Using Machine Learning Methods to Assess the Imapact of Twitter Sentiment on Stock Prices

Sarah Kim, Arturo Valera, Pipi Gao

12/13/2021

Abstract

Our project goal is to understand to what extent Twitter can reflect a company's value. We wanted to examine the value of a company from two approaches. One way was to examine how the influential figures of each company, such as the CEOs, utilize Twitter space to promote values of a company. Our finding was that not all CEO use Twitter space to promote their brand values. Second approach was to look at the stock value of a company. We were curious to see to what extent the public sentiment of a company is reflected on Twitter, and how these sentiments may or may not influence the companies' stock prices. We concluded that Twitter sentiments alone cannot capture or influence the rise or the fall of company stock prices. However, we were able to conclude that the Twitter volume is a significant predictor of stock price rise or fall.

Introduction

The importance of numerous social media platforms has risen over the past years. As more people spend time online, the more information about people's preferences, dislikes, societal trends, and opinions have been collected in these platforms in real time. There are numerous social media platforms, such as Twitter, Facebook, Reddit, Youtube etc, that allow people to openly share their opinions and values. We chose Twitter as our main social media platform to gain insights about real-time information regarding current societal trends and opinions about companies. As of 2021, Twitter has 396.5 million users among which 206 million users access Twitter daily. Twitter is a rich source of real-time information regarding current trends and opinions. Our project will provide relavant insights to whether Twitter sentiment can bring light to the stock market trend.

The motivating question for our project was "to what extent does Twitter capture the value of a company?" Our goal was to analyze social media data about different companies and predict its future stock trend using sentiment analysis and machine learning methods. We were interested to see how influential figures like CEO's utilizes Twitter space to promote company values. We also wanted to examine to what extent Twitter sentiments influence the stock market trend for these companies. We examined this problem in three parts:

First, we will be looking at the Twitter contents and comments for several companies of our choice, such as Apple, Microsoft, Amc, Amazon, and Twitter. Looking into each CEO's Twitter posts and their comments, we will be building Wordclouds to visualize the most frequently used words on their Twitter spaces in order to gauge to what extent these influential figures utilize the Twitter space to promote values of their company.

Second, we will conduct sentiment analysis on the public's discussions of the hashtags for each companies' stock code from the CEO's Twitter posts. We are curious to see whether the Twitter sentiment on these posts and hashtags reflect the change in stock market prices for these companies. Our hypothesis is that Twitter is a pivotal factor in influencing and shaping perceptions; therefore, Twitter sentiment will affect stock market trends.

Finally, we looked at two different datasets to verify our hypothesis. (More information about our data will be explained in the next section "Data"). Due to the limitation of the Twitter API, we were only able to look at the change in Twitter sentiment and stock prices for only a week. One week is too short to generalize the validity of our hypothesis for all time; therefore, we also looked at a Kaggle dataset that already contains information about twitter sentiment and stock prices for Apple Inc from the period 2016 to 2019. We then created classification models to verify our hypothesis.

Data

(1) Twitter API v2 Our Twitter data was obtained using the Twitter API v2 which can be accessed by signing up for a developer account. There are different accounts for different purposes. As college students, we were allowed to use the "Elevated Developer Account". We have access to some historical and real time data on variables such as tweets, users, spaces, lists, trends ...etc. However, we are limited in terms of the number of requests we can make within a specific time period. We also are limited in terms of the number of posts we can obtain from the pacakge. The data we've obtained from the package was used for generating wordclouds and conducting sentiment analysis.

```
api_key <- '37osUadblZABTOkc2mqlhERZu'
api_secret_key <- 'k2HdTUg5GERupy8iUyOdrd8Z2J67S4qUeNaZhbgvCSd0p3JmY6'
access_token <- '1453471187653201923-i7wAcAEJSI1youyTHsObUPNuDZENXH'
access_secret_token <- 'Da6L4g5a4IlqEQRpMlLla2ZgvgZLDXd19SaCLULWzauCR'
appname <- "comp_stats_proj"
key <- '37osUadblZABTOkc2mqlhERZu'
secret <- 'k2HdTUg5GERupy8iUyOdrd8Z2J67S4qUeNaZhbgvCSd0p3JmY6'

twitter_token <- create_token(
  app = appname,
  consumer_key = key,
  consumer_secret = secret,
  access_token = access_token,
  access_secret = access_secret_token
)
```

(2) Alpha Vantage API We used Alpha Vantage to acquire stock data for Apple, Microsoft, Tesla, Amazon, and Twitter. The dataframe includes timestamps of transaction intervals, open and close prices, and the volume of transaction within each interval.

```
#alpha vantage api key
av_api_key("YOUR_API_KEY")
```

(3) Kaggle Dataset The data set used for our classification models was provided by kaggle and can be found at the following link: https://www.kaggle.com/nadun94/twitter-sentiments-aapl-stock.

The dataset has data on AAPL stock from 2016 to 2019 on a daily basis. It has the date, the price when the stock market opened, the lowest and highest value of the stock recorded that day, the price when the stock market closed, the volume of stocks traded, and twitter sentiment (i.e. ts_polarity assigns a numeric value to the perceived sentiment of a tweet - positive or negative), and twitter volume. We will be using this dataset to build classification models that verifies our hypothesis–Twitter sentiment does/does not influence a company's stock trend.

Analysis

Twitter Posts and Comments Due to the internal limit in the number of posts we can scrap using the Twitter pacakge, we were only able to take most, but not all, of the recent Twitter posts from each CEO of our interest.

```
## CEO's Twitter Posts
timcook <- get_timelines("tim_cook", n = 1255) #AAPL
billgates <- get_timeline("BillGates", n=3719)#MSFT
adamarom <- get_timelines("CEOAdam", n = 4660)#AMC
paraga <- get_timelines("paraga", n = 3243 )#Twitter
```

Building a wordcloud Word Cloud

We created a function that takes data set of tweeter posts and generates a wordcloud

```
exportWordCloud <- function(tweet_category){
  tweets <- tweet_category # make a copy
  tweets <- tweets %>% dplyr::select("text") # selects just the text column
  tweets$text <- gsub("[^[:alnum:][:blank:]?&/\\-]", "",tweets$text) # remove alphanumeric characters
  tweets$text <- gsub("https\\S*", "",tweets$text) # remove hyperlinks
  tweets$text <- gsub("amp", "",tweets$text) # amp just keeps showing up, remove it!!
  #create a corpus to allow us clean the text column with tm
  tweets.corpus <- Corpus(VectorSource(tweets$text))
  tweets.corpus <- tweets.corpus %>%
    tm_map(removeNumbers) %>% # removes numbers from text
    tm_map(removePunctuation) %>% # removes punctuation from text
    tm_map(stripWhitespace) %>% # trims the text of whitespace
    tm_map(content_transformer(tolower)) %>% # convert text to lowercase
    tm_map(removeWords,stopwords("english")) %>% # remove stopwords
    tm_map(removeWords,stopwords("SMART")) # remove stopwords not removed from previous line
    tdm <- TermDocumentMatrix(tweets.corpus) %>% # create a term document matrix
    as.matrix()
  words <- sort(rowSums(tdm), decreasing = TRUE) # count all occurences of each word and group them
  df <- data.frame(word = names(words), freq = words) # convert it to a dataframe
  set.seed(1234) # for reproducibility, sorta
  wcloud <- wordcloud2(df,   # generate word cloud
                       size = 1,
                       color= 'random-dark', # set colors
                       rotateRatio = 0) #horizontal looks better, but what do you think?
  wcloud
}
```

After building the function for generating wordcloud, we called each CEO's Twitter account to visualize the most frequently used words on their accounts. It was interesting to see that CEO's utilize their Twitter space for both personal use as much as they do for their own company promotions. Adam Arom, the CEO of AMC, for example, mentions "Sixers" the most in his Twitter account because he is a huge fan of "Sixers." However, we could also see promotion of values and social good in Twitter posts from Billgates. In his posts, the most frequently mentioned words include "People, world, progress". Tim Cook from Apple and Parag Agrawal from Twitter mentions their brand name the most along with positive words like "Great, Good, Proud." Perhaps, these posts do contribute in constructing an overall perception of a company.

```
# exportWordCloud(timcook)
#exportWordCloud(billgates)
#exportWordCloud(adamarom)
#exportWordCloud(paraga)
```

In order to see how the public is reacting to these posts from these influential figures of their companies, we then looked at the replies and retweets from these posts. However, to our surprise, we observed that a lot of the retweets do not contain many comments because people tend to re-tweet without commenting much on the post. As for the replies, we had to use a mix of Python and R package to manually acquire replies from each post (using the Tweet ID). When we automated the process, we ran into a problem because Twitter limits the number of request we can make to the Twitter API. We therefore could not proceed with our original plan of evaluating sentiment from these replies. Instead, we decided to take posts using hashtags and conduct sentiment analysis.

conversation_id	child_tweet_id	tweet_text
1465050134274785282	1465052303006830594	@tim_cook can you help me with money? 🙏😔
1465050134274785282	1465050583937605633	@tim_cook Please sir please give me only 4 million dollars please I request you 😭😭😭
1465050134274785282	1465050548789387267	@tim_cook I am gay gay life is too much difficult in india Plus Poverty my last hope please I request you 😭😭😭
1465050134274785282	1465050448465829893	@tim_cook Please sir please give me only 4 million dollars please I request you 😭😭😭 my father work in construction line and his age 60 years old and I am just 19 year old please I request you 😭😭😭

conversation_id	child_tweet_id	tweet_text
1465050134274785282	1465056353391628293	@tim_cook It's so heart warming to see you reach out to other diversities, as little as it may seems (just a tweet) it makes such a big happiness to so many who appreciate your words. Thank you!
1463236596505841665	1463282956659793927	@tim_cook Loving my MacBook Pro 14". It amazes me each time I use it. Love Apple. Happy thanksgiving sir
1463236596505841665	1463236916527214601	@tim_cook @Apple you deserve a medal for the democratisation of performance, energy efficiency and opportunity
1463236596505841665	1463236683470626822	@tim_cook Fantastic as always brother 💗🏆

Even though we were not able to acquire all replies as we wished. Looking at several replies from the CEO's posts, we soon realized that even if we had obtained the entire reply sections, we would still have a very noisy dataset that will not be useful to our evaluation. Most of the replies were not helpful. There were only a few replies that either criticized or praised the company. Most replies were useless and foolish asking for free iphones or promoting their own Twitter accounts or other opinions irrelevant to the CEO's posts.

