afinn_RK %>%
  ggplot(aes(retwtCAT)) +
  geom_bar() +
  xlab("Retweet category") +
  ggtitle("Distribution of the #COVID19 tweets into retweet categories")
```



Most of the tweets appear to come from "N" and "S" categories which we should keep in mind when trying to predict tweets which will be retweeted more than 10 times. This makes sense intuitively since we are looking at the tweets posted recently only.

Let's display the sentiment distribution for each category

```r
afinn_RK %>%
  ggplot(aes(retwtCAT,score)) +
  geom_boxplot() +
  xlab("Retweet category") +
  ylab("Sentiment score") +
  ggtitle("Sentiment distribution of #COVID19 tweets for each retweet category")
```



Let's perform the multinomial logistic regression next since our oucome can be in one of the 4 categories: "N", "S", "M" or "L".

```
modelMLR_A <- multinom(retwtCAT ~ followers + friends + score + mentions + hashtags + urls, data = afin
```

```
## # weights:  32 (21 variable)
## initial  value 35107.904695
## iter  10 value 25428.153889
## iter  20 value 23004.192467
## iter  30 value 19690.253613
## iter  40 value 19590.157312
## iter  50 value 19585.351322
## iter  60 value 19584.941286
## final  value 19584.734081
## converged
```

```
summary(modelMLR_A)
```

```
## Call:
## multinom(formula = retwtCAT ~ followers + friends + score + mentions +
##     hashtags + urls, data = afinn_RK)
##
## Coefficients:
##   (Intercept)    followers      friends        score    mentions     hashtags
## S   -1.002714 9.991086e-06 6.768742e-06  0.003109297 0.132883412  0.007456708
## M   -2.398565 1.040022e-05 1.228556e-05 -0.034332470 0.127097504 -0.130969085
## L   -4.139833 1.045772e-05 1.317566e-05 -0.041228149 0.001579279 -0.173894369
##          urls
## S  0.01559093
## M -0.39711328
## L -1.05238796
##
## Std. Errors:
##    (Intercept)    followers      friends        score     mentions     hashtags
## S 1.077502e-10 4.497181e-07 1.843898e-06 2.689415e-11 4.780264e-11 4.971897e-10
## M 4.337142e-11 4.502617e-07 2.259948e-06 2.187849e-11 2.653308e-11 1.388035e-10
## L 2.166419e-11 4.506841e-07 3.478151e-06 1.951601e-11 8.852119e-12 5.846691e-11
##           urls
## S 7.831633e-11
## M 2.868787e-11
## L 8.412770e-12
##
## Residual Deviance: 39169.47
## AIC: 39211.47
```

Notice that for all the fit coefficients are referenced to the "N" category:

$$\log \frac{Pr(CAT)}{Pr(N)} = \beta_0 + \beta^T x$$

where CAT is "S", "M" or "L". Let's estimate the statistical significance of the fit coefficients using the p-value for Wald's test:

```r
z <- summary(modelMLR_A)$coefficients/summary(modelMLR_A)$standard.errors

z
```

```
##     (Intercept) followers  friends       score   mentions    hashtags
## S   -9305912030  22.21633 3.670888   115612403 2779834237    14997714
## M  -55302895786  23.09817 5.436216 -1569233977 4790153508  -943557423
## L -191091020168  23.20410 3.788123 -2112529314  178406853 -2974235520
##            urls
## S     199076327
## M  -13842550907
## L -125094113561
```

