Stock prediction, linear regression/SVM

Jacob Mengistu

2024-09-23

Load Libraries

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggpubr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## v readr     2.1.5

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(broom)
library(AICcmodavg)
library(plyr)
```

```
## ------------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------------
##
## Attaching package: 'plyr'
##
## The following object is masked from 'package:purrr':
##
##     compact
##
## The following object is masked from 'package:ggpubr':
##
##     mutate
##
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(rstatix)
```

```
##
## Attaching package: 'rstatix'
##
## The following objects are masked from 'package:plyr':
##
##     desc, mutate
##
## The following object is masked from 'package:stats':
##
##     filter
```

```
library(stats)
library("lubridate")
library(e1071)
```

Load dataset and view dataframe

```
df<-read.csv('ADANIPORTS.csv')
head(df)
```

```
##         Date     Symbol Series Prev.Close   Open    High Low Last   Close
## 1 2007-11-27 MUNDRAPORT     EQ     440.00 770.00 1050.00 770  959  962.90
```

```
## 2 2007-11-28 MUNDRAPORT     EQ     962.90 984.00  990.00 874  885  893.90
## 3 2007-11-29 MUNDRAPORT     EQ     893.90 909.00  914.75 841  887  884.20
## 4 2007-11-30 MUNDRAPORT     EQ     884.20 890.00  958.00 890  929  921.55
## 5 2007-12-03 MUNDRAPORT     EQ     921.55 939.75  995.00 922  980  969.30
## 6 2007-12-04 MUNDRAPORT     EQ     969.30 985.00 1056.00 976 1049 1041.45
##      VWAP   Volume     Turnover Trades Deliverable.Volume X.Deliverble
## 1  984.72 27294366 2.687719e+15     NA            9859619       0.3612
## 2  941.38  4581338 4.312765e+14     NA            1453278       0.3172
## 3  888.09  5124121 4.550658e+14     NA            1069678       0.2088
## 4  929.17  4609762 4.283257e+14     NA            1260913       0.2735
## 5  965.65  2977470 2.875200e+14     NA             816123       0.2741
## 6 1015.39  4849250 4.923867e+14     NA            1537667       0.3171
```

Create a year column to group by

```
df$year <- year(df$Date)
```

Check and remove rows w/null values

```
sum(is.na(df))
```

```
## [1] 866
```

```
df <- na.omit(df)
```

Summarizing open, close, high, low by year

Open summary by year

```
 Open_summary <- df %>% group_by(year) %>% summarize(mean = mean(Open), median = median(Open), min= min
print(Open_summary)
```

```
##       mean  median min max
## 1 284.9406 299.475 108 857
```

Close summary by year

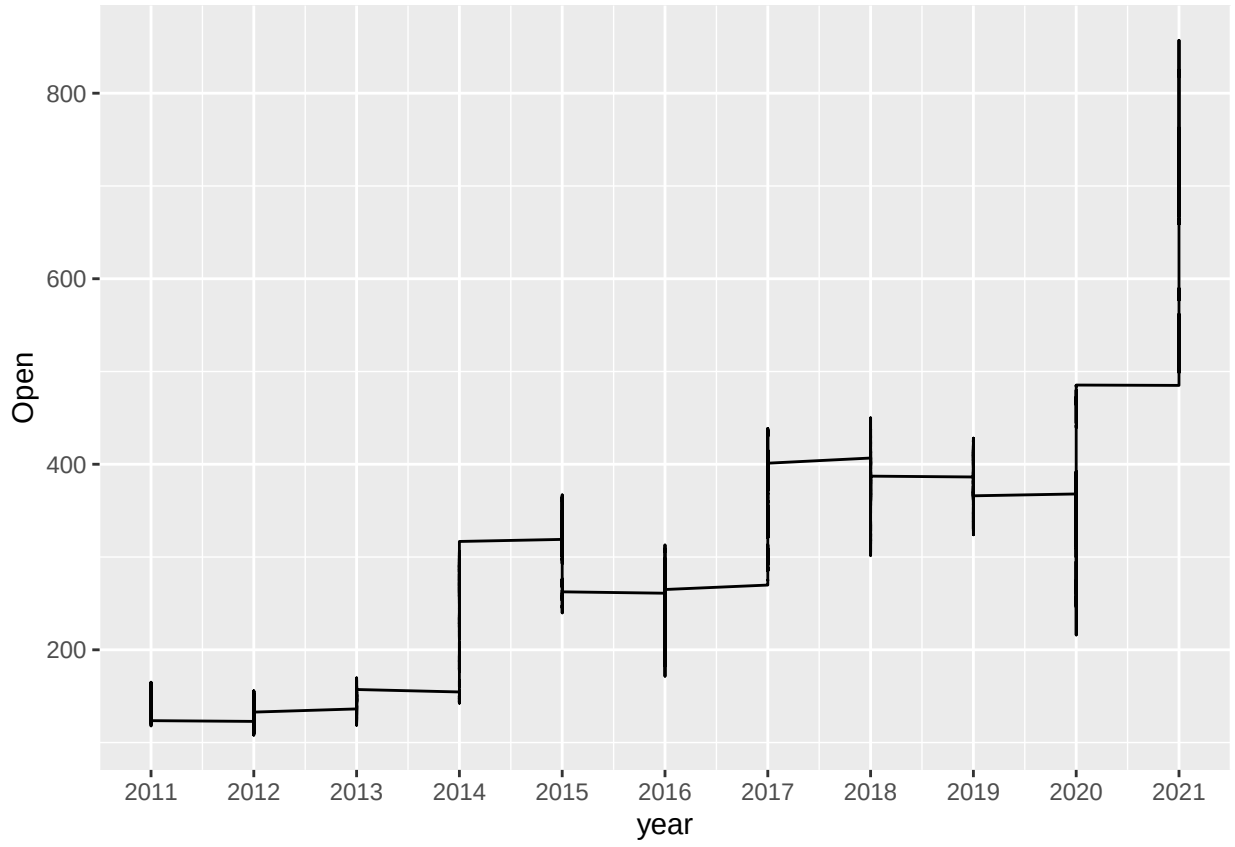
```
Close_summary <- df %>% group_by(year) %>% summarize(mean = mean(Close), median = median(Close), min= m
print(Close_summary)
```