Twitter Sentiment Analysis & Stock Trend for One Week (2021.12.06 - 2021.12.14)

We aim to analyze Twitter sentiments toward companies. Since the shareholders normally post their opinions under the hashtag of the target company's ticker symbol (for example, #AAPL for Apple, #TSLA for Tesla), we collected the hashtaged tweets and assigned a sentiment polarity score to each text. Due to the restrictions of Twitter's API (and unfortunately, the GetoldTweets3 library is no longer working since Twitter's update last year), we are only able to retrieve tweets from the past 10 days by using the **rtweet** package (https://github.com/ropensci/rtweet).

The sentiment polarity score is computed with the **sentimentr** package (https://github.com/trinker/sentimentr). We first cleaned the text by removing attached pictures and links. **sentimentr** is an augmented dictionary lookup algorithm that takes each word in the text, searches through multiple dictionaries of words with pre-labeled sentiment scores, takes valence shifters (i.e., negators, intensifiers, de-amplifiers, and adversative conjunctions) into account, and returns a score for which the positive indicates positive sentiment, the negative indicates negative, and 0 as purely neutral. It can also convert emojis and slangs to comprehensive descriptions.

Then, we retrieved stock data through Alpha Vantage and the package **alphavantager** (https://cran.r-project.org/web/packages/alphavantager/). We want to juxtapose the shifts in stock price and Twitter sentiment to see if the sentiment of shareholders online reflect/impact companies' stock price.

Codes

A function for text cleaning:

```
# pre-processing text:
clean.text = function(x)
{
  # remove urls
  x = qdapRegex::rm_twitter_url(x)
  # remove tabs
  x = gsub("[ |\t]{2,}", "", x)
  # remove blank spaces at the beginning
  x = gsub("^ ", "", x)
  # remove blank spaces at the end
  x = gsub(" $", "", x)
  # some other cleaning text
  x = gsub("([,])|[[:punct:]]", " ", x)
  x = gsub('[^[:graph:]]', ' ',x)
  #emojis and internet expressions to words
  x = replace_emoji(x)
  x = replace_word_elongation(x)
  x = replace_internet_slang(x)
  x = replace_emoticon(x)
}
```

Now we retrieve the most recent tweets under the hashtag #AAPL. We set the number of retrieving calls to a large number that exceeds the tweet volume under the same hashtag for the past ten days.

```
AAPL_tweet <- search_tweets("#AAPL",  n = 5000, type = "recent",
  include_rts = TRUE, lang = "en") %>% arrange(created_at)
```

The timestamps of each tweet is marked in UTC and specified to seconds. In order to juxtapose tweet sentiment with stock prices, we round up tweet timestamps to the nearest 5 minute and convert them to EST, the timezone corresponding to the stock market. Afterwards, we calculate the sentiment polarity score of each tweet.

```
AAPL_sentiment <- AAPL_tweet %>%
  mutate(est = round_time(created_at, "5 min",tz = "America/New_York"))%>%
  select(est, created_at, text)%>%
  mutate(clean = clean.text(text)) %>%
  mutate(pol = sentiment(clean)) %>%
  tidyr::unnest(cols = c(pol))%>%
  select(-element_id,-sentence_id, -word_count)%>%
  arrange(est)
```

```
## Warning: Each time ‘sentiment‘ is run it has to do sentence boundary disambiguation when a
## raw ‘character‘ vector is passed to ‘text.var‘. This may be costly of time and
## memory.  It is highly recommended that the user first runs the raw ‘character‘
## vector through the ‘get_sentences‘ function.
```

Since there could be multiple tweets sent within the same 5-min slot, we calculate the average of the sentiment polarity score for every 5 minute. Since Alpha Vantage can get stock value data for a longer time period (a month), we save the earliest time when tweets could be retrieved to be the start point of our future plots.

```r
AAPL_pol <- AAPL_sentiment %>%
  group_by(est) %>% summarise(polarity = mean(sentiment))

AAPL_start <- AAPL_pol[1,1] %>% pull()
print(AAPL_start)
```

```
## [1] "2021-12-06 08:15:00 EST"
```

Use Alpha Vantage and the package $**alphavantager**(“https://cran.r-project.org/web/packages/alphavantager/”) to get intraday stock value every 5 minutes. The data include the stock’s openning price, closing price, and transaction volume within each 5 minute interval. To make each 5-min-interval corresponds with one stock value, we calculate the average of opening and closing prices.

```r
AAPL <-
  av_get(symbol     = "AAPL",
       av_fun     = "TIME_SERIES_INTRADAY",
       interval   = "5min",
       outputsize = "full") %>%
  select(timestamp, open, close, volume) %>%
  mutate(avg = (open+close)/2) %>%
  mutate(hour = hour(timestamp)) %>%
  mutate(day = wday(timestamp)) #Sun=1, Mon=2, Tue=3, Wed=4, Thu=5, Fri=6, Sat=7
```

```
## Rows: 3719 Columns: 6
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (5): open, high, low, close, volume
## dttm (1): timestamp
```

```
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

(Some wrangling details) A problem we discovered in this data frame is that, due to the 5-min interval, daily data ends at 20:00 and begins at 4:05, so when plotting the stock price, the line between these two points won’t be horizontal. The code below manually adds rows for 4:00, for which the stock values remain the same as they were at 20:00 the previous day (e.g. Monday 4:00 should duplicate the values from Friday 20:00; Tuesday 4:00 should duplicate the values from Monday 20:00).

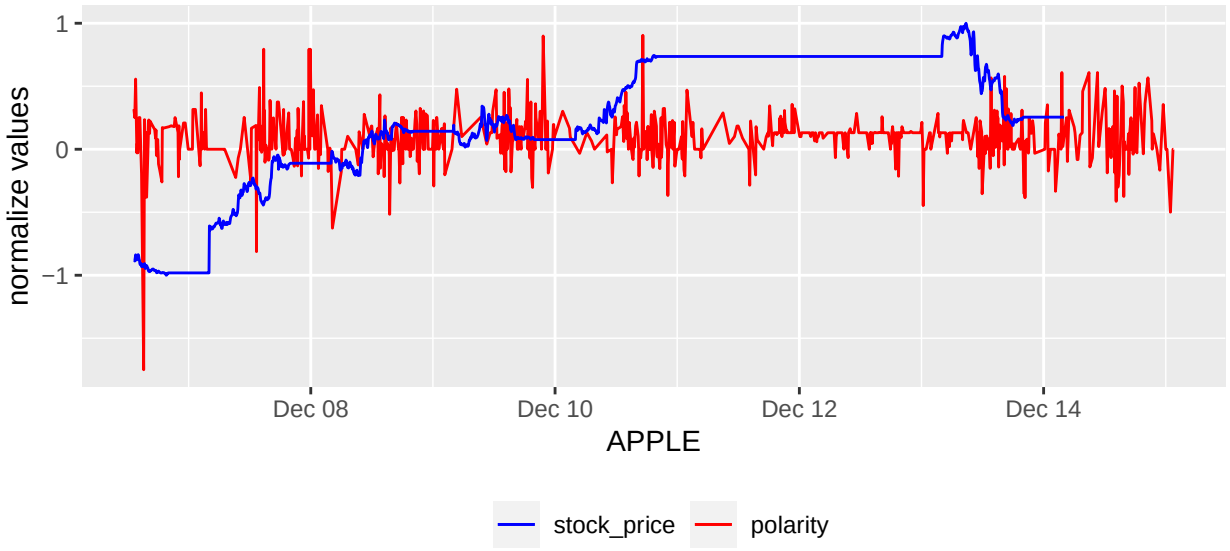
```r
#First create a separate dataframe for the stock prices at 20:00, and then change timestamp to correspo

AAPL_20 <- AAPL %>%
        filter(hour == 20) %>%
        mutate(hour = 4) %>%
        mutate(add_hours = if_else(day == 6, 56, 8)) %>% #if it's Friday 20:00, then add 56 hours to ch
        mutate(timestamp = timestamp + hours(add_hours)) %>%
        mutate(day = wday(timestamp)) %>%
        select(-add_hours)

#Finalize the dataframe for visualization. Since the sentiment polarity score and the stock value are o
```