```r
p <- (1 - pnorm(abs(z), 0, 1)) * 2
p
```

```
##   (Intercept) followers      friends score mentions hashtags urls
## S           0         0 2.417092e-04     0        0        0    0
## M           0         0 5.442402e-08     0        0        0    0
## L           0         0 1.517898e-04     0        0        0    0
```

It looks like coefficients for followers, friends, sentiment score, mentions, hashtags and urls are statistically significant. Let's now look in more details at each category but first we need to add additional binary indicators to the dataset:

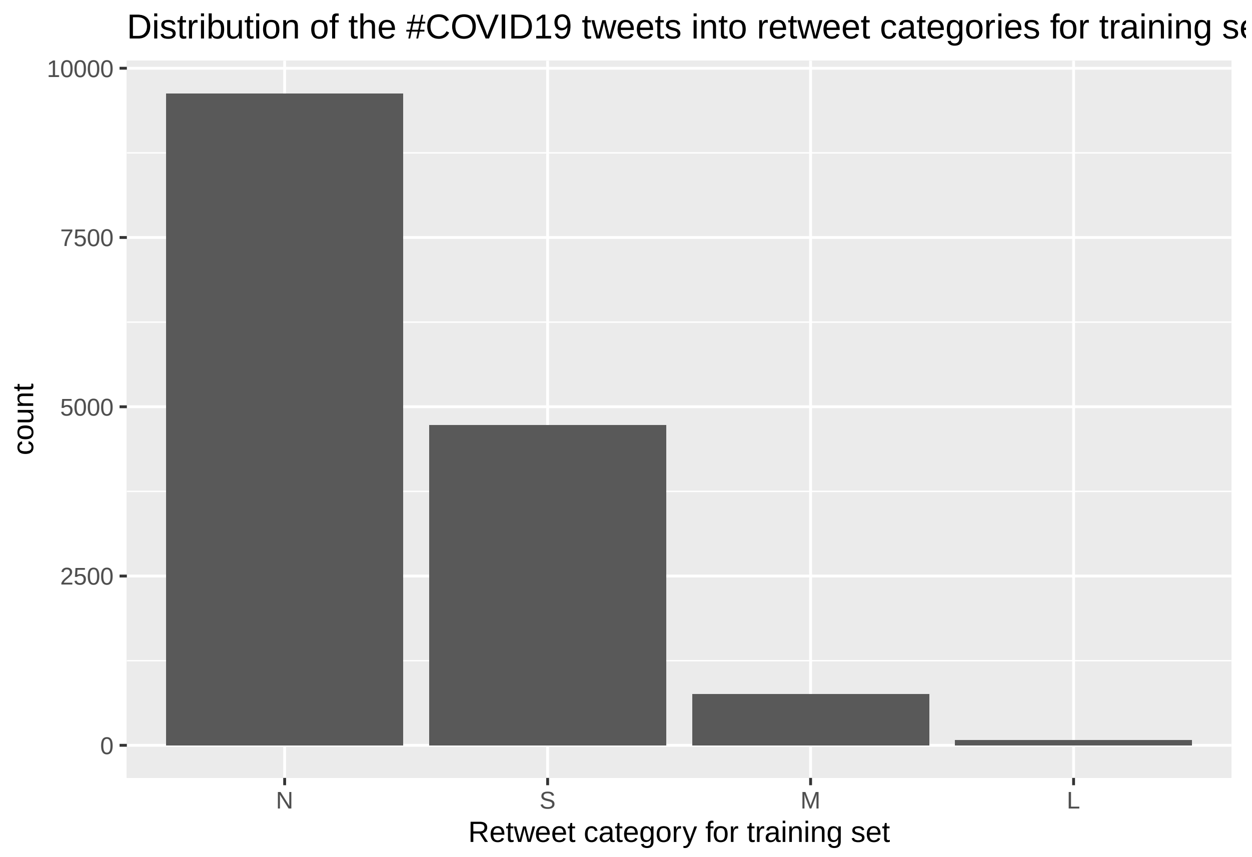
```r
afinn_RK <- afinn_RK %>%
  mutate(isSmall = ifelse(retwtCAT == "S", 1, 0)) %>%
  mutate(isMedium = ifelse(retwtCAT == "M", 1, 0)) %>%
  mutate(isLarge = ifelse(retwtCAT == "L", 1, 0))
```

Split the data into train and test sets:

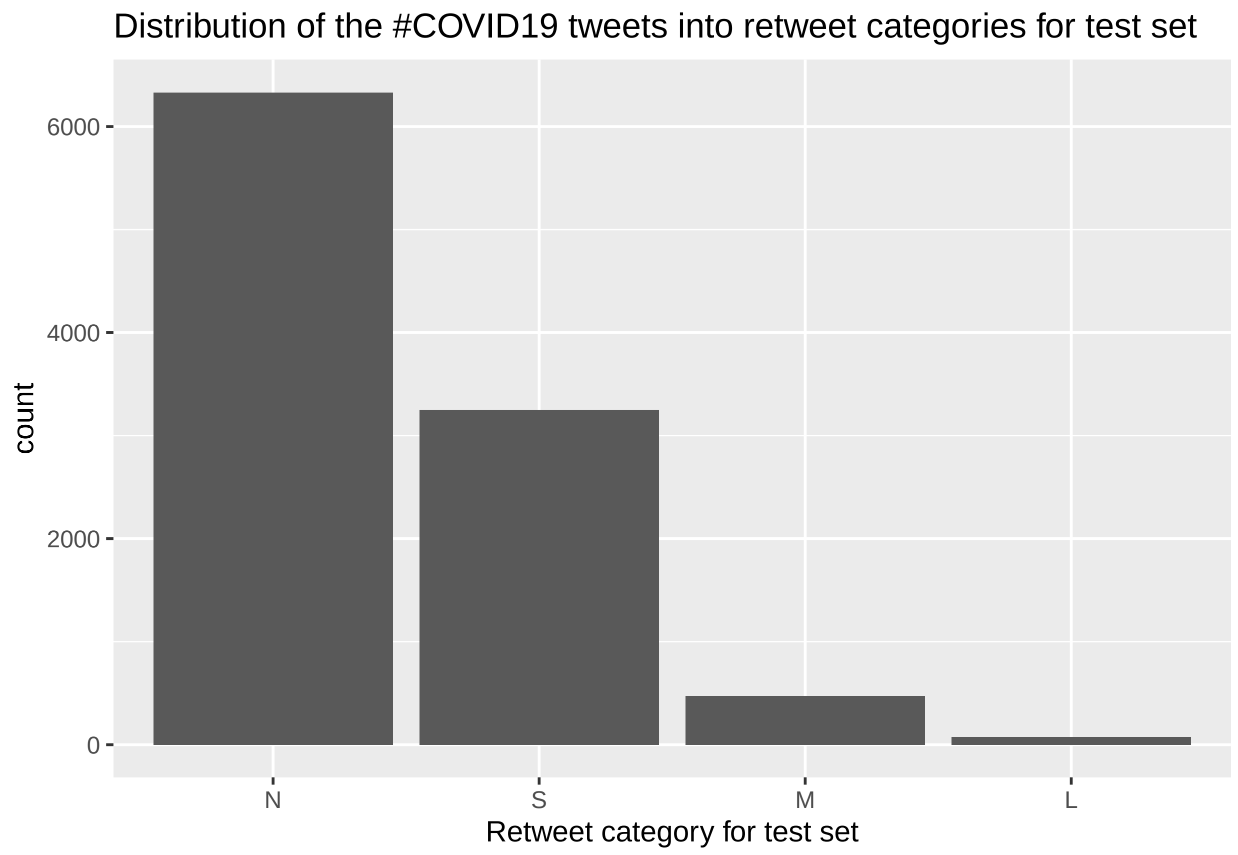
```
tr2 <- sample(nrow(afinn_RK),round(nrow(afinn_RK)*0.6)) # split into training and test subsets
trainRK <- afinn_RK[tr2,]
testRK <- afinn_RK[-tr2,]
```

Let's see how many tweets per category there are in our training and test samples to make sure that the distributions are representative of the total sample:

```
trainRK %>%
  ggplot(aes(retwtCAT)) +
  geom_bar() +
  xlab("Retweet category for training set") +
  ggtitle("Distribution of the #COVID19 tweets into retweet categories for training set")
```



```
testRK %>%
  ggplot(aes(retwtCAT)) +
  geom_bar() +
  xlab("Retweet category for test set") +
  ggtitle("Distribution of the #COVID19 tweets into retweet categories for test set")
```



As we can see from the histograms, the relative distributions look very similar, which provides strong indication that the best model training and benchmarked will be extendable to other future tweet analysis on this topic.

```
modelRK_LR1 <- glm(isSmall ~ followers + friends + hashtags + mentions + urls + score, data = trainRK, f
summary(modelRK_LR1)
```

```
##
## Call:
## glm(formula = isSmall ~ followers + friends + hashtags + mentions +
##     urls + score, family = binomial(link = "logit"), data = trainRK)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0322  -0.8582  -0.8109   1.4685   1.6834
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.562e-01  3.297e-02 -28.997  < 2e-16 ***
## followers    3.781e-08  2.190e-08   1.726   0.0843 .
## friends      1.028e-05  1.777e-06   5.784 7.30e-09 ***
## hashtags    -5.496e-03  5.521e-03  -0.995   0.3195
## mentions     1.189e-01  1.289e-02   9.224  < 2e-16 ***
## urls         1.369e-01  3.264e-02   4.193 2.76e-05 ***
## score        7.363e-03  4.891e-03   1.505   0.1322
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 18849  on 15194  degrees of freedom
## Residual deviance: 18698  on 15188  degrees of freedom
## AIC: 18712
##
## Number of Fisher Scoring iterations: 4
```

```
modelRK_LR2 <- glm(isMedium ~ followers + friends + hashtags + mentions + urls + score, data = trainRK,
summary(modelRK_LR2)
```

