Predicting the Direction of the Stock Market using Logistic Regression, LDA, and KNN

Introduction

The purpose of this project is to compare the accuracy of three different classification approaches in predicting the direction of the stock market on a particular day:

1. Logistic regression
2. Linear Discriminant Analysis (LDA)
3. K-Nearest Neighbors (KNN)

For this project, I have selected two publicly traded social networking companies to analyze their price series, Twitter (TWTR) and Facebook (FB). For both of these stock-price series, I will be using data from Yahoo Finance of adjusted daily closing prices over the period from November 7th, 2013 (when TWTR went public) until January 1st, 2021. So, there are about 1,800 observations in each series. I am using the `quantmod` package in `R` to download this pricing data.

```r
install.packages("quantmod", repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/_g/f0wltjmx68x38dqpldhxy8b00000gp/T//RtmpGgQTjK/downloaded_packages
```

```r
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

Visualizing the Data

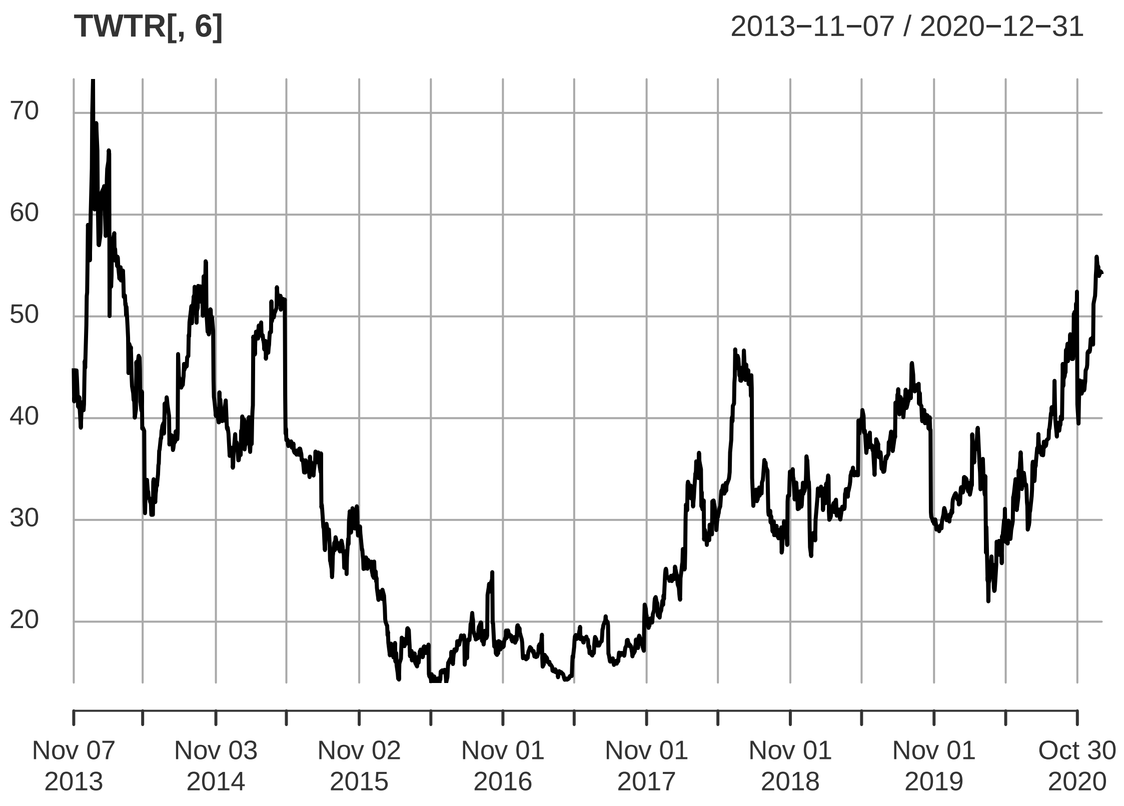
Here are plots of the adjusted daily closing prices for each series:

```r
getSymbols("TWTR", src = "yahoo", from = as.Date("2013-11-07"), to = as.Date("2021-01-01"))
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## [1] "TWTR"
```

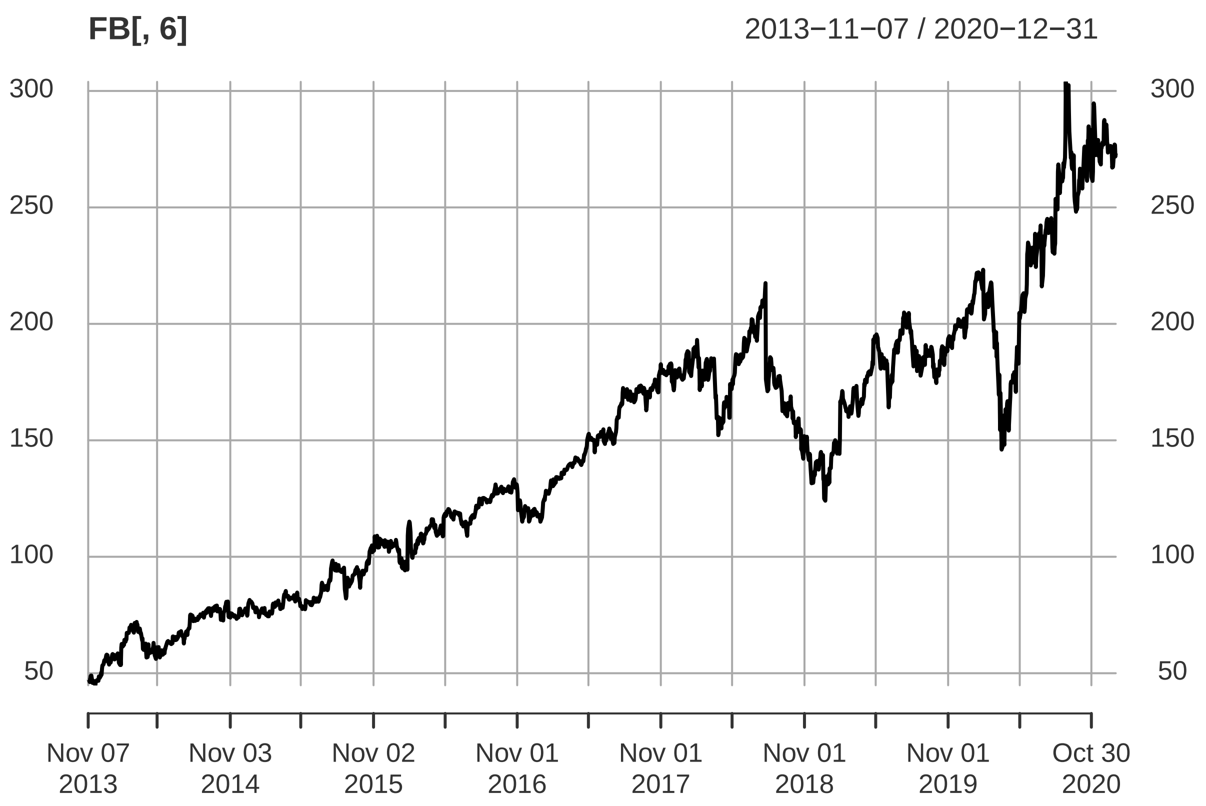
```r
plot(TWTR[,6])
```



```r
getSymbols("FB", src = "yahoo", from = as.Date("2013-11-07"), to = as.Date("2021-01-01"))
```