```
AAPL_final <- rbind(AAPL,AAPL_20) %>%
        arrange(timestamp) %>% #arrange to inserted the rbind rows in chronological order.
        filter(timestamp > AAPL_start)%>%
        mutate(stock_price= 2*(avg - min(avg))/(max(avg)-min(avg)) - 1) %>%
        mutate(stock_volume = 2*(volume - min(volume))/(max(volume)-min(volume)) - 1)
```

The tweet dataframe and stock dataframe has the same timestamp column, so we fulljoin the two dataframes and plot the curves for stock changes and sentiment score changes.

Apple

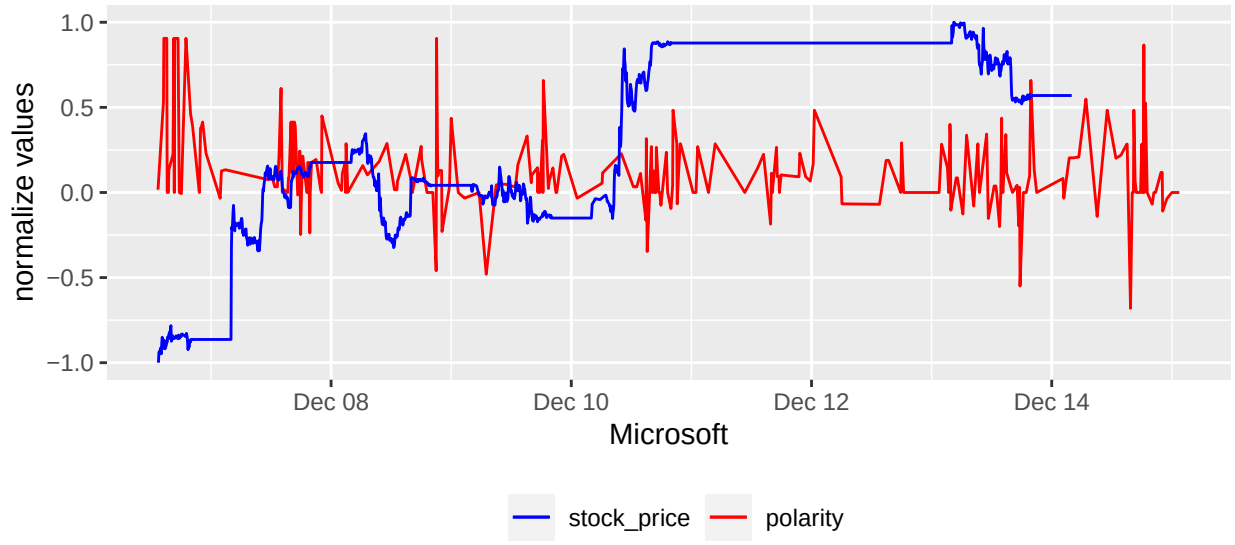
```
plot_AAPL <- AAPL_final %>%
  full_join(AAPL_pol, by = c( "timestamp" = "est" ))

colors <- c("stock_price" = "blue", "polarity" = "red")

ggplot()  +
  geom_line(data = plot_AAPL[!is.na(plot_AAPL$polarity),], aes(x = timestamp,y = polarity, color = 'pola
  #geom_line(data = plot_AAPL[!is.na(plot_AAPL$stock_volume),], aes(x = timestamp,y = stock_volume, col
  geom_line(data = plot_AAPL[!is.na(plot_AAPL$stock_price),], aes(x = timestamp,y = stock_price, color =
  labs(x = "APPLE",
         y = "normalize values",
         color = "") +
  scale_color_manual(values = colors)+
  theme(legend.position = "bottom")+
  theme(aspect.ratio=1/3)
```



Now, repeat the procedure for other companies.

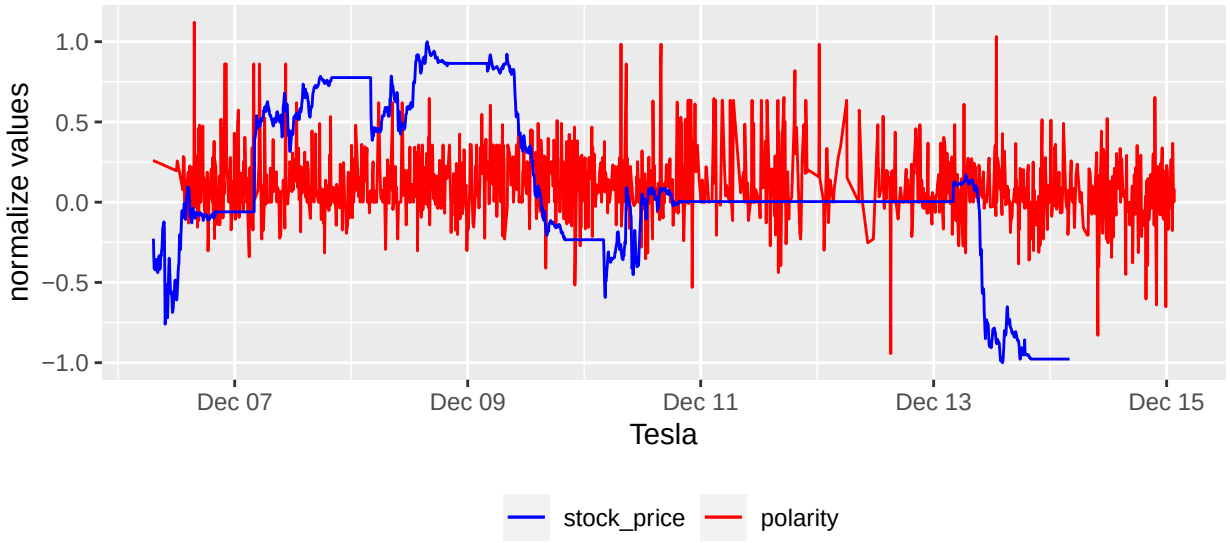
Microsoft:

```
ggplot()  +
  geom_line(data = plot_MSFT[!is.na(plot_MSFT$polarity),], aes(x = timestamp,y = polarity, color = 'pola
  geom_line(data = plot_MSFT[!is.na(plot_MSFT$stock_price),], aes(x = timestamp,y = stock_price, color =
  labs(x = "Microsoft",
         y = "normalize values",
         color = "") +
  scale_color_manual(values = colors)+
  theme(legend.position = "bottom")+
  theme(aspect.ratio=1/3)
```



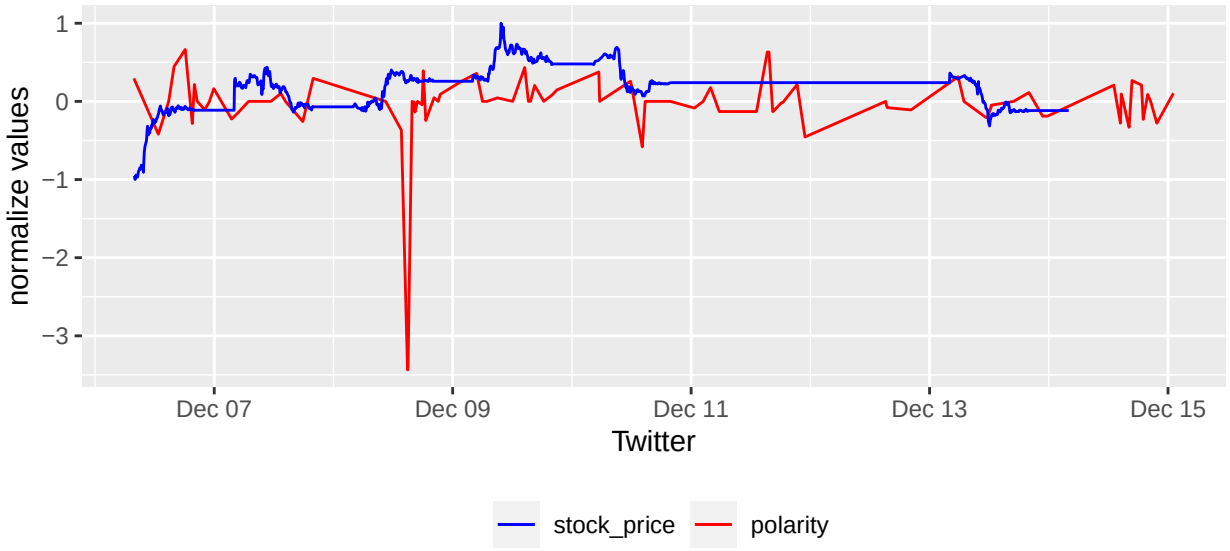
Tesla:

```
ggplot()  +
  geom_line(data = plot_TSLA[!is.na(plot_TSLA$polarity),], aes(x = timestamp,y = polarity, color = 'pola
  geom_line(data = plot_TSLA[!is.na(plot_TSLA$stock_price),], aes(x = timestamp,y = stock_price, color =
  labs(x = "Tesla",
         y = "normalize values",
         color = "") +
  scale_color_manual(values = colors)+
  theme(legend.position = "bottom")+
  theme(aspect.ratio=1/3)
```



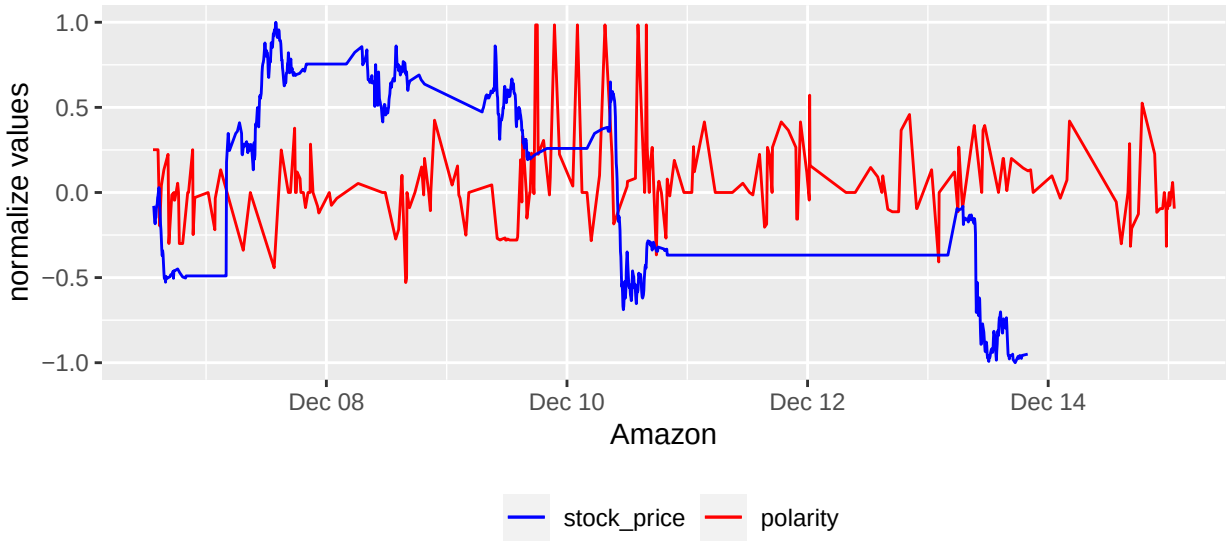
Twitter:

```
ggplot()  +
  geom_line(data = plot_TWTR[!is.na(plot_TWTR$polarity),], aes(x = timestamp,y = polarity, color = 'pola
  geom_line(data = plot_TWTR[!is.na(plot_TWTR$stock_price),], aes(x = timestamp,y = stock_price, color =
  labs(x = "Twitter",
         y = "normalize values",
         color = "") +
  scale_color_manual(values = colors)+
  theme(legend.position = "bottom")+
  theme(aspect.ratio=1/3)
```



Amazon:

```
ggplot()  +
  geom_line(data = plot_AMZN[!is.na(plot_AMZN$polarity),], aes(x = timestamp,y = polarity, color = 'pola
  geom_line(data = plot_AMZN[!is.na(plot_AMZN$stock_price),], aes(x = timestamp,y = stock_price, color =
  labs(x = "Amazon",
         y = "normalize values",
         color = "") +
  scale_color_manual(values = colors)+
  theme(legend.position = "bottom")+
  theme(aspect.ratio=1/3)
```



We plotted the sentiment score of tweet text under the hashtag of a company's stock ticker (e.g. #AAPL for Apple) and the stock price (normalized to range [-1,1] in order to match the sentiment score range) over time. The curve in blue is the stock price trend, and the curve in red is the shifts in Twitter sentiment. However, after making the juxtaposed plots for multiple companies, we don't see a matching pattern between the two scores.

Overall, the sentiment scores are noisy. There are multiple periods where sentiments are positive yet the stock price is falling, as well as times when sentiments are negative but the stock price is rising. Moreover, since the stock market doesn't operate during nights and weekends(but tweets are sent from all over the world all times), there are limited observations of stock trends for us to discern a possible correlation to the sentiment.

Twitter Sentiment Analysis & Stock Trend for Three Years (2016.01.01 ~ 2019.08.31)

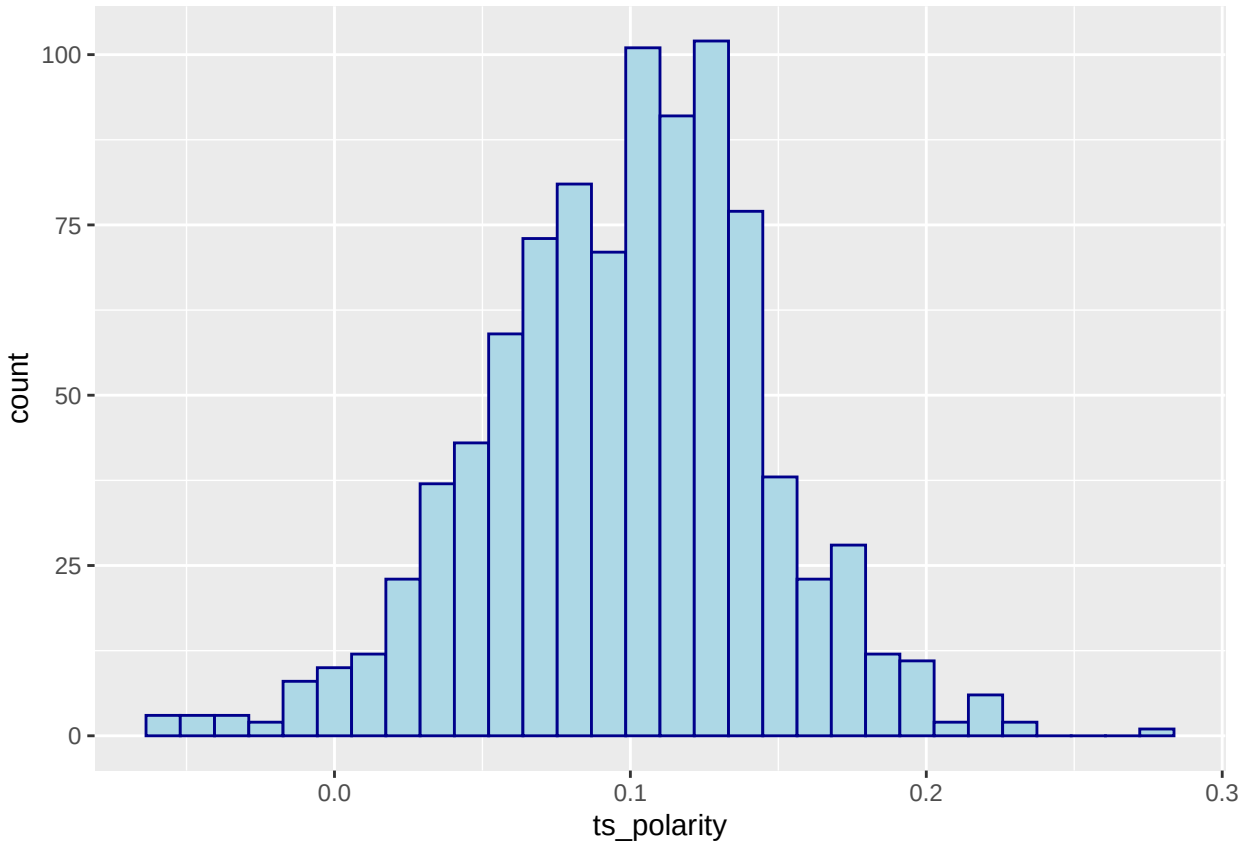
Previously, we were only averaging the sentiment scores for every five minute interval which means we are only capturing sentiments of a few limited people who happened to tweet at that time. It will be ideal if we can calculate the daily sentiment and compare it to a longer period of daily stock values. We happen to find a data set on Kaggle which includes Twitter sentiment and APPL stock prices for weekdays from January 1st, 2016 to August 31st, 2019. We wanted to use this dataset to predict the rise or the fall or stock prices. We understand that stock value prices are complex and have many components to it. Therefore, we wanted to see if the twitter sentiments are correlated to the stock trend. We ran classification models where the outcome variable is the trend of the stock market, either rising or falling.

```r
# Importing Kaggle Dataset for AAPL
df <- read.csv(file = 'AAPL.csv')
df$ts_polarity = as.numeric(as.character(df$ts_polarity))

# Calculating the stock price difference days
df <- df %>% add_column('Stock_difference' = 0)
for (i in 2:nrow(df)) {
  df$Stock_difference[i] <- df[i, 6] - df[i-1, 6]
}

# Categorizing by "rise" and "fall"
df <- df %>%
  mutate(trend = case_when(Stock_difference > 0 ~ "rise", Stock_difference <= 0 ~ "fall"))
```

Taking an initial look at the polarity score, the mean is 0.1 and we have a very few negative scores. This is interesting because according to this histogram, it seems like people tend to express more positive sentiments when tweeting about APPL.

```r
# Change line color and fill color
ggplot(df, aes(x=ts_polarity))+
  geom_histogram(color="darkblue", fill="lightblue")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We will now run several classification models: CART, Random Forest, and SVM

CLASSIFICATION CART

We first split the data a training set and test set.

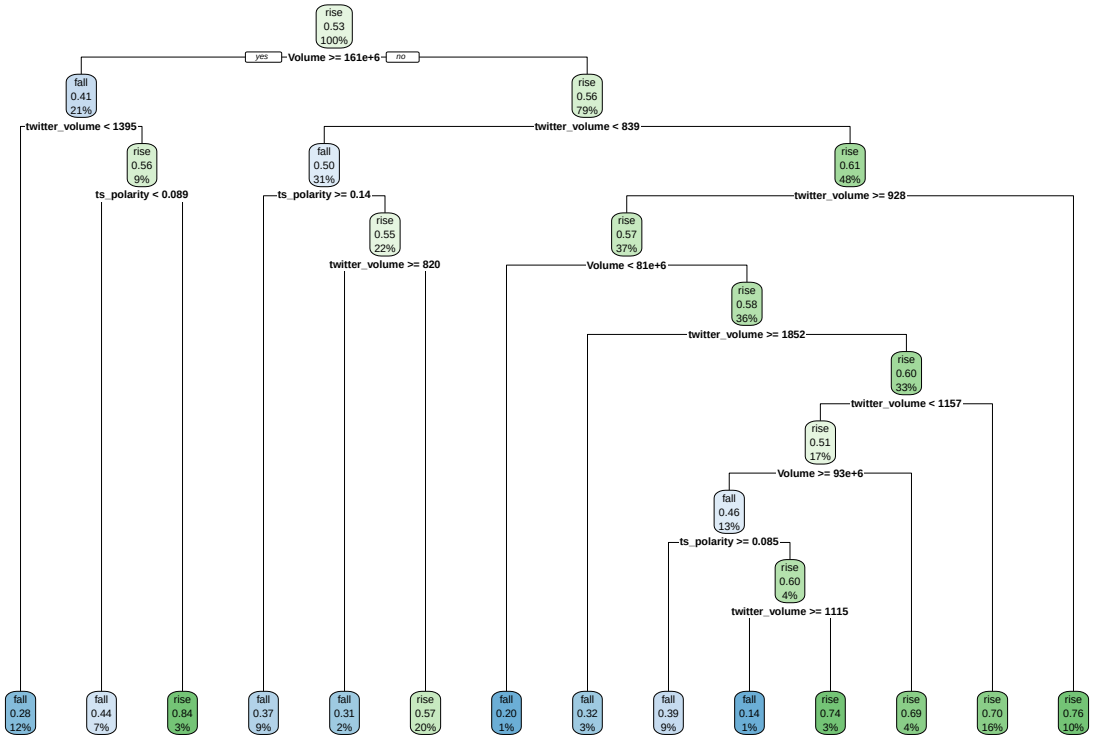
```
set.seed(47)

df_split <- initial_split(df, prop = 3/4)
df_train <- training(df_split)
df_test <- testing(df_split)
```