```
##
## Call:
## glm(formula = isMedium ~ followers + friends + hashtags + mentions +
##     urls + score, family = binomial(link = "logit"), data = trainRK)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8883  -0.3345  -0.2947  -0.2474   3.3855
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.547e+00  7.424e-02 -34.306  < 2e-16 ***
## followers    5.325e-07  3.519e-08  15.132  < 2e-16 ***
## friends      9.379e-06  2.045e-06   4.587 4.50e-06 ***
## hashtags    -1.374e-01  1.889e-02  -7.277 3.42e-13 ***
## mentions     6.064e-02  2.489e-02   2.436  0.01483 *
## urls        -3.644e-01  7.640e-02  -4.770 1.84e-06 ***
## score       -2.865e-02  1.065e-02  -2.691  0.00713 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5999.0  on 15194  degrees of freedom
## Residual deviance: 5502.2  on 15188  degrees of freedom
## AIC: 5516.2
##
## Number of Fisher Scoring iterations: 6
```

```
modelRK_LR3 <- glm(isLarge ~ followers + friends + hashtags + mentions + urls + score, data = trainRK, 
summary(modelRK_LR3)
```

```
##
## Call:
## glm(formula = isLarge ~ followers + friends + hashtags + mentions +
##     urls + score, family = binomial(link = "logit"), data = trainRK)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1977  -0.1181  -0.0818  -0.0655   3.5384
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.457e+00  2.152e-01 -20.708  < 2e-16 ***
## followers    3.333e-07  3.375e-08   9.875  < 2e-16 ***
## friends      5.907e-06  4.452e-06   1.327  0.18455
## hashtags    -1.620e-01  6.215e-02  -2.607  0.00913 **
## mentions    -8.843e-02  1.047e-01  -0.845  0.39833
## urls        -1.112e+00  2.587e-01  -4.300 1.71e-05 ***
## score       -3.315e-02  3.103e-02  -1.068  0.28536
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 999.05  on 15194  degrees of freedom
## Residual deviance: 907.69  on 15188  degrees of freedom
## AIC: 921.69
##
## Number of Fisher Scoring iterations: 9
```

Let's look at the confusion matrices for "S", "M" and "L" categories:

```
probS_LR1 <- predict(modelRK_LR1,testRK, type = "response")
clS_LR1 <- ifelse(probS_LR1 > 0.5, 1, 0) # predicted binary classification
confusionMatrix(factor(clS_LR1), factor(testRK$isSmall),positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6825 3221
##          1   55   29
##
##                Accuracy : 0.6766
##                  95% CI : (0.6674, 0.6857)
##     No Information Rate : 0.6792
##     P-Value [Acc > NIR] : 0.7139
##
##                   Kappa : 0.0013
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.008923
##             Specificity : 0.992006
##          Pos Pred Value : 0.345238
##          Neg Pred Value : 0.679375
##              Prevalence : 0.320829
##          Detection Rate : 0.002863
##    Detection Prevalence : 0.008292
##       Balanced Accuracy : 0.500464
##
##        'Positive' Class : 1
##
```

```
probM_LR2 <- predict(modelRK_LR2,testRK, type = "response")
clM_LR2 <- ifelse(probM_LR2 > 0.5, 1, 0) # predicted binary classification
confusionMatrix(factor(clM_LR2), factor(testRK$isMedium),positive = '1')
```

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

```
##
##           Reference
## Prediction    0    1
##          0 9633  447
##          1   22   28
##
##                Accuracy : 0.9537
##                  95% CI : (0.9494, 0.9577)
##     No Information Rate : 0.9531
##     P-Value [Acc > NIR] : 0.4006
##
##                   Kappa : 0.0986
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.058947
##             Specificity : 0.997721
##          Pos Pred Value : 0.560000
##          Neg Pred Value : 0.955655
##              Prevalence : 0.046890
##          Detection Rate : 0.002764
##    Detection Prevalence : 0.004936
##       Balanced Accuracy : 0.528334
##
##        'Positive' Class : 1
##
```