```
## [1] "FB"
```

```
plot(FB[,6])
```



These series both look pretty volatile. Furthermore, notice that neither the average or the volatility appears to be constant over time in either plot above. This is an indication that the series are not stationary.

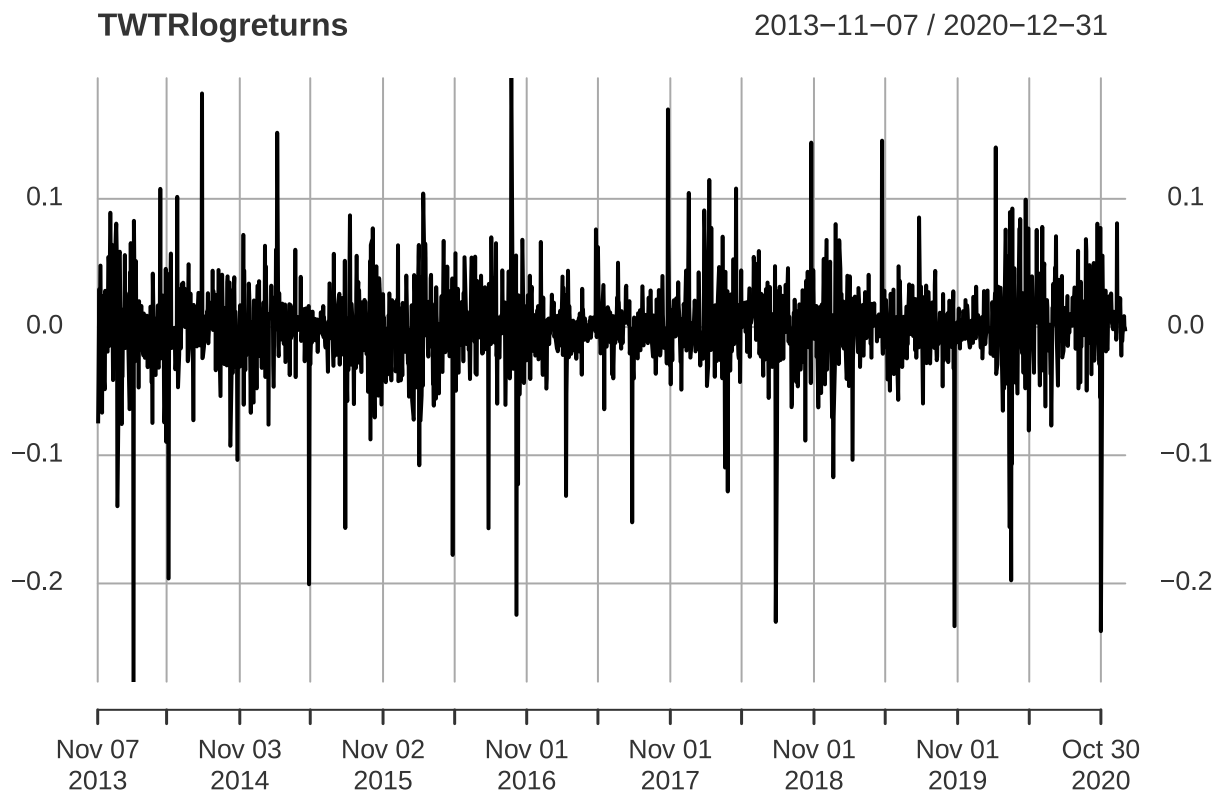
In the first plot of TWTR's daily adjusted closing prices, notice that the average is much lower between 2015 and 2017 than before or after that period. Moreover, the average of FB's prices actually appears to steadily increase over time. Therefore, the average is not constant over time in either series.

In addition to the averages, the volatilities of both series also appear to fluctuate over time. In the first plot, TWTR's daily adjusted closing prices seem to be the most volatile between 2013 and mid-2014, and the least volatile between 2015 and 2017. In the second plot, the volatility appears to steadily increase over time, especially after 2018.

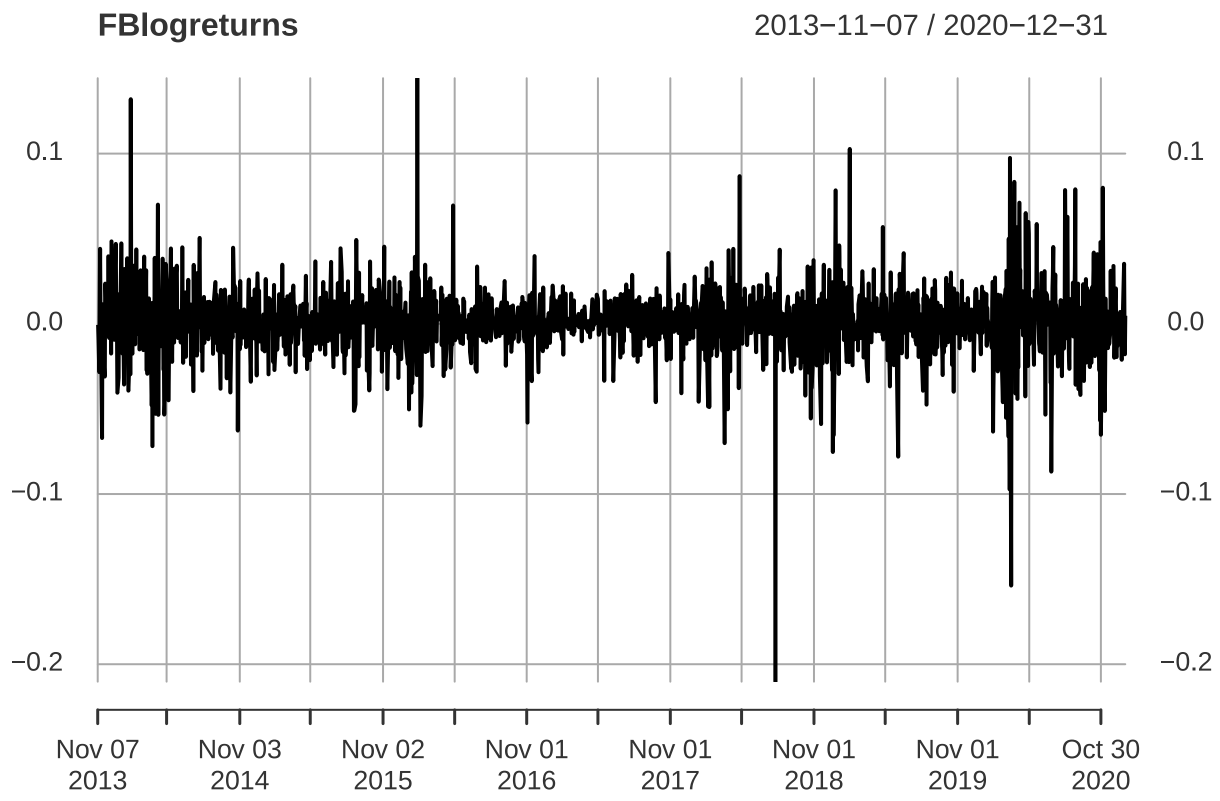
Based on the plots above, these series are probably nonstationary. So, it may be a good idea to use log-transformed prices for our analysis instead.

Let's take a look at the daily log-returns next. Here are the new plots:

```
TWTRlogreturns <- diff(log(TWTR[,6]))
plot(TWTRlogreturns)
```



```
FBlogreturns <- diff(log(FB[,6]))
plot(FBlogreturns)
```



These log-return series appear to be much more stationary than the previous two plots of ordinary adjusted daily returns. The average price seems to be constant over time, and the volatility does not change much over time either.

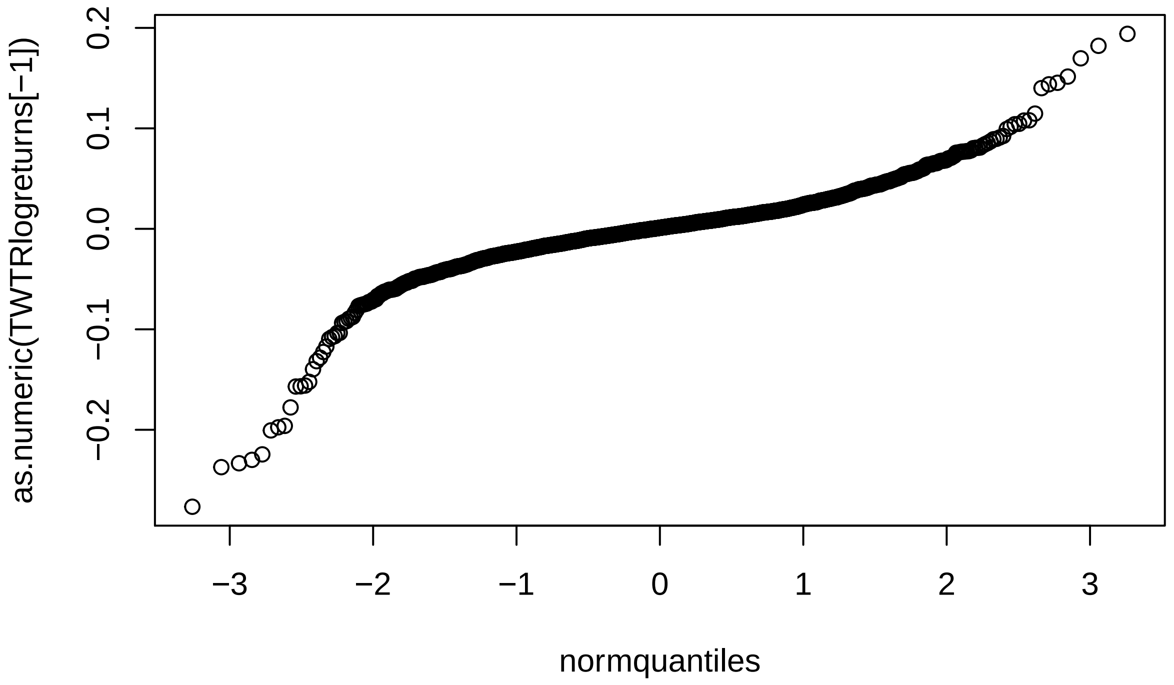
Since these log-returns are much less volatile and more stationary than the ordinary daily returns, I am going to use them for this analysis. Transforming your data is not always necessary, but since we are using the K-Nearest Neighbors approach here, it is important for our variables to be on a comparable scale.

Now, let's look at the distribution of our data. Linear Discriminant Analysis assumes that the data is drawn from a multivariate Gaussian distribution, and typically outperforms Logistic Regression when this assumption approximately holds. So, examining the distributions of the data set before performing performing our analysis can help us to better understand the results and why one approach may dominate the others.

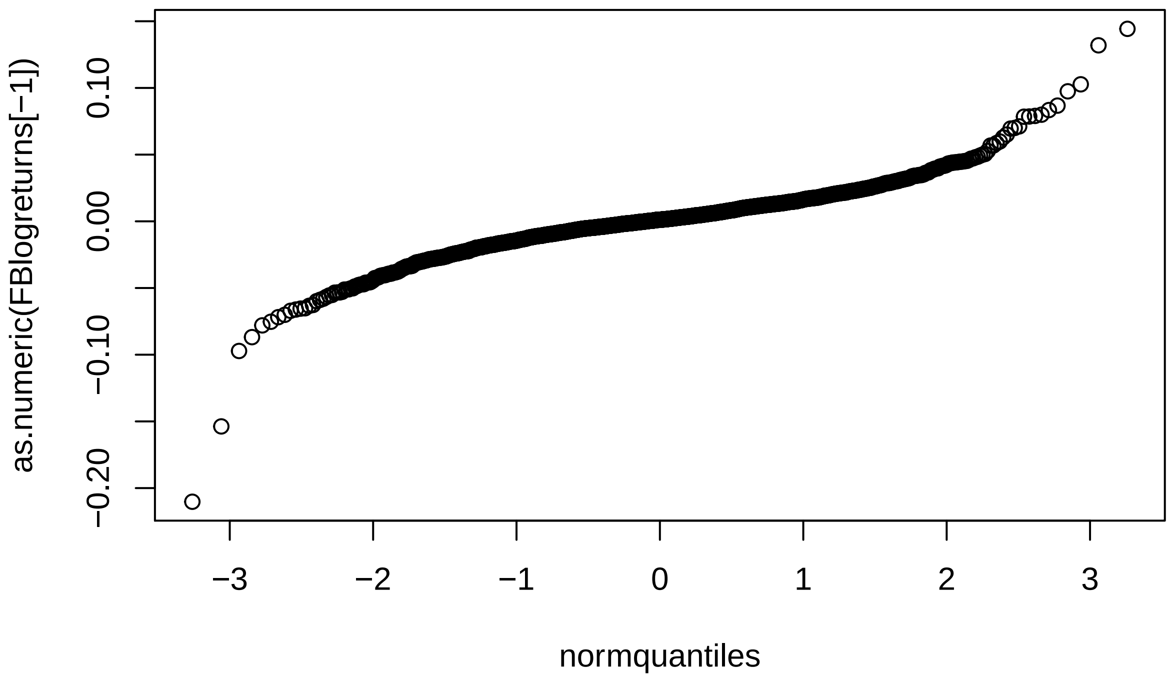
In order to determine whether the normal distribution is a reasonable model for a set of data, we can create normal probability plots. I am going to use `R`'s `qqplot` function here so that I don't have to sort the data before plotting.

Here are the quantile-quantile plots for Twitter and Facebook's log-return series:

```r
normquantiles <- qnorm(1:1800/1801)
qqplot(normquantiles,as.numeric(TWTRlogreturns[-1]))
```



```
qqplot(normquantiles,as.numeric(FBlogreturns[-1]))
```



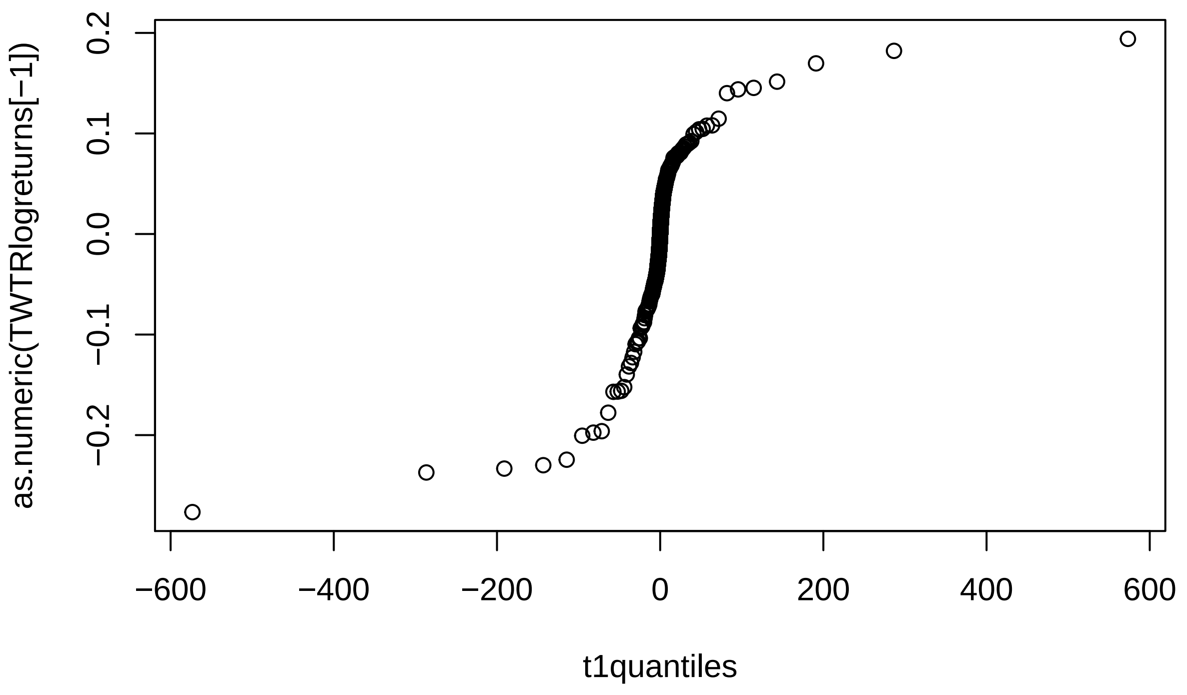
Neither of the plots above look exactly linear, but the normal plot for Facebook's log returns is a bit straighter through the bulk of the data than Twitter's. Additionally, while both of these plots appear to have some deviations from straightness in the extreme left and right ends, their shapes are different. Skewness in both tails of the quantile-quantile plots suggests that the log-return distributions of both stocks have longer tails than the normal distribution. In the first plot of TWTR's log returns, the left tail actually looks more skewed and has more points deviating from the straight line than in the right tail. This implies that extreme negative returns are probably more likely than extreme positive returns for TWTR.

Based on these quantile-quantile plots, we do not have evidence that the normal distribution is a very good fit for either log-return series. But, it looks like the normal distribution may be a slightly better fit for FB's log return data than TWTR's. Therefore, I am predicting that the LDA method may work a little better for Facebook's data than Twitter's.

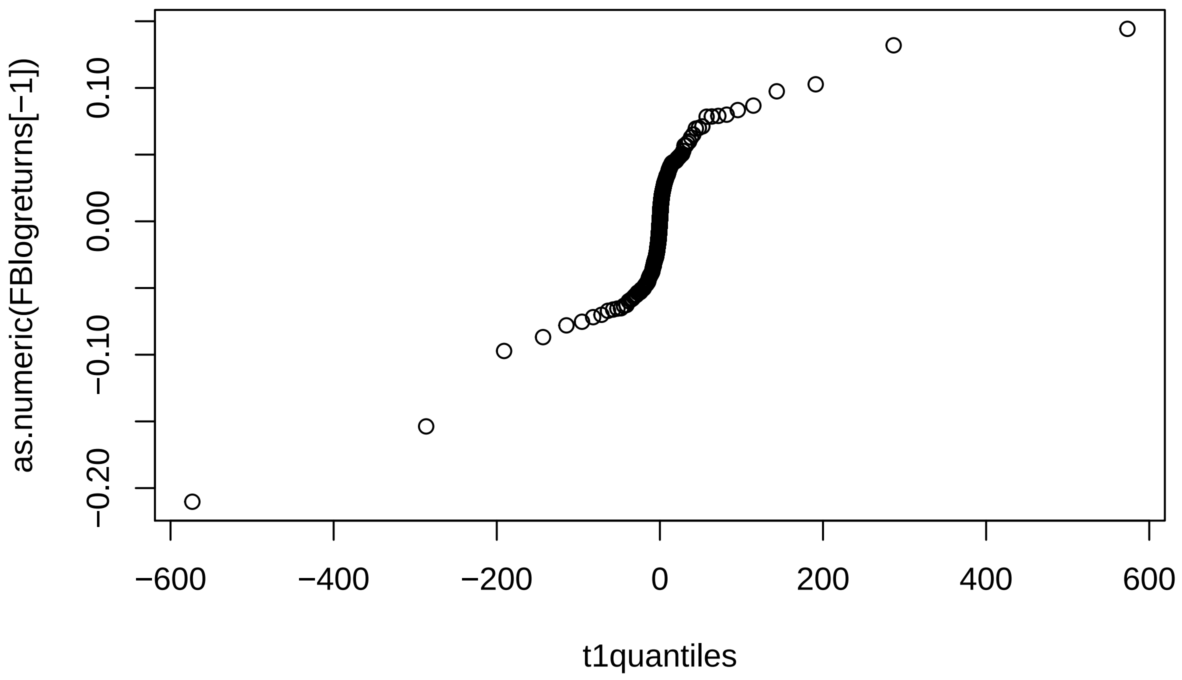
Since this data does not look exactly normal, let's see if the Student-t distribution is a better fit. Student-t probability plots can be produced in a similar manner as the normal plots above. I am going to use the `qqplot` function again to create t-plots for the log-return series with 1, 4, 6, and 20 degrees of freedom.

Here are the quantile-quantile plots for the log-return series with 1 degree of freedom:

```r
t1quantiles <- qt(1:1800/1801,df=1)
qqplot(t1quantiles,as.numeric(TWTRlogreturns[-1]))
```



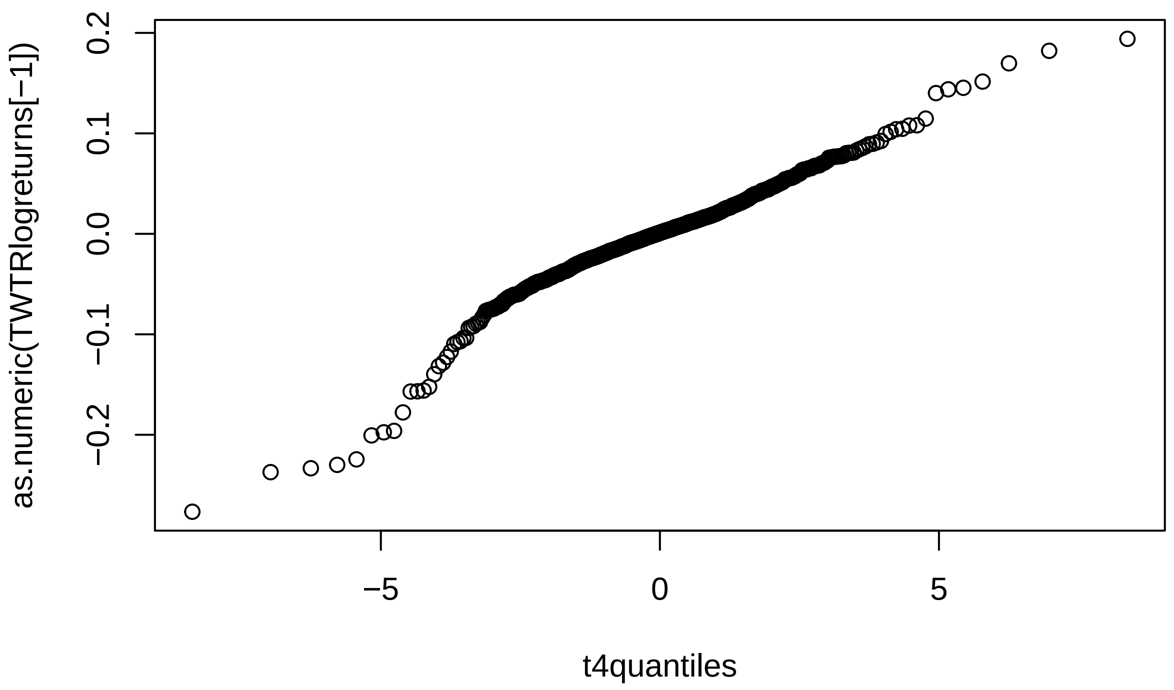
```r
qqplot(t1quantiles,as.numeric(FBlogreturns[-1]))
```



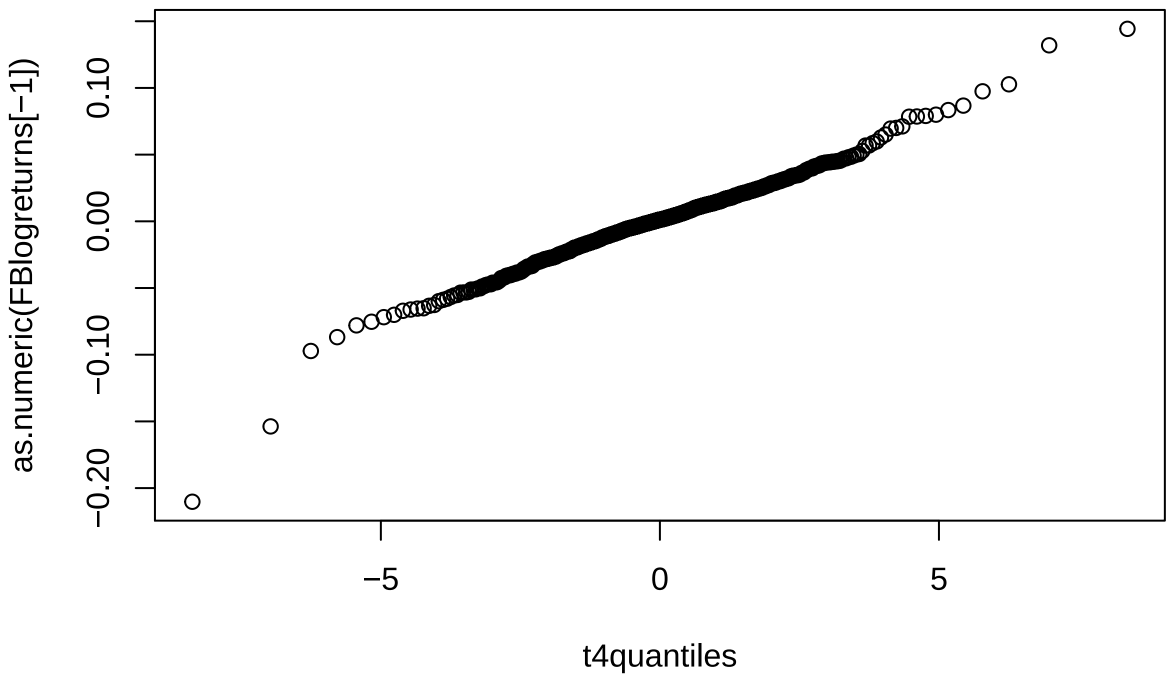
In both of these t-plots, there is an even smaller portion of the points that fall in a straight line than in the previous two normal plots. Moreover, the tails also appear much more skewed here in the t-plots than in the normal plots. So, we do not have evidence that the Student-t distribution with 1 degree of freedom is a good fit for either series, and it actually may be a worse fit for the series than the normal distribution.

Now, consider the t-plots for the series with 4 degrees of freedom:

```
t4quantiles <- qt(1:1800/1801,df=4)
qqplot(t4quantiles,as.numeric(TWTRlogreturns[-1]))
```



```r
qqplot(t4quantiles,as.numeric(FBlogreturns[-1]))
```

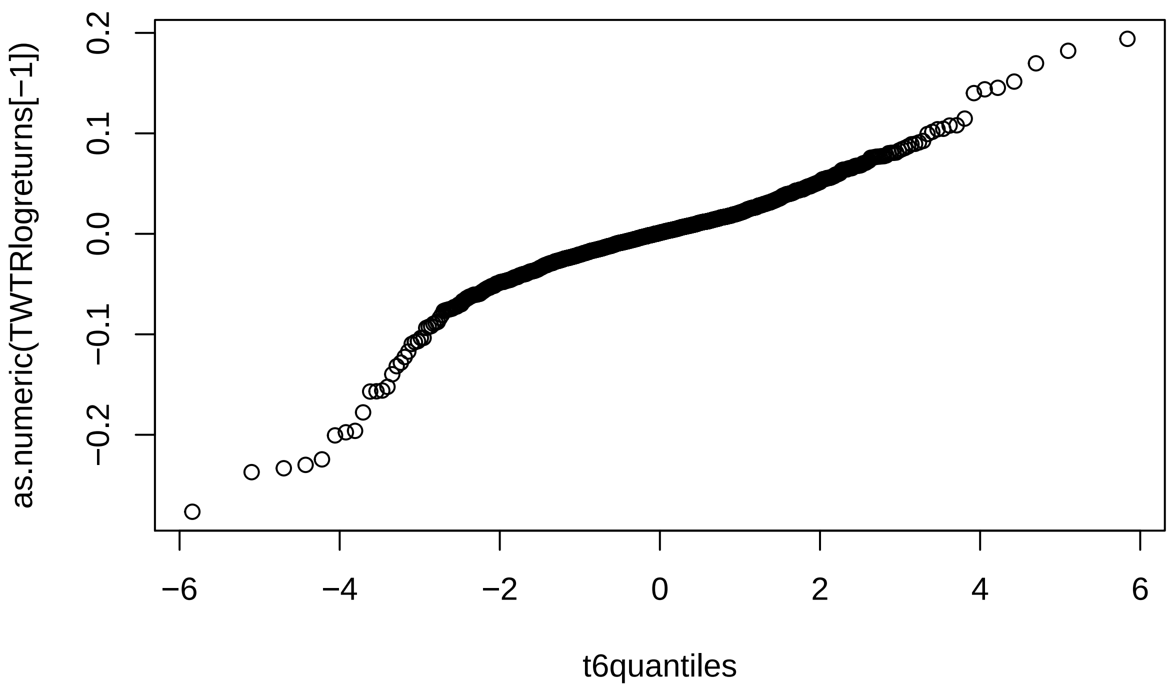


In both plots above with 4 degrees of freedom, the points look much straighter than in the previous ones, so this distribution is definitely a better fit for the series than the normal distribution or the Student-t distribution with 1 degree of freedom. But, also notice that the line is much straighter in the second plot than the first. Therefore, I believe this model is slightly better for FB's data than TWTR's, but we do have evidence that it is a pretty good fit for both stocks.

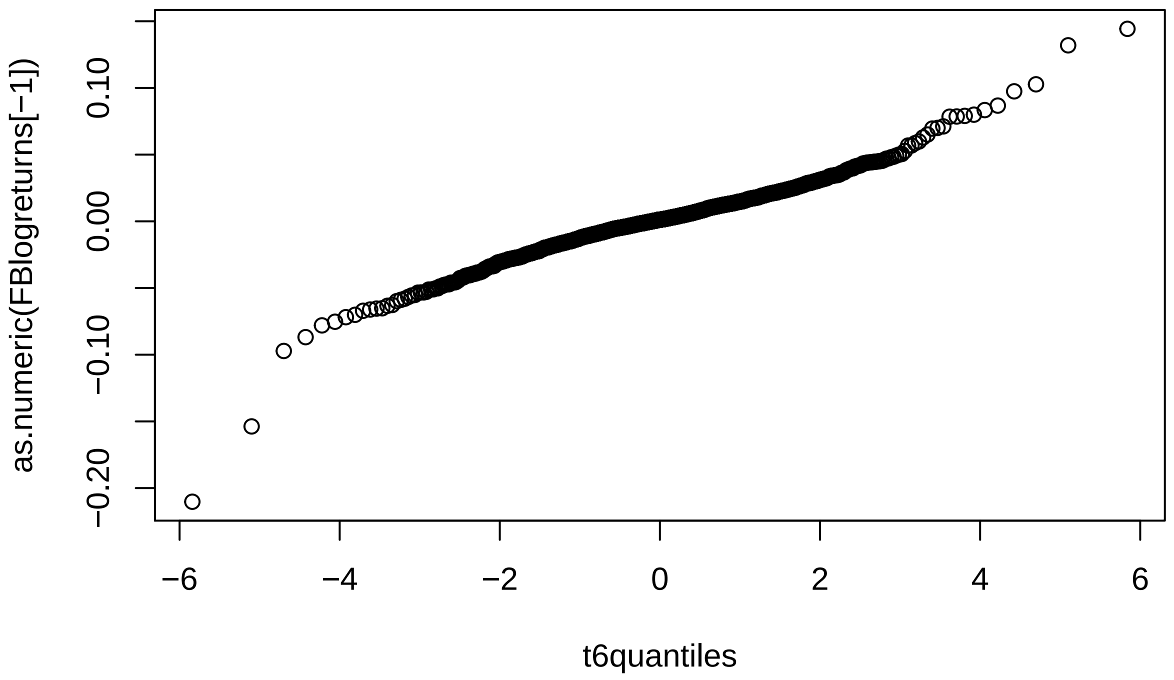
Furthermore, notice that TWTR's return distribution is more skewed in the left tail than the right. This implies that extreme negative returns are more likely to occur for TWTR than extreme positive returns, which is consistent with the conclusion I drew from the normal plot above.

Here are the t-plots with 6 degrees of freedom:

```r
t6quantiles <- qt(1:1800/1801,df=6)
qqplot(t6quantiles,as.numeric(TWTRlogreturns[-1]))
```



```r
qqplot(t6quantiles,as.numeric(FBlogreturns[-1]))
```

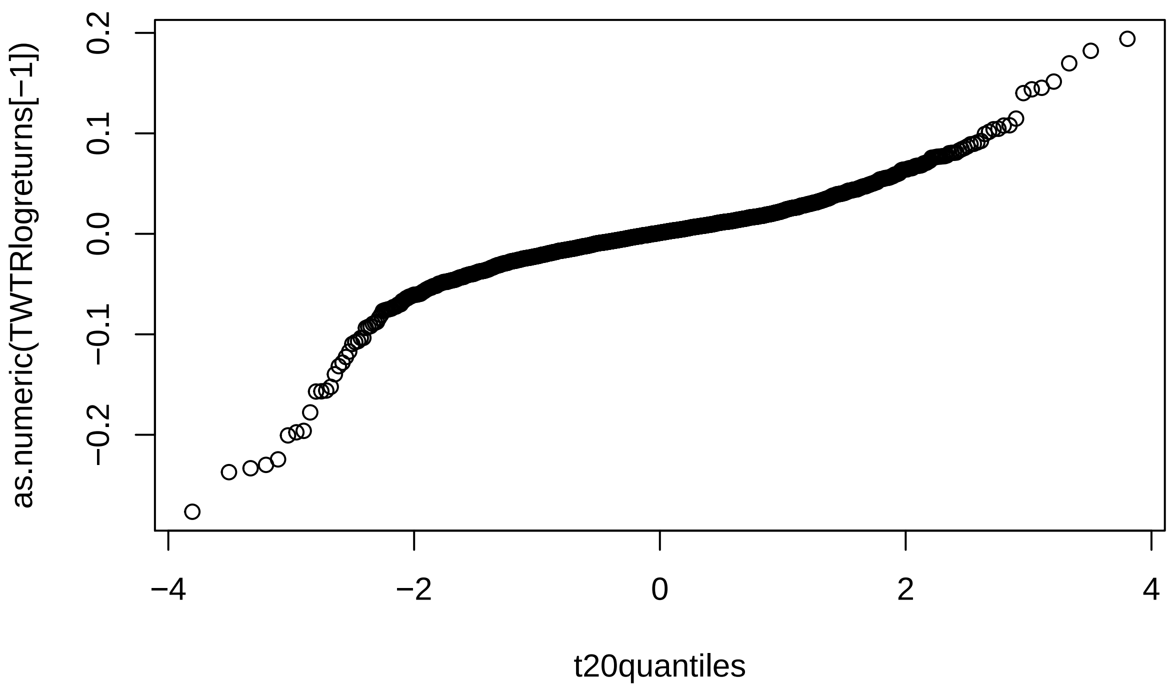


These t-plots above with 6 degrees of freedom look very similar to the previous two with 4 degrees of freedom. Once again, we do have evidence that this model with 6 degrees of freedom is a pretty good fit for both series since the points lie in roughly straight lines.

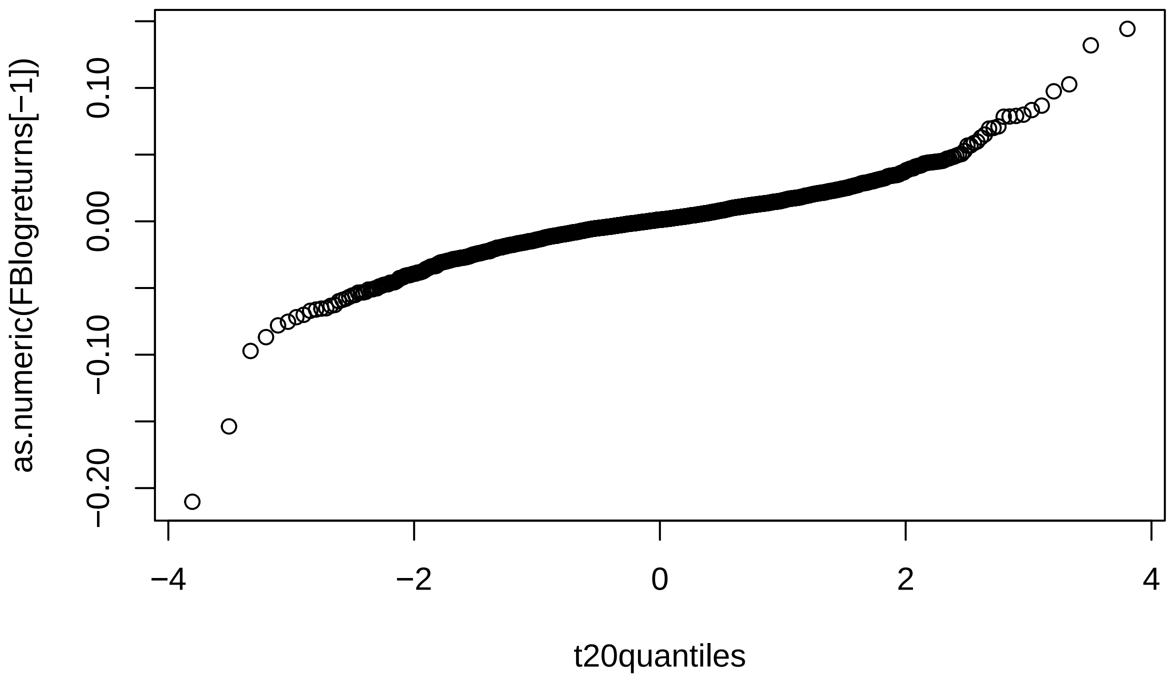
However, notice that the straight portions on the previous two t-plots for both series with 4 degrees of freedom look a little longer than those on the plots above. This suggests that the t-distribution with 4 degrees of freedom is still a slightly better fit for the series than this one with 6 degrees of freedom. Additionally, the skewness in the left and right tails of the t-plots above looks more pronounced to me than in the previous two plots with 4 degrees of freedom. This implies that the tails are fatter in TWTR and FB's distributions of log-returns than in the t-model with 6 degrees of freedom, which is consistent with my conclusion that the Student-t distribution with 4 degrees of freedom is a better fit for these series. This is because for the Student-t distribution, the tails become heavier as the degrees of freedom parameter decreases, so a t-distribution with 4 degrees of freedom has fatter tails than a t-distribution with 6 degrees of freedom.

Finally, here are the t-plots with 20 degrees of freedom:

```
t20quantiles <- qt(1:1800/1801,df=20)
qqplot(t20quantiles,as.numeric(TWTRlogreturns[-1]))
```



```r
qqplot(t20quantiles,as.numeric(FBlogreturns[-1]))
```



Notice that these t-plots actually look very similar to both of the normal plots above. From both of these plots, one can see that the tails of the distributions of log-returns for both stocks are fatter than the tails of the t-distribution with 20 degrees of freedom. This is consistent with all of the results I have obtained above.

Furthermore, while neither of these looks exactly linear, the line formed by Facebook's log returns data is a bit straighter than Twitter's. Notice that the tails on the first t-plot of TWTR's series also appear to deviate more from the straight portion in the middle than the tails on the plot of FB's log returns series. So, although the Student-t distribution with 20 degrees of freedom is not as good of a fit for either series as the t-distributions with 4 or 6 degrees of freedom, it is still probably a slightly better fit for FB's log returns data than TWTR's.

Overall, the Student-t distribution with 4 degrees of freedom looks like the best fit for FB and TWTR's log return series.

Obtaining the Predictors

Now, we are almost ready to fit our three models. With each of these methods, I would like to predict the direction of the stock price (up or down) on a particular day based on the log-returns for each of the five previous trading days. So, I will create a data frame for each stock containing the log-returns for each date between November 7th, 2013 through January 1st, 2021, along with the log-returns for the five previous trading days and whether the market was `up` or `down` on that date.

Here is the data frame for Twitter:

```
TWTRlr <- function(i)(return(as.numeric(TWTRlogreturns[i])))
TWTRToday <- c(NA, rbind(TWTRlr(2:1800)))
```

```
TWTRLag1 <- c(rep(NA,2), rbind(TWTRlr(2:1799)))
TWTRLag2 <- c(rep(NA,3), rbind(TWTRlr(2:1798)))
TWTRLag3 <- c(rep(NA,4), rbind(TWTRlr(2:1797)))
TWTRLag4 <- c(rep(NA,5), rbind(TWTRlr(2:1796)))
TWTRLag5 <- c(rep(NA,6), rbind(TWTRlr(2:1795)))
TWTRDirection <- c(NA, rep(0,1799))
for(i in 2:1800){if (TWTRToday[i]>0) {TWTRDirection[i] <- "up"} else {TWTRDirection[i] <- "down"}}
TWTRDirection <- as.factor(TWTRDirection)
olavarria_TWTR <- data.frame(TWTRToday, TWTRLag1, TWTRLag2, TWTRLag3, TWTRLag4, TWTRLag5, TWTRDirection
head(olavarria_TWTR)
```