Next, we build our recipe, model, workflow, and fit it to the training data.

```
# recipe
cart_recipe <- recipe(trend ~ ts_polarity + twitter_volume + Volume, data = df_train)

# model
cart_model <- decision_tree() %>%
  set_engine("rpart") %>%
  set_mode("classification")

# workflow
cart_wflow <- workflow() %>%
  add_model(cart_model) %>%
  add_recipe(cart_recipe)

# fit
cart_fit <- cart_wflow %>%
  fit(data = df_train)
```

Below is the image of the tree. The first split is on the volume of stocks traded, meaning that volume is the most important determinant of stock price trends, given the data that we have. This is an unpruned tree and tuning our model could help us improve our accuracy.

```
cart_plot <-
  cart_fit %>%
  extract_fit_parsnip()
rpart.plot(
  cart_plot$fit,
  roundint = FALSE)
```



CART TUNUNG

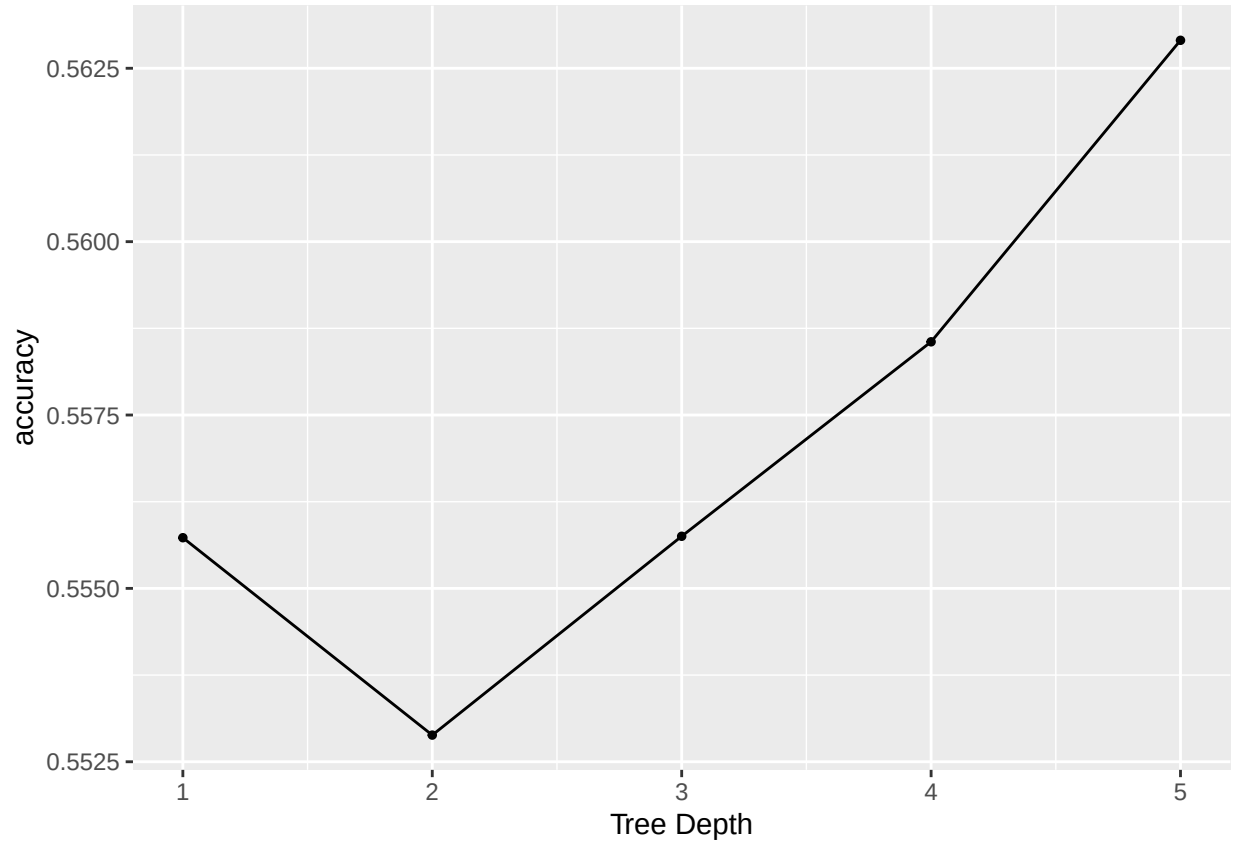
Given the size of our training data, we choose to do a 5-fold cross validation to tune our model.

```
set.seed(47)

# create folds
cart_vfold <- vfold_cv(df_train, v = 5, strata = trend)

# vector of tree depths
cart_grid <- expand.grid(tree_depth = seq(1, 5, by = 1))

# workflow
cart_tune <- decision_tree(tree_depth = tune()) %>%
  set_engine("rpart") %>%
  set_mode("classification")

cart_wflow_tune <- workflow() %>%
  add_model(cart_tune) %>%
  add_recipe(cart_recipe)

# tuning
cart_tuned <- cart_wflow_tune %>%
  tune_grid(resamples = cart_vfold,
```

```
            grid = cart_grid)

# best value of tree depth
cart_tuned %>% collect_metrics() %>%
  filter(.metric == "accuracy")
```

```
## # A tibble: 5 x 7
##   tree_depth .metric  .estimator  mean     n std_err .config
##        <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
## 1          1 accuracy binary     0.556     5 0.0112  Preprocessor1_Model1
## 2          2 accuracy binary     0.553     5 0.0101  Preprocessor1_Model2
## 3          3 accuracy binary     0.556     5 0.00503 Preprocessor1_Model3
## 4          4 accuracy binary     0.559     5 0.00563 Preprocessor1_Model4
## 5          5 accuracy binary     0.563     5 0.00703 Preprocessor1_Model5
```

```
cart_tuned %>%
autoplot(metric = "accuracy")
```




```
cart_tuned %>%
  collect_metrics() %>%
  filter(.metric == "accuracy") %>% arrange(desc(mean))
```

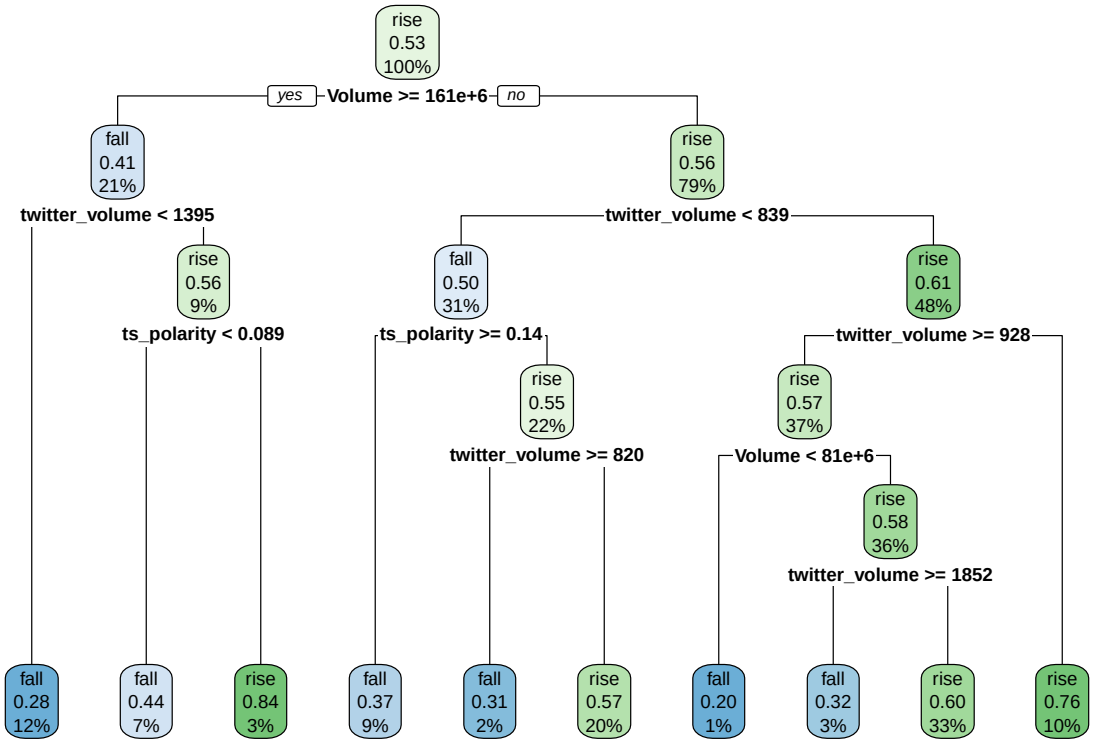
```
## # A tibble: 5 x 7
##   tree_depth .metric  .estimator  mean     n std_err .config
```

```
##         <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
## 1           5 accuracy binary     0.563     5 0.00703 Preprocessor1_Model5
## 2           4 accuracy binary     0.559     5 0.00563 Preprocessor1_Model4
## 3           3 accuracy binary     0.556     5 0.00503 Preprocessor1_Model3
## 4           1 accuracy binary     0.556     5 0.0112  Preprocessor1_Model1
## 5           2 accuracy binary     0.553     5 0.0101  Preprocessor1_Model2
```