```
probL_LR3 <- predict(modelRK_LR3,testRK, type = "response")
clL_LR3 <- ifelse(probL_LR3 > 0.5, 1, 0) # predicted binary classification
confusionMatrix(factor(clL_LR3), factor(testRK$isLarge),positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10054    73
##          1     3     0
##
##                Accuracy : 0.9925
##                  95% CI : (0.9906, 0.9941)
##     No Information Rate : 0.9928
##     P-Value [Acc > NIR] : 0.6653
##
##                   Kappa : -6e-04
##
##  Mcnemar's Test P-Value : 2.476e-15
##
##             Sensitivity : 0.0000000
##             Specificity : 0.9997017
##          Pos Pred Value : 0.0000000
##          Neg Pred Value : 0.9927915
##              Prevalence : 0.0072063
##          Detection Rate : 0.0000000
##    Detection Prevalence : 0.0002962
##       Balanced Accuracy : 0.4998509
```

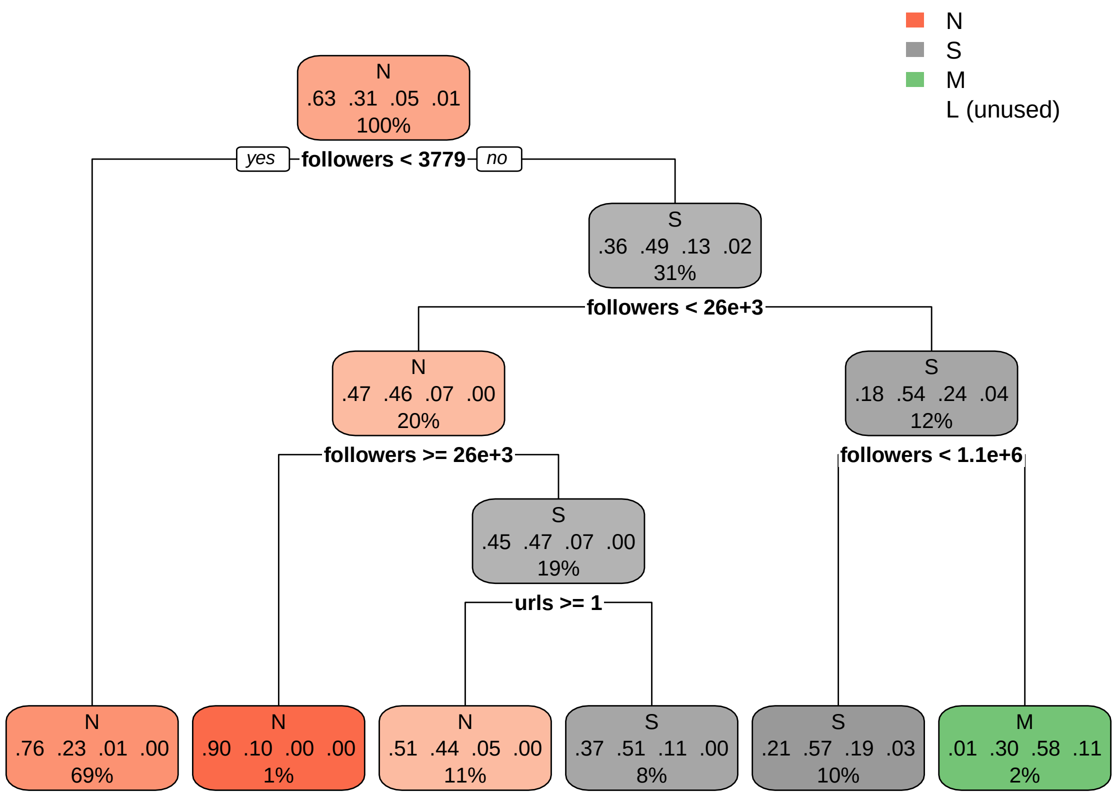
```
##
##        'Positive' Class : 1
##
```

It is important to note that while the accuracy for the logit model to predict "L" category is the highest, it is not a useful metric in this case before there are very little "L" category tweets. Therefore, we should focus on sensitivity and specificity instead as useful metrics for predicing categories with "S", "M" and "L" indicators.

Decision trees to determine retweet categories

Before we proceed with model fitting analysis we need to split the retweet categorical data into train and test subsets:

```
modelRK_DT <- rpart(retwtCAT ~ followers + friends + score + mentions + hashtags + urls, data = trainRK

predRK_DT <- predict(modelRK_DT,testRK, type = "class")