```
##     TWTRToday    TWTRLag1    TWTRLag2    TWTRLag3    TWTRLag4 TWTRLag5
## 1          NA          NA          NA          NA          NA       NA
## 2 -0.07513642          NA          NA          NA          NA       NA
## 3  0.02957046 -0.07513642          NA          NA          NA       NA
## 4 -0.02358600  0.02957046 -0.07513642          NA          NA       NA
## 5  0.01656833 -0.02358600  0.02957046 -0.07513642          NA       NA
## 6  0.04789553  0.01656833 -0.02358600  0.02957046 -0.07513642       NA
##   TWTRDirection
## 1          <NA>
## 2          down
## 3            up
## 4          down
## 5            up
## 6            up
```

Here is the data frame for Facebook:

```
FBlr <- function(i)(return(as.numeric(FBlogreturns[i])))
FBToday <- c(NA, rbind(FBlr(2:1800)))
FBLag1 <- c(rep(NA,2), rbind(FBlr(2:1799)))
FBLag2 <- c(rep(NA,3), rbind(FBlr(2:1798)))
FBLag3 <- c(rep(NA,4), rbind(FBlr(2:1797)))
FBLag4 <- c(rep(NA,5), rbind(FBlr(2:1796)))
FBLag5 <- c(rep(NA,6), rbind(FBlr(2:1795)))
FBDirection <- c(NA, rep(0,1799))
for(i in 2:1800){if (FBToday[i]>0) {FBDirection[i] <- "up"} else {FBDirection[i] <- "down"}}
FBDirection <- as.factor(FBDirection)
olavarria_FB <- data.frame(FBToday, FBLag1, FBLag2, FBLag3, FBLag4, FBLag5, FBDirection, stringsAsFacto
head(olavarria_FB)
```

```
##         FBToday        FBLag1        FBLag2        FBLag3        FBLag4 FBLag5
## 1            NA            NA            NA            NA            NA     NA
## 2 -0.0006310233            NA            NA            NA            NA     NA
## 3 -0.0283812499 -0.0006310233            NA            NA            NA     NA
## 4  0.0088353121 -0.0283812499 -0.0006310233            NA            NA     NA
## 5  0.0440691954  0.0088353121 -0.0283812499 -0.0006310233            NA     NA
## 6  0.0057319092  0.0440691954  0.0088353121 -0.0283812499 -0.0006310233     NA
##   FBDirection
## 1        <NA>
## 2        down
## 3        down
## 4          up
## 5          up
## 6          up
```

We can also examine the summaries of our data sets:

```
summary(olavarria_TWTR)
```

```
##    TWTRToday             TWTRLag1            TWTRLag2
##  Min.   :-0.2765773   Min.   :-0.276577   Min.   :-0.2765773
##  1st Qu.:-0.0142302   1st Qu.:-0.014270   1st Qu.:-0.0143101
##  Median : 0.0010528   Median : 0.001055   Median : 0.0010566
##  Mean   : 0.0001041   Mean   : 0.000106   Mean   : 0.0001064
##  3rd Qu.: 0.0150310   3rd Qu.: 0.015032   3rd Qu.: 0.0150328
##  Max.   : 0.1940614   Max.   : 0.194061   Max.   : 0.1940614
##  NA's   :1            NA's   :2           NA's   :3
##     TWTRLag3            TWTRLag4            TWTRLag5         TWTRDirection
##  Min.   :-0.276577   Min.   :-0.276577   Min.   :-0.276577   down:872
##  1st Qu.:-0.014310   1st Qu.:-0.014311   1st Qu.:-0.014311   up  :927
##  Median : 0.001060   Median : 0.001057   Median : 0.001060   NA's:  1
##  Mean   : 0.000107   Mean   : 0.000103   Mean   : 0.000106
##  3rd Qu.: 0.015038   3rd Qu.: 0.015044   3rd Qu.: 0.015049
##  Max.   : 0.194061   Max.   : 0.194061   Max.   : 0.194061
##  NA's   :4           NA's   :5           NA's   :6
```

```
summary(olavarria_FB)
```

```
##     FBToday              FBLag1               FBLag2
##  Min.   :-0.2102387   Min.   :-0.2102387   Min.   :-0.2102387
##  1st Qu.:-0.0081638   1st Qu.:-0.0081649   1st Qu.:-0.0081616
##  Median : 0.0011904   Median : 0.0011867   Median : 0.0011904
##  Mean   : 0.0009717   Mean   : 0.0009696   Mean   : 0.0009801
##  3rd Qu.: 0.0112295   3rd Qu.: 0.0112295   3rd Qu.: 0.0112296
##  Max.   : 0.1442860   Max.   : 0.1442860   Max.   : 0.1442860
##  NA's   :1            NA's   :2            NA's   :3
##      FBLag3              FBLag4              FBLag5          FBDirection
##  Min.   :-0.210239   Min.   :-0.210239   Min.   :-0.210239   down:833
##  1st Qu.:-0.008163   1st Qu.:-0.008164   1st Qu.:-0.008165   up  :966
##  Median : 0.001192   Median : 0.001190   Median : 0.001192   NA's:  1
##  Mean   : 0.000981   Mean   : 0.000962   Mean   : 0.000964
##  3rd Qu.: 0.011236   3rd Qu.: 0.011229   3rd Qu.: 0.011230
##  Max.   : 0.144286   Max.   : 0.144286   Max.   : 0.144286
##  NA's   :4           NA's   :5           NA's   :6
```

Now, we are ready to fit our models using these six predictors.

Logistic Regression

In order to fit and test a logistic regression model for the log-return data I organized in the two data frames above, I need to download the ISLR package.

```r
install.packages("ISLR", repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/_g/f0wltjmx68x38dqpldhxy8b00000gp/T//RtmpGgQTjK/downloaded_packages
```

```r
library(ISLR)
```

I am now going to use the `glm` instruction to fit a generalized linear model for `Direction` as a function of `Lag1` through `Lag5`.

Here is the logistic regression model for Twitter's log-return data:

```r
attach(olavarria_TWTR)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     TWTRDirection, TWTRLag1, TWTRLag2, TWTRLag3, TWTRLag4, TWTRLag5,
##     TWTRToday
```

```r
TWTRglm <- glm(TWTRDirection ~ TWTRLag1 + TWTRLag2 + TWTRLag3 + TWTRLag4 + TWTRLag5, data = olavarria_TW
summary(TWTRglm)
```

```
##
## Call:
## glm(formula = TWTRDirection ~ TWTRLag1 + TWTRLag2 + TWTRLag3 +
##     TWTRLag4 + TWTRLag5, family = binomial, data = olavarria_TWTR)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.480  -1.200   1.070   1.151   1.327
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.060993   0.047309   1.289    0.197
## TWTRLag1    -2.056697   1.377667  -1.493    0.135
## TWTRLag2     0.674614   1.365107   0.494    0.621
## TWTRLag3    -1.995584   1.377033  -1.449    0.147
## TWTRLag4    -0.497326   1.365246  -0.364    0.716
## TWTRLag5     0.003385   1.361547   0.002    0.998
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2485.4  on 1793  degrees of freedom
## Residual deviance: 2480.4  on 1788  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 2492.4
##
## Number of Fisher Scoring iterations: 3
```

From the results above, it appears that `Lag1` may be the best predictor for the response variable `Direction` for Twitter's log-return data, since it has the smallest p-value for this model. `Lag1` represents the log-return from the previous trading day, so a negative coefficient for this regressor indicates that if TWTR's log-return yesterday was positive, it is less likely to increase today. `Lag3` also has a relatively low p-value compared to the other variables (0.147), but it is still larger than the p-value of `Lag1` (0.135). So, `Lag1` probably contributes more significantly to the model than `Lag3`, but we still do not have much evidence that either predictor is meaningful since both of their p-values are relatively large.

Additionally, notice that the p-values associated with a some of the predictors are extremely high, so it is possible that we may have too many predictors in this model. The p-value of `Lag5` is 0.998 (almost 1), so it is probably not be necessary in this model. In particular, if `Lag5` were actually not part of this model and had a coefficient of 0, the chances of obtaining an estimate at least as large as 0.003385 are almost 100%. `Lag2` and `Lag4` also have pretty high p-values (0.621 and 0.716), so I am not confident that either of these predictor variables contribute meaningfully to the model either.

Similarly, here is the model for Facebook's log-returns data:

```r
attach(olavarria_FB)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     FBDirection, FBLag1, FBLag2, FBLag3, FBLag4, FBLag5, FBToday
```

```r
FBglm <- glm(FBDirection ~ FBLag1 + FBLag2 + FBLag3 + FBLag4 + FBLag5, data = olavarria_FB, family = bi
summary(FBglm)
```