After running the code above, we see that the tree depth 5 yields the best result. According to the cross validation, when our model uses the best parameters, we predict with 56.2% accuracy.

```
# pruned model
cart_final <- decision_tree(tree_depth = 5) %>%
set_engine("rpart") %>%
set_mode("classification")

# pruned model workflow
cart_final_wflow <- workflow() %>%
add_model(cart_final) %>%
add_recipe(cart_recipe)

# pruned fit
cart_final_fit <- cart_final_wflow %>%
fit(data = df_train)
```

```
final_cart_plot <-
  cart_final_fit %>%
  extract_fit_parsnip()
rpart.plot(
  final_cart_plot$fit,
  roundint = FALSE)
```



RANDOM FOREST MODEL

Now we turn to a random forests. From what we learned in class, trees suffer from high variance, so, to reduce variability, we implement a random forset.

First we create a recipe and mode for our random forest.

```
set.seed(47)

# recipe
rf_recipe <- recipe(trend ~ ts_polarity + twitter_volume + Volume, data = df_train)

# model
rf_model <- rand_forest() %>%
  set_engine("ranger", importance = "permutation") %>%
  set_mode("classification")
```

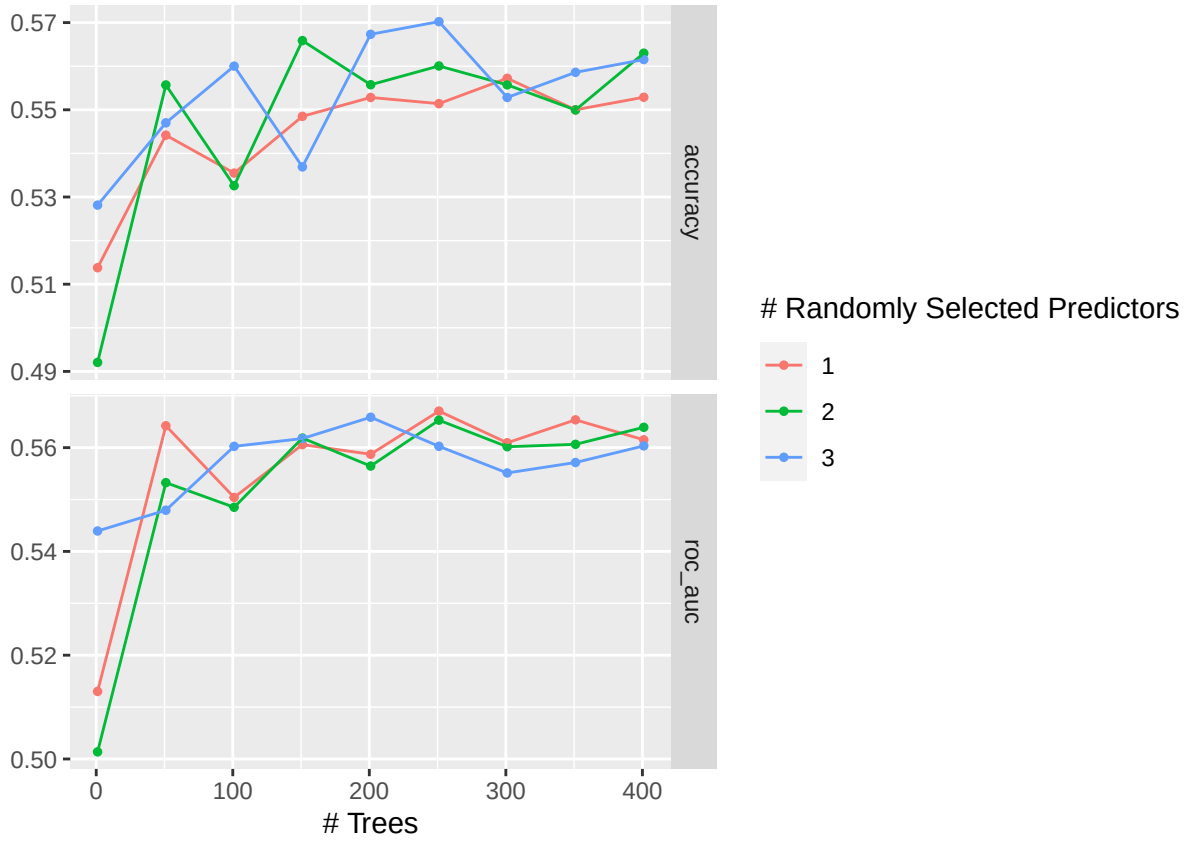
Now we tune the parameters to establish the best fit of our model. Again, we conduct a 5-fold cross validation and test the ideal number of trees between 1 and 401. According to the plot, the best random forest model when fitted to the training data would use and mtry of 3 and 251 trees.

```
set.seed(47)
# CV
rf_folds <- vfold_cv(df_train,  v = 5)
```

```r
# vector of tree depths and mtry
rf_grid <- expand.grid(mtry=seq(1,3), trees = seq(1,401, by =50))

# workflow
rf_tune <-
  rand_forest(trees = tune(), mtry = tune()) %>%
  set_engine("ranger", importance = "permutation") %>%
  set_mode("classification")

rf_wflow_tune <- workflow() %>%
  add_model(rf_tune) %>%
  add_recipe(rf_recipe)

# tuning
rf_tuned <- rf_wflow_tune %>%
tune_grid(resamples = rf_folds, grid = rf_grid)

# best accuracy
rf_tuned %>%
  select_best("accuracy")
```

```
## # A tibble: 1 x 3
##    mtry trees .config
##   <int> <dbl> <fct>
## 1     3   251 Preprocessor1_Model18
```

```r
# plot
rf_tuned %>%
  autoplot()
```



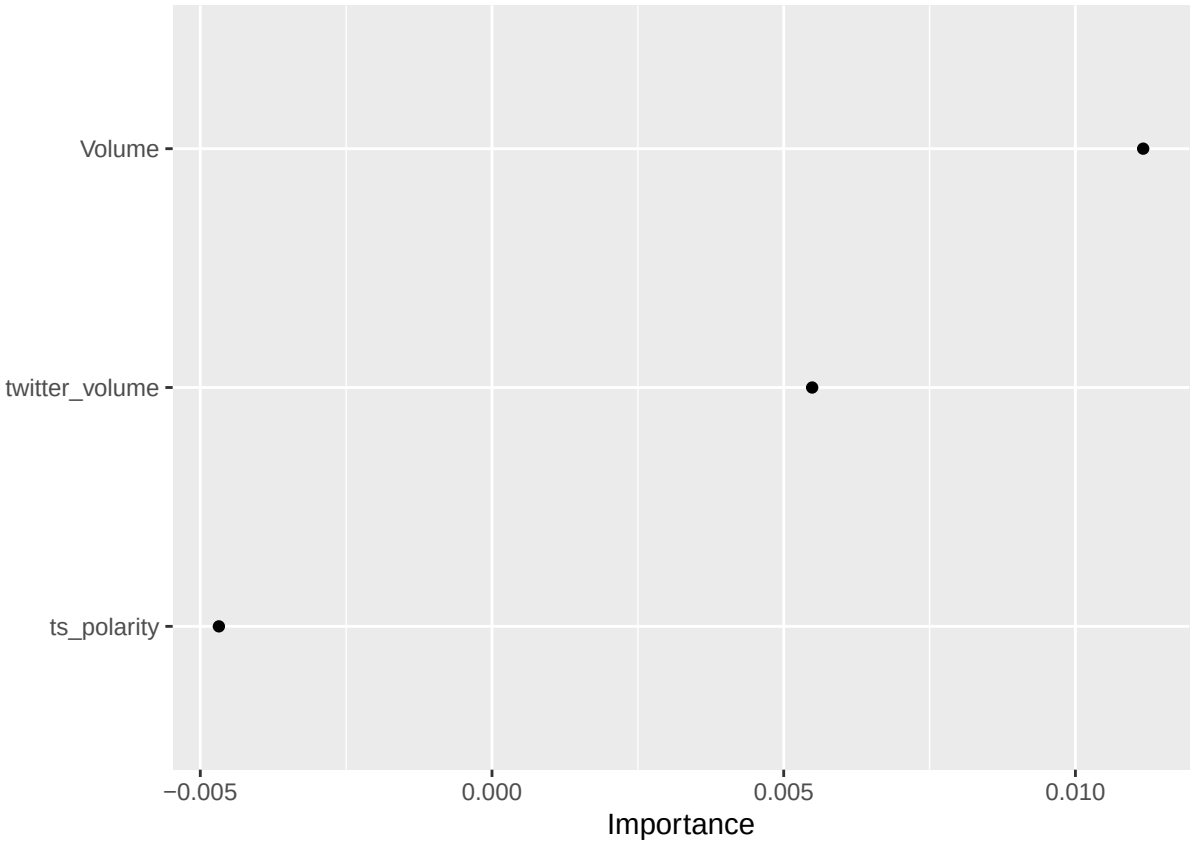

```
rf_tuned %>%
  collect_metrics() %>%
  filter(.metric == "accuracy") %>% arrange(desc(mean))
```

```
## # A tibble: 27 x 8
##     mtry trees .metric  .estimator  mean     n std_err .config
##    <int> <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
##  1     3   251 accuracy binary     0.570     5 0.0105  Preprocessor1_Model18
##  2     3   201 accuracy binary     0.567     5 0.00795 Preprocessor1_Model15
##  3     2   151 accuracy binary     0.566     5 0.00992 Preprocessor1_Model11
##  4     2   401 accuracy binary     0.563     5 0.0119  Preprocessor1_Model26
##  5     3   401 accuracy binary     0.562     5 0.00780 Preprocessor1_Model27
##  6     2   251 accuracy binary     0.560     5 0.0118  Preprocessor1_Model17
##  7     3   101 accuracy binary     0.560     5 0.00815 Preprocessor1_Model09
##  8     3   351 accuracy binary     0.559     5 0.00986 Preprocessor1_Model24
##  9     1   301 accuracy binary     0.557     5 0.0138  Preprocessor1_Model19
## 10     2   201 accuracy binary     0.556     5 0.0110  Preprocessor1_Model14
## # ... with 17 more rows
```