rpart.plot(modelRK_DT)
```



```
confusionMatrix(predRK_DT, testRK$retwtCAT)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    N    S    M    L
##          N 5797 2207  107    8
##          S  534  988  289   44
##          M    1   55   79   21
##          L    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.6776
##                  95% CI : (0.6684, 0.6867)
##     No Information Rate : 0.6251
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2665
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: N Class: S Class: M Class: L
## Sensitivity            0.9155  0.30400 0.166316 0.000000
## Specificity            0.3886  0.87398 0.992025 1.000000
## Pos Pred Value         0.7140  0.53261 0.506410      NaN
## Neg Pred Value         0.7340  0.72665 0.960297 0.992794
## Prevalence             0.6251  0.32083 0.046890 0.007206
## Detection Rate         0.5723  0.09753 0.007799 0.000000
## Detection Prevalence   0.8015  0.18312 0.015400 0.000000
## Balanced Accuracy      0.6521  0.58899 0.579170 0.500000
```

In fact it looks like a simple decision tree performs much better in predicting tweets in “S” and “M” categories. By comparing the $F_1 = \frac{2}{1/sensitivity + 1/specificity}$ score of the decision tree fit with the one-vs-all logistic regression models above, we can see that a nonlinear model performs much better here.

** KNN classifier

I was interested in seeing how k-nearest-neighbors classifier will perform on this multivariate classification challenge for retweet categories. However, before we can efficiently apply it on our tweet data set, I needed to transform the tweet features since they are of very diferent scales: while mentions and urls are usually a small number, followers can be in the hundreds or thousands. Moreover, sentiment score can be positive or negative while all the other variables can only take values larger than zero. Thus, I rescaled and centered the features before applying knn classification:

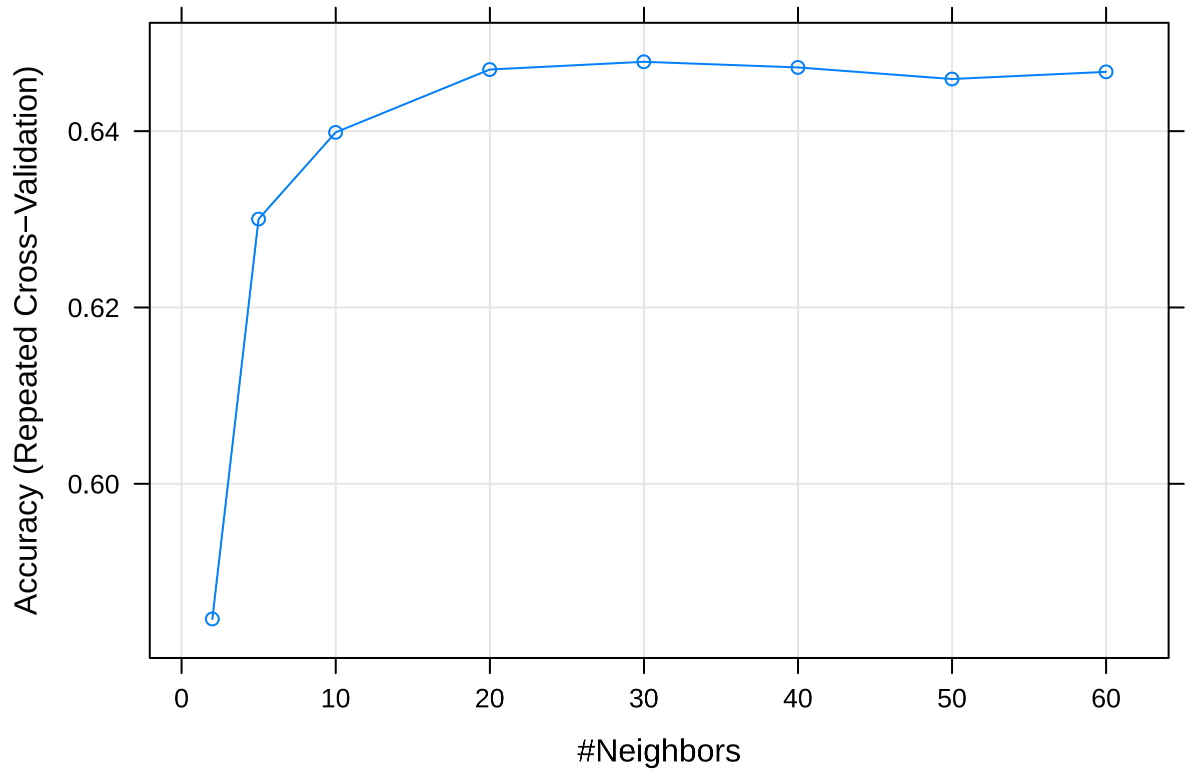
```
trCtrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

knn_fit <- train(retwtCAT ~ followers + friends + score + mentions + hashtags + urls, data = trainRK, me
                 trControl = trCtrl,
                 preProcess = c("center","scale"),
                 tuneGrid = expand.grid(k = c(2,5,10,20,30,40,50,60)))
```

Let’s look at the results now:

```
knn_fit
```

```
## k-Nearest Neighbors
##
## 15195 samples
##     6 predictor
##     4 classes: 'N', 'S', 'M', 'L'
##
## Pre-processing: centered (6), scaled (6)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 13675, 13676, 13675, 13675, 13675, 13676, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    2  0.5846670  0.14656302
##    5  0.6300316  0.17496674
##   10  0.6398598  0.16478954
##   20  0.6469897  0.14454859
##   30  0.6478666  0.12619739
##   40  0.6472309  0.11180705
##   50  0.6459149  0.10142414
##   60  0.6467265  0.09903681
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 30.
```

```r
plot(knn_fit)
```



It looks like using 30 neighbors provides the highest accuracy. However, we need to see how out model performs on the test data.