```
##
## Call:
## glm(formula = FBDirection ~ FBLag1 + FBLag2 + FBLag3 + FBLag4 +
##     FBLag5, family = binomial, data = olavarria_FB)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7296  -1.2201   0.9758   1.1140   1.6928
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.16847    0.04809   3.503 0.000460 ***
## FBLag1      -4.94351    2.38090  -2.076 0.037864 *
## FBLag2      -5.34574    2.37844  -2.248 0.024602 *
## FBLag3      -9.18325    2.45674  -3.738 0.000185 ***
## FBLag4       1.54656    2.34368   0.660 0.509328
## FBLag5      -0.99963    2.34574  -0.426 0.669999
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2477.3  on 1793  degrees of freedom
## Residual deviance: 2453.6  on 1788  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 2465.6
##
## Number of Fisher Scoring iterations: 4
```

At first glance, this logistic regression model for Facebook's log-return data appears to be much better than the previous model for Twitter's, since all of the p-values for the coefficients are lower.

Based on this summary output, it appears that `Lag3` is the best predictor for the `Direction` variable out of Lags 1 through 5 since it has a p-value of 0.000185, which is significant at the 0.001 level. `Lag1` and `Lag2` also have relatively low p-values compared to the rest of the variables (0.037864 and 0.024602), which both significant at the 0.05 level. So, Lags 1 and 2 probably contribute meaningfully to the model, but `Lag2` may be a better predictor for `Direction` than `Lag1` since it has a slightly lower p-value.

The worst predictors for the direction of FB's log-returns seem to be Lags 4 and 5, since they have the highest p-values above (0.509328 and 0.669999). Therefore, I am not confident that either of these variables contributes significantly to the model, but `Lag4` is probably a slightly better predictor than `Lag5`.

Now, I can use the `predict` function to predict the probability that the direction of the market will go up (or down) on a particular day, given values of the predictor variables. The option `type = "response"` indicates that the probabilities in the output will be of the form $P(Y = 1|X)$. I can confirm that these probabilities correspond to the probability that the direction of the market is `up` using the `contrasts` function:

```
contrasts(TWTRDirection)
```

```
##      up
## down  0
## up    1
```

```
contrasts(FBDirection)
```

```
##      up
## down  0
## up    1
```

This output indicates that `R` has created dummy variables with 1s for `up` and 0s for `down`. So, we will be predicting the probability that the market is going `up` given values of Lags 1 through 5.

After obtaining my predictions, I will use the `table` function to produce a confusion matrix in order to determine how many of these observations were correctly or incorrectly classified as `up` or `down`.

Here are the predictions for Twitter's log-return data:

```
TWTRprobs <- predict(TWTRglm, type = "response")
TWTRpred <- c(rep("down", 1800))
TWTRpred[TWTRprobs > 0.5] <- "up"
table(TWTRpred, TWTRDirection)
```

```
##         TWTRDirection
## TWTRpred down  up
##     down  196 186
##     up    676 741
```

So, the logistic regression model for Twitter's log-returns correctly predicted that the market will go `up` on 741 days and that it will go `down` on 196 days (out of $1,799$ total, since we have no data for the first trading day). So, we have a total of $741 + 196 = 937$ correct predictions here.

Here are the predictions for Facebook's log-return data:

```
FBprobs <- predict(FBglm, type = "response")
FBpred <- c(rep("down", 1800))
FBpred[FBprobs > 0.5] <- "up"
table(FBpred, FBDirection)
```

```
##       FBDirection
## FBpred down  up
##   down  188 184
##   up    645 782
```

These results indicate that the model for Facebook's log-returns correctly predicted that the market will go `up` on 782 days and that it will go `down` on 188 days (out of 1,799 total). So, we have a total of $782+188 = 970$ correct predictions here.

We can also compute the fraction of days for which the predictions were correct with the `mean` function. Notice that I am using the `as.numeric` command here, so the elements in the objects we are computing the averages of will be `0` or `1`, rather than `TRUE` or `FALSE`. Additionally, I am omitting the first entries in the objects below because they are both `NA`s (since we do not have any data for the log-returns of the first trading day in either data set).

```r
mean(as.numeric(TWTRpred==TWTRDirection)[-1])
```

```
## [1] 0.5208449
```

```r
mean(as.numeric(FBpred==FBDirection)[-1])
```

```
## [1] 0.5391884
```

Therefore, the logistic regression model correctly predicted the movement of the market about 52.09% of the time for Twitter's log-returns, and about 53.92% of the time for Facebook's log-returns. However, since we trained and tested these models on the same sets of data, this implies that the training error rates for the models are $100 - 52.09 = 47.91\%$ and $100 - 53.92 = 46.08\%$. The training error rate tends to underestimate the test error rate, so we need a more realistic error rate in order to better assess the accuracy of these logistic regression models. In order to do this, we can fit the model using only part of the data, and then observe how accurately it predicts the remaining data.

First, I am going to fit each model using only a subset of the observations that correspond to the first 1,200 trading days (or, the first 1,199 observations that we have log-return data for), the training set. Then, I will predict the probabilities of the market going up for each of the days in the test set containing the observations corresponding to the last 600 trading days. Finally, I will produce a confusion matrix and calculate the fraction of days for which the predictions were correct in addition to the test set error rate.

Here are the results for Twitter's data:

```r
TWTRtrain = olavarria_TWTR[1:1200,]
TWTRtest = olavarria_TWTR[1201:1800,]
TWTRtestDir = TWTRDirection[1201:1800]
TWTRglm2 <- glm(TWTRDirection ~ TWTRLag1 + TWTRLag2 + TWTRLag3 +  TWTRLag4 + TWTRLag5, data = TWTRtrain
TWTRprobs2 <- predict(TWTRglm2, TWTRtest, type = "response")
TWTRpred2 <- c(rep("down", 600))
TWTRpred2[TWTRprobs2 > 0.5] <- "up"
table(TWTRpred2, TWTRtestDir)
```

```
##          TWTRtestDir
## TWTRpred2 down  up
##      down  153 167
##      up    119 161
```

```
mean(as.numeric(TWTRpred2==TWTRtestDir)[-1])
```

```
## [1] 0.5225376
```

```
mean(as.numeric(TWTRpred2!=TWTRtestDir)[-1])
```

```
## [1] 0.4774624
```

So, about 52.25% of the predictions that the direction of the market would be `up` for each of the days in the test set were correct. Additionally, we obtain a test set error rate of about 47.75%, which is only slightly better than random guessing.

Now, here are the results for Facebook's log-return data:

```
FBtrain = olavarria_FB[1:1200,]
FBtest = olavarria_FB[1201:1800,]
FBtestDir = FBDirection[1201:1800]
FBglm2 <- glm(FBDirection ~ FBLag1 + FBLag2 + FBLag3 + FBLag4 + FBLag5, data = FBtrain, family = binomi
FBprobs2 <- predict(FBglm2, FBtest, type = "response")
FBpred2 <- c(rep("down", 600))
FBpred2[FBprobs2 > 0.5] <- "up"
table(FBpred2, FBtestDir)
```

```
##        FBtestDir
## FBpred2 down  up
##    down   66  63
##    up    220 251
```

```
mean(as.numeric(FBpred2==FBtestDir)[-1])
```

```
## [1] 0.5292154
```

```
mean(as.numeric(FBpred2!=FBtestDir)[-1])
```

```
## [1] 0.4707846
```

Based on these results, about 52.92% of the predictions that the direction of the market would be `up` for each of the days in the test set are correct. Furthermore, we obtain a test set error rate of about 47.08%, which is also only slightly better than random guessing.

Overall, the predictions from the logistic regression model for Facebook's log-return series are slightly more accurate than the predictions for Twitter's. This is not what I was expecting, since Facebook's log-return data looked a little closer to the normal distribution than Twitter's in the quantile-quantile plots I created. So, I initially thought that the logistic regression method may produce slightly more accurate results for Twitter's data than Facebook's. However, both of these test set errors are pretty high, so I am not confident that this model is a great fit for either series.

Linear Discriminant Analysis

Now, I am going to perform Linear Discriminant Analysis (LDA) on the log-return data for Twitter and Facebook. In order to fit and test an LDA model for this data, I will use the `MASS` library and the `lda` function. Once again, I will be fitting these models for the first 1,200 observations in the data sets and then testing them on the last 600 observations.

```
library(MASS)
```

Here is the LDA model for Twitter:

```
TWTRlda <- lda(TWTRDirection ~ TWTRLag1 + TWTRLag2 + TWTRLag3 + TWTRLag4 + TWTRLag5, data = TWTRtrain)
TWTRlda
```

```
## Call:
## lda(TWTRDirection ~ TWTRLag1 + TWTRLag2 + TWTRLag3 + TWTRLag4 +
##     TWTRLag5, data = TWTRtrain)
##
## Prior probabilities of groups:
##      down        up
## 0.5008375 0.4991625
##
## Group means:
##           TWTRLag1      TWTRLag2      TWTRLag3      TWTRLag4      TWTRLag5
## down  0.0009186748 -0.0011487527  0.0007924836  0.0000200131 -5.846732e-04
## up   -0.0013603914  0.0007008667 -0.0012890805 -0.0004707146  5.311052e-05
##
## Coefficients of linear discriminants:
##                LD1
## TWTRLag1 -17.268927
## TWTRLag2  14.500424
## TWTRLag3 -15.807860
## TWTRLag4  -4.393341
## TWTRLag5   5.567170
```

From this LDA output, one can see that $\hat{\pi}_1 = 0.5008375$ and $\hat{\pi}_2 = 0.4991625$. This indicates that about 50.08% of the observations in Twitter's log-return data correspond to days in which the market went `down`, while about 49.92% correspond to days when the market went `up`.

We also have information about the average log-returns for the previous 5 trading days on days when Twitter's stock price went `down` or `up` from the group means in this output. These group means imply that on days when Twitter's price went `down`, there was a tendency for the previous 5 days' log-returns to be positive, except for the return 2 and 5 days before. Additionally, on days when Twitter's price went `up`, there was a tendency for the previous 5 days' log-returns to be negative, except for the returns 2 and 5 days before.

Here is the LDA model for Facebook:

```
FBlda <- lda(FBDirection ~ FBLag1 + FBLag2 + FBLag3 + FBLag4 + FBLag5, data = FBtrain)
FBlda
```