According to the cross validation, when our model uses the best parameters, we predict with 57% accuracy. Volume is the most important variable followed by twitter volume.

```
set.seed(47)

# Final Random Forest Model
```

```
rf_best <- rand_forest(mtry = 3, trees = 251) %>%
  set_engine("ranger", importance = "permutation") %>%
  set_mode("classification")

final_rf_model <-
  workflow() %>%
  add_model(rf_best) %>%
  add_recipe(rf_recipe) %>%
  fit(data = df_train)

# vip
final_rf_model %>% extract_fit_parsnip() %>%
  vip(geom = "point")
```



SVM

Lastly, we want to test if an SVM model would better predict stock price trends given twitter sentiment.

First we implement a linear SVM which we tune using 4-fold cross validation and tuning the cost to get the best model given our training data.

```
set.seed(47)

# recipe
```

```
svm_recipe <- recipe(trend ~ ts_polarity + twitter_volume + Volume, data = df_train) %>% step_normalize

# workflow
svm_lin <-
svm_linear(cost = tune()) %>%
set_engine("kernlab") %>%
set_mode("classification")

svm_lin_wflow <- workflow() %>%
add_model(svm_lin) %>%
add_recipe(svm_recipe)

# vfolds
folds <- vfold_cv(df_train, v=4)
cost_grid <- grid_regular(cost(), levels = 8)

# tune
svm_lin_tune <- svm_lin_wflow %>% tune_grid(resamples=folds, grid=cost_grid)

svm_lin_tune %>%
  collect_metrics() %>%
  filter(.metric == "accuracy") %>% arrange(desc(mean))
```

```
## # A tibble: 8 x 7
##        cost .metric  .estimator  mean     n std_err .config
##       <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
## 1  0.0190   accuracy binary     0.540     4  0.0187 Preprocessor1_Model3
## 2  0.000977 accuracy binary     0.531     4  0.0134 Preprocessor1_Model1
## 3  0.00431  accuracy binary     0.531     4  0.0134 Preprocessor1_Model2
## 4  0.0841   accuracy binary     0.524     4  0.0153 Preprocessor1_Model4
## 5  0.371    accuracy binary     0.521     4  0.0105 Preprocessor1_Model5
## 6  1.64     accuracy binary     0.521     4  0.0105 Preprocessor1_Model6
## 7  7.25     accuracy binary     0.520     4  0.0109 Preprocessor1_Model7
## 8 32        accuracy binary     0.520     4  0.0109 Preprocessor1_Model8
```

According to the cross validation, when our model uses the best parameters, we predict with 54% accuracy.

```
# best model
svm_lin_best <- finalize_model(svm_lin, select_best(svm_lin_tune, "accuracy"))

# fit
svm_lin_tuned_fit <-
  workflow() %>%
  add_model(svm_lin_best) %>%
  add_recipe(svm_recipe) %>%
  fit(data = df_train)
```

```
##  Setting default kernel parameters
```

We will test a polynomial SVM next which we also tune.

POLYNOMIAL SVM

```
set.seed(47)

svm_poly <- svm_poly(
cost = tune(),
degree = tune()) %>%
set_engine("kernlab") %>%
set_mode("classification")

# workflow
svm_poly_wflow <- workflow() %>%
add_model(svm_poly) %>%
add_recipe(svm_recipe)

# parameters
grid_poly <- grid_regular(cost(),
                          degree(c(1,5)), levels = 5)
# tune
svm_poly_tune <-
svm_poly_wflow %>%
tune_grid(resamples = folds,
grid = grid_poly)
```

```
## x Fold1: preprocessor 1/1, model 18/25 (predictions): Error: $ operator is invali...

## x Fold1: preprocessor 1/1, model 19/25 (predictions): Error: $ operator is invali...

## x Fold1: preprocessor 1/1, model 23/25 (predictions): Error: $ operator is invali...

## x Fold1: preprocessor 1/1, model 24/25 (predictions): Error: $ operator is invali...

## x Fold1: preprocessor 1/1, model 25/25 (predictions): Error: $ operator is invali...

## x Fold2: preprocessor 1/1, model 23/25 (predictions): Error: $ operator is invali...

## x Fold2: preprocessor 1/1, model 25/25 (predictions): Error: $ operator is invali...

## x Fold4: preprocessor 1/1, model 22/25 (predictions): Error: $ operator is invali...
```

```
svm_poly_tune %>%
collect_metrics() %>%
filter(.metric == "accuracy") %>%
arrange(desc(mean))
```

```
## # A tibble: 25 x 8
##         cost degree .metric  .estimator  mean     n std_err .config
##        <dbl>  <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
##  1  2.38          5 accuracy binary     0.550     3 0.0317  Preprocessor1_Model~
##  2  0.0131        2 accuracy binary     0.550     4 0.0184  Preprocessor1_Model~
```

```
##  3  0.000977      5 accuracy binary     0.549     4 0.0233  Preprocessor1_Model~
##  4  0.177         2 accuracy binary     0.549     4 0.00942 Preprocessor1_Model~
##  5  0.0131        5 accuracy binary     0.547     3 0.0268  Preprocessor1_Model~
##  6  0.0131        4 accuracy binary     0.547     4 0.0117  Preprocessor1_Model~
##  7  2.38          4 accuracy binary     0.546     3 0.0177  Preprocessor1_Model~
##  8  0.177         4 accuracy binary     0.544     3 0.0184  Preprocessor1_Model~
##  9  2.38          2 accuracy binary     0.544     4 0.0150  Preprocessor1_Model~
## 10 32             2 accuracy binary     0.544     4 0.0150  Preprocessor1_Model~
## # ... with 15 more rows
```

```
svm_poly_tune %>%
  collect_metrics() %>%
  filter(.metric == "accuracy") %>% arrange(desc(mean))
```

```
## # A tibble: 25 x 8
##         cost degree .metric  .estimator  mean     n std_err .config
##        <dbl>  <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
##  1  2.38          5 accuracy binary     0.550     3 0.0317  Preprocessor1_Model~
##  2  0.0131        2 accuracy binary     0.550     4 0.0184  Preprocessor1_Model~
##  3  0.000977      5 accuracy binary     0.549     4 0.0233  Preprocessor1_Model~
##  4  0.177         2 accuracy binary     0.549     4 0.00942 Preprocessor1_Model~
##  5  0.0131        5 accuracy binary     0.547     3 0.0268  Preprocessor1_Model~
##  6  0.0131        4 accuracy binary     0.547     4 0.0117  Preprocessor1_Model~
##  7  2.38          4 accuracy binary     0.546     3 0.0177  Preprocessor1_Model~
##  8  0.177         4 accuracy binary     0.544     3 0.0184  Preprocessor1_Model~
##  9  2.38          2 accuracy binary     0.544     4 0.0150  Preprocessor1_Model~
## 10 32             2 accuracy binary     0.544     4 0.0150  Preprocessor1_Model~
## # ... with 15 more rows
```

According to the cross validation, when our model uses the best parameters, we predict with 55% accuracy.

```
# best model
svm_poly_best <- finalize_model(
  svm_poly,
  select_best(svm_poly_tune, "accuracy"))
svm_poly_best
```

```
## Polynomial Support Vector Machine Specification (classification)
##
## Main Arguments:
##   cost = 2.37841423000544
##   degree = 5
##
## Computational engine: kernlab
```

```
# fit
svm_poly_tuned_fit <-
  workflow() %>%
  add_model(svm_poly_best) %>%
  add_recipe(svm_recipe) %>%
  fit(data = df_train)
```