```r
test_pred <- predict(knn_fit, testRK)
confusionMatrix(test_pred, testRK$retwtCAT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    S    M    L
##          N 5968 2741  295   36
##          S  363  471  118   17
##          M    1   38   62   20
##          L    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.6418
##                  95% CI : (0.6323, 0.6511)
##     No Information Rate : 0.6251
##     P-Value [Acc > NIR] : 0.0002622
##
##                   Kappa : 0.1282
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: N Class: S Class: M Class: L
## Sensitivity            0.9425  0.14492  0.13053 0.000000
## Specificity            0.1912  0.92762  0.99389 1.000000
## Pos Pred Value         0.6602  0.48607  0.51240      NaN
## Neg Pred Value         0.6661  0.69665  0.95874 0.992794
## Prevalence             0.6251  0.32083  0.04689 0.007206
## Detection Rate         0.5891  0.04650  0.00612 0.000000
## Detection Prevalence   0.8924  0.09566  0.01194 0.000000
## Balanced Accuracy      0.5668  0.53627  0.56221 0.500000
```

Caret library makes it easy to try other machine learning methods. RandomForest performed well for the binary classification problem whether tweet will get retweeted or not so I excpeted it to do well here to. Let's see how it performs. Notice that I chose not too rescale the features since randomforest should be robust against different scales in the variables.

```
trCtrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = 'random')

rf_fit <- train(retwtCAT ~ followers + friends + score + mentions + hashtags + urls, data = trainRK, me
                trControl = trCtrl,
                metric = 'Accuracy',
                tuneLength = 10)
```

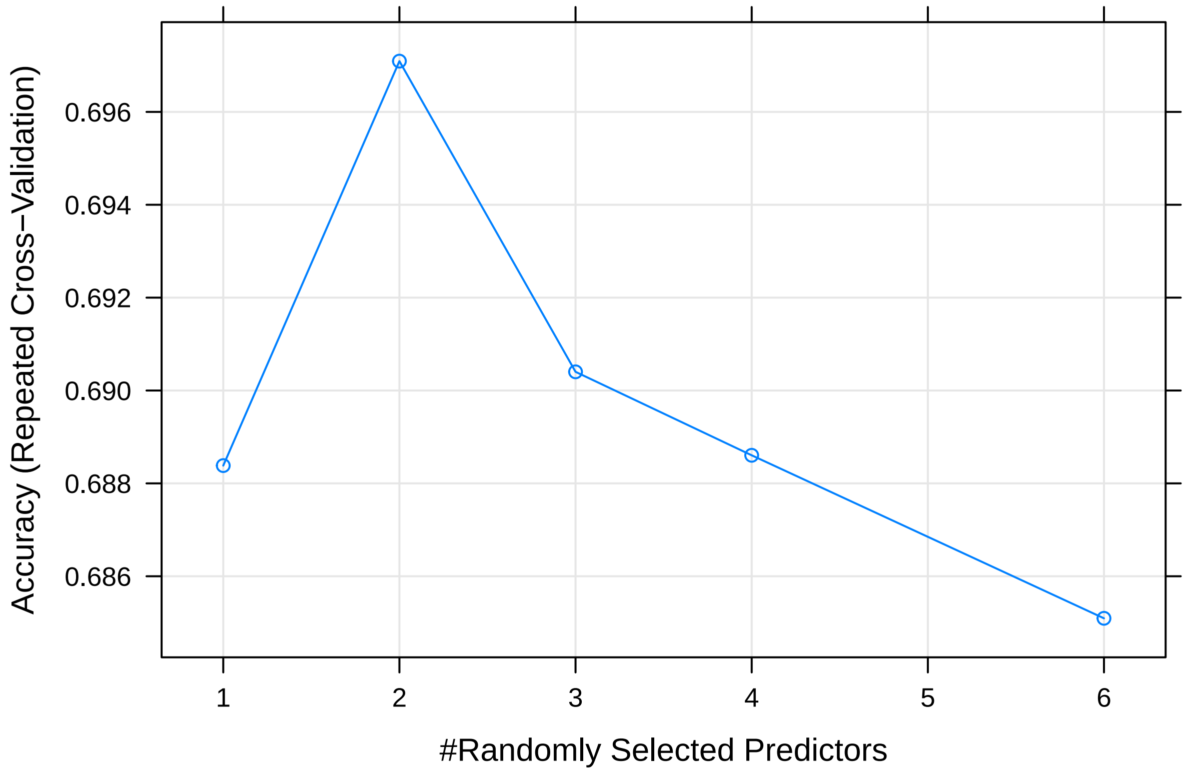
Let's look at the results:

```
rf_fit
```

```
## Random Forest
##
## 15195 samples
##     6 predictor
##     4 classes: 'N', 'S', 'M', 'L'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 13676, 13676, 13676, 13674, 13676, 13675, ...
## Resampling results across tuning parameters:
##
```

```
##   mtry  Accuracy   Kappa
##   1     0.6883834  0.2588233
##   2     0.6970931  0.3271509
##   3     0.6904026  0.3220294
##   4     0.6886047  0.3207931
##   6     0.6850942  0.3153899
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
plot(rf_fit)
```



Let's see what the out-of-sample error is:

```
test_pred <- predict(knn_fit, testRK)
confusionMatrix(test_pred, testRK$retwtCAT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    S    M    L
##          N 5975 2740  300   35
##          S  356  471  111   18
##          M    1   39   64   20
##          L    0    0    0    0
##
## Overall Statistics
##
```

```
##                Accuracy : 0.6426
##                  95% CI : (0.6332, 0.652)
##     No Information Rate : 0.6251
##     P-Value [Acc > NIR] : 0.0001291
##
##                   Kappa : 0.1299
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: N Class: S Class: M Class: L
## Sensitivity            0.9436  0.14492 0.134737 0.000000
## Specificity            0.1904  0.92951 0.993786 1.000000
## Pos Pred Value         0.6602  0.49268 0.516129      NaN
## Neg Pred Value         0.6694  0.69708 0.958925 0.992794
## Prevalence             0.6251  0.32083 0.046890 0.007206
## Detection Rate         0.5898  0.04650 0.006318 0.000000
## Detection Prevalence   0.8934  0.09437 0.012241 0.000000
## Balanced Accuracy      0.5670  0.53721 0.564261 0.500000
```

Summary of the retweet degree prediction

By performing one-vs-all logistic regression I identified that different features are important for predicting retweet outcome classes: for example, urls, hashtags and followers are statistically significant in order to predict tweets with >100 retweets, while mentions, friends and urls are the most important features for predicting the 1-10 retweet category. While I found that other tweet parameters influence the number of retweets much more than the sentiment score, analyzing the sentiment of the tweets can still be important for understanding the public opinion and it's change over time. Below I look at the tweet sentiments using "bing" lexicon instead.

Use Bing lexicon for sentiment analysis

While "bing" lexicon only specifies whether a word is positive or negative it still can be useful for sentiment analysis

now lets try using "bing" lexicon

twts_bing <- twts_clean %>% inner_join(get_sentiments("bing"), by = 'word')

twts_bing %>% count(word, sentiment, sort = TRUE) %>% acast(word ~ sentiment, value.var = "n", fill = 0) %>% comparison.cloud(colors = c("red","blue"),max.words =100)

twts_bing %>% ggplot(aes(retweets)) + geom_bar() + xlim(0,50) scale_x_log10()

retwts_bing_sent <- twts_bing %>% group_by(id,retweets,followers,golden) %>% # filter(retweets > 5) %>% count(id,retweets,sentiment) %>% spread(sentiment, n, fill = 0) %>% mutate(sentiment = positive - negative) %>% mutate(word_cnt = positive + negative)

retwts_bing_sent %>% ggplot(aes(sentiment,retweets)) + geom_bar(stat = "identity")

twts_bing %>% group_by(sentiment) %>% top_n(10, retweets) %>% arrange(retweets) %>% ungroup() %>% mutate(word = factor(word, unique(word))) %>% ungroup() %>% ggplot(aes(word, retweets, fill = sentiment)) + geom_col(show.legend = FALSE) + facet_wrap(~ sentiment, scales = "free", ncol = 2) + coord_flip() + labs(x = NULL, y = "Median # of retweets for tweets containing each word")

bing_word_counts <- twts_bing %>% count(word, sentiment, sort = TRUE) %>% ungroup()

```
bing_word_counts %>% group_by(sentiment) %>% top_n(10) %>% ungroup() %>% mutate(word =
reorder(word, n)) %>% ggplot(aes(word, n, fill = sentiment)) + geom_col(show.legend = FALSE) +
facet_wrap(~sentiment, scales = “free_y”) + labs(y = “Contribution to sentiment”, x = NULL) + coord_flip()
```