```
## Call:
## lda(FBDirection ~ FBLag1 + FBLag2 + FBLag3 + FBLag4 + FBLag5,
##     data = FBtrain)
```

```
##
## Prior probabilities of groups:
##      down        up
## 0.4564489 0.5435511
##
## Group means:
##            FBLag1       FBLag2        FBLag3       FBLag4       FBLag5
## down 0.0018848965 0.0018238566  0.0030112115 0.0007943251 0.0017415270
## up   0.0004315588 0.0005525165 -0.0004069527 0.0014284438 0.0006206137
##
## Coefficients of linear discriminants:
##               LD1
## FBLag1 -20.444889
## FBLag2 -17.400959
## FBLag3 -44.369011
## FBLag4   5.833104
## FBLag5 -17.972322
```

The LDA output above tells us that the prior probabilities of groups are $\hat{\pi}_1 = 0.4564489$ and $\hat{\pi}_2 = 0.5435511$, meaning that about 45.64% of the observations in Facebook's log-return data correspond to days in which the market went `down`, while about 54.36% correspond to days when the market went `up`.

Additionally, we have information about the average log-returns for the previous 5 trading days on days when Facebook's stock price went `down` or `up`. These group means suggest that there was a tendency for the previous 5 days' log-returns to be positive on days when Facebook's stock price went `down`. On the other hand, on days when Facebook's stock price went `up`, there was a tendency for the previous 5 days' log-returns to be positive.

Now, I am going to test the accuracy of the predictions obtained from these models using the test sets containing only data from the last 600 trading days.

Here are the predictions for Twitter's data:

```r
TWTRlda.pred <- predict(TWTRlda, TWTRtest)
TWTRlda.class <- TWTRlda.pred$class
table(TWTRlda.class, TWTRtestDir)
```

```
##              TWTRtestDir
## TWTRlda.class down  up
##          down  153 167
##          up    119 161
```

```r
mean(as.numeric(TWTRlda.class==TWTRtestDir)[-1])
```

```
## [1] 0.5225376
```

The LDA model above correctly predicted that the direction of Twitter's stock price would be `up` for about 52.25% of the observations in the test set. Notice that the results of this test are almost identical to the ones we obtained from the logistic regression model. So, there does not seem to be much of a difference between the accuracy of the LDA and logistic regression approaches in predicting the direction of the market for Twitter's data.

Here are the LDA predictions for Facebook:

```
FBlda.pred <- predict(FBlda, FBtest)
FBlda.class <- FBlda.pred$class
table(FBlda.class, FBtestDir)
```

```
##            FBtestDir
## FBlda.class down  up
##        down   62  58
##        up    224 256
```

```
mean(as.numeric(FBlda.class==FBtestDir)[-1])
```

```
## [1] 0.5308848
```

Now, about 53.09% of the predictions that the direction of the market would be `up` for each of the days in the test set were correct. This is slightly higher than what we obtained from the logistic regression model, so the LDA predictions are probably a little better than the logistic regression predictions for Facebook's log-return data. This is consistent with the normal probability plots I produced above, because LDA tends to provide more accurate results when the Gaussian assumption approximately holds and Facebook's data is much closer to the normal distribution than Twitter's.

Overall, the LDA approach appears to have slightly more accurate predictions for the observations in Facebook's data than Twitter's, but it is still not much better than the logistic regression model for either.

K-Nearest Neighbors

For this problem, I will fit and test a K-Nearest-Neighbors (KNN) model for the log-return data using the `knn` function from the `class` library.

```
library(class)
```

The `knn` command does not work if there are missing entries in any of the inputs, so notice that I am omitting the first 6 rows in each training set that contain `NA`s. Here are the results of this test for Twitter's log-return data with `k = 1`:

```
train.TWTR <- data.frame(TWTRLag1, TWTRLag2, TWTRLag3,  TWTRLag4, TWTRLag5)[7:1200,]
test.TWTR <- data.frame(TWTRLag1, TWTRLag2, TWTRLag3,  TWTRLag4, TWTRLag5)[1201:1800,]
train.TWTR.Dir = TWTRDirection[7:1200]
set.seed(1)
knn.pred.TWTR <- knn(train.TWTR, test.TWTR, train.TWTR.Dir, k = 1)
table(knn.pred.TWTR, TWTRtestDir)
```

```
##              TWTRtestDir
## knn.pred.TWTR down  up
##          down  138 161
##          up    134 167
```

```
mean(as.numeric(knn.pred.TWTR==TWTRtestDir))
```

```
## [1] 0.5083333
```

So, about 50.83% of the observations are correctly predicted in this model using $K = 1$.

I am going to repeat this argument again for Twitter's data using `k = 3`:

```
set.seed(1)
knn.pred.TWTR <- knn(train.TWTR, test.TWTR, train.TWTR.Dir, k = 3)
table(knn.pred.TWTR, TWTRtestDir)
```

```
##              TWTRtestDir
## knn.pred.TWTR down  up
##          down  137 168
##          up    135 160
```

```
mean(as.numeric(knn.pred.TWTR==TWTRtestDir))
```

```
## [1] 0.495
```

Now, only about 49.50% of the observations were correctly predicted using $K = 3$, so the model above with $K = 1$ is slightly better.

Here are the results for `k = 9`:

```
set.seed(1)
knn.pred.TWTR <- knn(train.TWTR, test.TWTR, train.TWTR.Dir, k = 9)
table(knn.pred.TWTR, TWTRtestDir)
```

```
##              TWTRtestDir
## knn.pred.TWTR down  up
##          down  144 159
##          up    128 169
```

```
mean(as.numeric(knn.pred.TWTR==TWTRtestDir))
```

```
## [1] 0.5216667
```

The results have improved compared to the first two tests above, and about 52.17% of the observations were correctly predicted using $K = 9$. Additionally, increasing K further does not seem to improve our predictions here. So, I obtained the most accurate predictions for Twitter's log-return data using the k-nearest-neighbors approach with $K = 9$.

Now, I will repeat the same approach for Facebook. Here are the results using `k = 1`:

```
train.FB <- data.frame(FBLag1, FBLag2, FBLag3,  FBLag4, FBLag5)[7:1200,]
test.FB <- data.frame(FBLag1, FBLag2, FBLag3,  FBLag4, FBLag5)[1201:1800,]
train.FB.Dir = FBDirection[7:1200]
set.seed(1)
knn.pred.FB <- knn(train.FB, test.FB, train.FB.Dir, k = 1)
table(knn.pred.FB, FBtestDir)
```

```
##            FBtestDir
## knn.pred.FB down  up
##        down  123 137
##        up    163 177
```

```
mean(as.numeric(knn.pred.FB==FBtestDir))
```

```
## [1] 0.5
```

So, about half of the KNN predictions obtained for Facebook's log-return data with $K = 1$ are correct.

Here are the results using `k = 5`:

```
set.seed(1)
knn.pred.FB <- knn(train.FB, test.FB, train.FB.Dir, k = 5)
table(knn.pred.FB, FBtestDir)
```

```
##            FBtestDir
## knn.pred.FB down  up
##        down  111 138
##        up    175 176
```

```
mean(as.numeric(knn.pred.FB==FBtestDir))
```

```
## [1] 0.4783333
```

This result is slightly worse than the previous one obtained using $K = 1$. Now, only about 47.83% of the predictions are correct.

Once again, I am going to repeat this argument again for Facebook's data using `k = 7`:

```
set.seed(1)
knn.pred.FB <- knn(train.FB, test.FB, train.FB.Dir, k = 7)
table(knn.pred.FB, FBtestDir)
```

```
##            FBtestDir
## knn.pred.FB down  up
##        down  113 120
##        up    173 194
```

```
mean(as.numeric(knn.pred.FB==FBtestDir))
```

```
## [1] 0.5116667
```

The results above from using $K = 7$ are the best I have obtained so far for Facebook's data out of the past three. Now, about 51.17% of the predictions are correct. Additionally, increasing K further does not seem to improve the predictions here. So, I obtained the most accurate predictions for Facebook's log-return data using the k-nearest-neighbors approach with $K = 7$.

Overall, the performance of KNN was better for predicting the observations within Twitter's log-return data than Facebook's.

Conclusion

To summarize the results I have obtained from the logistic regression, linear discriminant analysis, and k-nearest neighbors approaches, here are the percentages of accurate predictions obtained from each approach.

Here are the results from Twitter's log-return series:

- Logistic regression: 52.25%
- LDA: 52.25%
- KNN ($K = 1$): 50.83%
- KNN ($K = 3$): 49.50%
- KNN ($K = 9$): 52.17%

Here are the results from Facebook's log-return series:

- Logistic regression: 52.92%
- LDA: 53.09%
- KNN ($K = 1$): 50.00%
- KNN ($K = 5$): 47.83%
- KNN ($K = 7$): 51.17%

Overall, the logistic regression and LDA approaches gave the most accurate predictions for both Facebook and Twitter's log-return series. But, although the results of the two approaches were almost identical for Twitter, LDA was a little better for Facebook than logistic regression. This is probably due to the different distributions of the data, since LDA is more accurate when the observations are drawn from a normal distribution. Furthermore, recall that this data includes observations from from November 7th, 2013 (when Twitter went public) until January 1st, 2021. Since Facebook has been publicly traded for longer than Twitter, they had more pricing data available and it was much less volatile. Therefore, it is not at all surprising that Facebook's log-returns more closely resemble the normal distribution than Twitter's, and thus the LDA approach worked better for Facebook's data. So, if I were to repeat this project again, I would select two companies in the same industry that have each been publicly traded for long enough that their prices have stabilized a bit.