RBF SVM

Lastly, we implement a RBF model which we also tune.

```
set.seed(47)

svm_rbf <- svm_rbf(
cost = tune(), rbf_sigma=tune()) %>%
set_engine("kernlab") %>%
set_mode("classification")

# workflow
svm_rbf_wflow <- workflow() %>%
add_model(svm_rbf) %>%
add_recipe(svm_recipe)

# parameters
grid_rbf <- grid_regular(cost(), rbf_sigma(), levels=5)

# tune
svm_rbf_tune <- svm_rbf_wflow %>% tune_grid(resamples = folds, grid = grid_rbf)

svm_rbf_tune %>%
collect_metrics() %>%
filter(.metric == "accuracy") %>%
arrange(desc(mean))
```

```
## # A tibble: 25 x 8
##         cost    rbf_sigma .metric  .estimator  mean     n std_err .config
##        <dbl>        <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
##  1  2.38     1            accuracy binary     0.560     4  0.0210 Preprocessor1~
##  2  0.177    1            accuracy binary     0.546     4  0.0224 Preprocessor1~
##  3  2.38     0.00316      accuracy binary     0.533     4  0.0148 Preprocessor1~
##  4  0.000977 0.0000000001 accuracy binary     0.531     4  0.0134 Preprocessor1~
##  5  0.0131   0.0000000001 accuracy binary     0.531     4  0.0134 Preprocessor1~
##  6  0.177    0.0000000001 accuracy binary     0.531     4  0.0134 Preprocessor1~
##  7  2.38     0.0000000001 accuracy binary     0.531     4  0.0134 Preprocessor1~
##  8 32        0.0000000001 accuracy binary     0.531     4  0.0134 Preprocessor1~
##  9  0.000977 0.0000000316 accuracy binary     0.531     4  0.0134 Preprocessor1~
## 10  0.0131   0.0000000316 accuracy binary     0.531     4  0.0134 Preprocessor1~
## # ... with 15 more rows
```

```
svm_rbf_best <- finalize_model(svm_rbf, select_best(svm_rbf_tune, "accuracy"))
svm_rbf_best
```

```
## Radial Basis Function Support Vector Machine Specification (classification)
##
## Main Arguments:
##   cost = 2.37841423000544
##   rbf_sigma = 1
##
## Computational engine: kernlab
```

```
# fit
svm_rbf_tuned_fit <-
  workflow() %>%
  add_model(svm_rbf_best) %>%
  add_recipe(svm_recipe) %>%
  fit(data = df_train)
```

According to the cross validation, when our model uses the best parameters, we predict with 56% accuracy. All of our models did not accurately predict the data. Overall, it seems like using Twitter sentiment to predict stock price trends is futile.

Test Accuracy

We now use our best model to predict the test data.

```
# predict on train data
final_rf_model %>%
predict(new_data = df_test) %>%
cbind(df_test) %>%
select(.pred_class, trend) %>%
  table()
```

```
##            trend
## .pred_class fall rise
##        fall   45   65
##        rise   63   58
```

```
accuracy <- (45+58)/(45+58+63+65)
accuracy
```

```
## [1] 0.4458874
```

The test accuracy was 44.6%.

Conclusion and Ethical Considerations

Prediction of stock price is an extremely complex and very challenging task because there are a lot more factors involved such as company scandals, macroeconomic policies, political events, and other factors (i.e. 2020 Covid-19 pandemic) which may impact the stock price. Due to these factors, it is difficult to find out the dependence of a single factor on future prices and trends.

In addition, we have to admit that our data is limited. Twitter API has limitations on the amount of data that can be accessed. It only allows us to retrieve the most recent tweets. If there are thousands of tweets with a certain hashtag every day, with our limited number of requests we can make, we are not able to get tweets from previous days and thus not use it with our stock data. Although we ended up using a Kaggle dataset with Twitter information regarding Apple stock. The Kaggle dataset also has limited data (Jan'16 - Aug'19) which restricted our analysis to limited number of trading days. In addition, because we got the pre-processed dataset from the Kaggle, we cannot access the original texts from Twitter, which means we do not know how exactly these polarity, or sentiment scores, were calculated.

From our project, we can conclude several findings.

(1) Generating wordclouds and analyzing comments obtained from each CEO's Twitte account, we were able to conclude that Twitter space may not be the best platform to capture an accurate depiction of the public perception toward a company. There are a lot more irrelevant comments that makes our data noisy. And the amount of replies were too small that we could not really parse signal from the noise. Even though Twitter may not be the accurate description of a company, we could see that they do potentiallly capture what each company is striving towards, since the wordclouds do reflect the most frequently used words on each CEO's Twitter space.

(2) With Twitter sentiment analysis and machine learning methods, we were able to conclude that, to our surprise, Twitter sentiment does not reflect the trends of stock prices. This could be because only limited people tweets about their stock purchase/sales under the stock ticker hashtags. People who tweet about these stocks are not the comprehensive representation of all shareholders.

For future models, we would consider conducting NLP on other text sources, such as newspapers and forums. Studies have shown there are successful models that predicts stock trends using text sources from news and forums, instead of social media (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7959635/). We do not want to downplay the power of NLP. We still do believe that public sentiment and how people feel about the market is an important factor of stock price trends. We may want to consider ways to capture a broader crowd of users that better represents a larger population.

The importance of numerous social media platforms in business field has risen drastically over the past years. Many social media platforms have abundant resource about the consumer activities and market trends. Twitter network can be a helpful tool to find the right target audience or to follow the latest marketing trends; however, it may just not be so adequate for predicting stock price trends.

Although our work is very limited, the takeaways are that there are powerful tools out there and that it's up to us to make the right decisions on which sources and tools to use in order to create more powerful predictive models.

Packages

Kearney, M. W. (2019). rtweet: Collecting and analyzing Twitter data, Journal of Open Source Software, 4, 42. 1829. doi:10.21105/joss.01829 (R package version 0.7.0)

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2021). dplyr: A Grammar of Data Manipulation. R package version 1.0.7. https://CRAN.R-project.org/package=dplyr

Silge J, Robinson D (2016). "tidytext: Text Mining and Analysis Using Tidy Data Principles in R."*JOSS*, *1*(3). doi: 10.21105/joss.00037 (URL: https://doi.org/10.21105/joss.00037), <URL:http://dx.doi.org/10.21105/joss.00037>.

Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL https://www.jstatsoft.org/v40/i03/.

Rinker, T. W. (2021). sentimentr: Calculate Text Polarity Sentiment version 2.9.0. https://github.com/trinker/sentimentr

Hadley Wickham (2019). stringr: Simple, Consistent Wrappers for Common String Operations. R package version 1.4.0. https://CRAN.R-project.org/package=stringr

Ingo Feinerer and Kurt Hornik (2020). tm: Text Mining Package. R package version 0.7-8. https://CRAN.R-project.org/package=tm

Ingo Feinerer, Kurt Hornik, and David Meyer (2008). Text Mining Infrastructure in R. Journal of Statistical Software 25(5): 1-54. URL: https://www.jstatsoft.org/v25/i05/.

Milan Bouchet-Valat (2020). SnowballC: Snowball Stemmers Based on the C 'libstemmer' UTF-8 Library. R package version 0.7.0. https://CRAN.R-project.org/package=SnowballC

Ian Fellows (2018). wordcloud: Word Clouds. R package version 2.6. https://CRAN.R-project.org/package=wordcloud

Erich Neuwirth (2014). RColorBrewer: ColorBrewer Palettes. R package version 1.1-2. https://CRAN.R-project.org/package=RColorBrewer

Dawei Lang and Guan-tin Chien (2018). wordcloud2: Create Word Cloud by 'htmlwidget'. R package version 0.2.1. https://CRAN.R-project.org/package=wordcloud2

Stephen Milborrow (2021). rpart.plot: Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'. R package version 3.1.0. https://CRAN.R-project.org/package=rpart.plot

Kuhn et al., (2020). Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles. https://www.tidymodels.org

Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

Jacob Kaplan (2020). fastDummies: Fast Creation of Dummy (Binary) Columns and Rows from Categorical Variables. R package version 1.6.3. https://CRAN.R-project.org/package=fastDummies

Brandon M. Greenwell and Bradley C. Boehmke (2020). Variable Importance Plots—An Introduction to the vip Package. The R Journal, 12(1), 343–366. URL https://doi.org/10.32614/RJ-2020-013.

Alexandros Karatzoglou, Alex Smola, Kurt Hornik, Achim Zeileis (2004). kernlab - An S4 Package for Kernel Methods in R. Journal of Statistical Software 11(9), 1-20. URL http://www.jstatsoft.org/v11/i09/

Matt Dancho and Davis Vaughan (2020). alphavantager: Lightweight R Interface to the Alpha Vantage API. R package version 0.1.2. https://CRAN.R-project.org/package=alphavantager

Bibliography

Effrosynidis, Dimitris. "How I Created a Real-Time Twitter Sentiment Analysis Tool for Covid." Medium, Towards Data Science, 16 Apr. 2020, https://towardsdatascience.com/how-i-created-a-real-time-twitter-sentiment-analysis-tool-for-covid-292ff6a6323b.

Kolasani, Sai Vikram, and Rida Assaf. "Predicting Stock Movement Using Sentiment Analysis of Twitter Feed with Neural Networks." Journal of Data Analysis and Information Processing, vol. 08, no. 04, 2020, pp. 309–319., https://doi.org/10.4236/jdaip.2020.84018.

Robinson, Julia Silge and David. "2 Sentiment Analysis with Tidy Data: Text Mining with R." 2 Sentiment analysis with tidy data | Text Mining with R, September 2, 2021. https://www.tidytextmining.com/sentiment.html.

Rul, Céline Van den. "How to Generate Word Clouds in R." Medium, Towards Data Science, 20 Oct. 2019, https://towardsdatascience.com/create-a-word-cloud-with-r-bde3e7422e8a.

Kaggle Dataset

Sirimevan, N. (2019). "Twitter Sentiments AAPL stock". Retrieved December 10, 2021 from https://www.kaggle.com/nadun94/twitter-sentiments-aapl-stock