```
##       mean median min    max
## 1 284.6273 298.75 108 835.55
```

Random plots for fun

compare opening price through the years w line graph

```
ggplot(data = df, aes(x = year, y =Open)) + geom_line()+ scale_x_continuous(breaks = seq(min(df$year), m
```

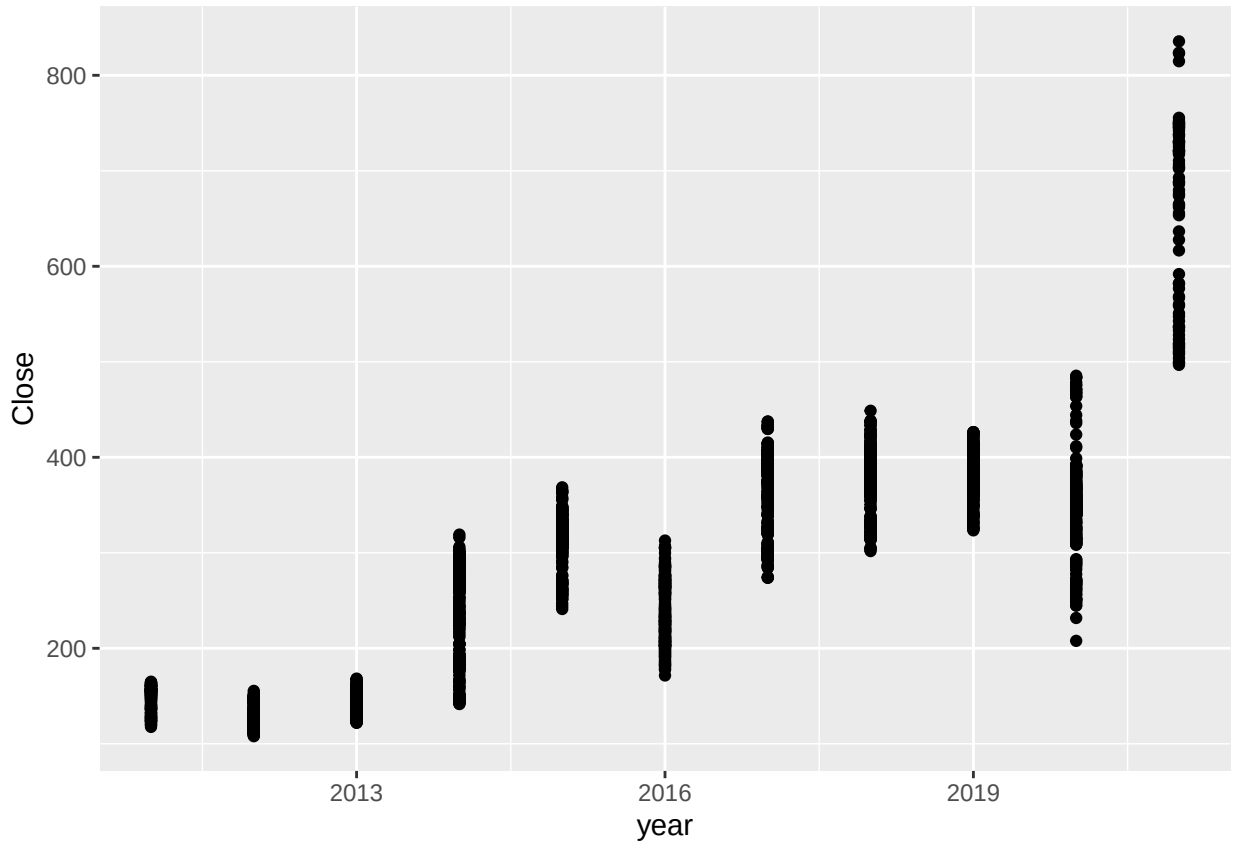


Create a scatter plot for open price

ggplot(data = df, aes(x = year, y = Open, color = year)) + geom_point()+ scale_x_continuous(breaks = seq(min(df$year), max(df$year), by = 1))

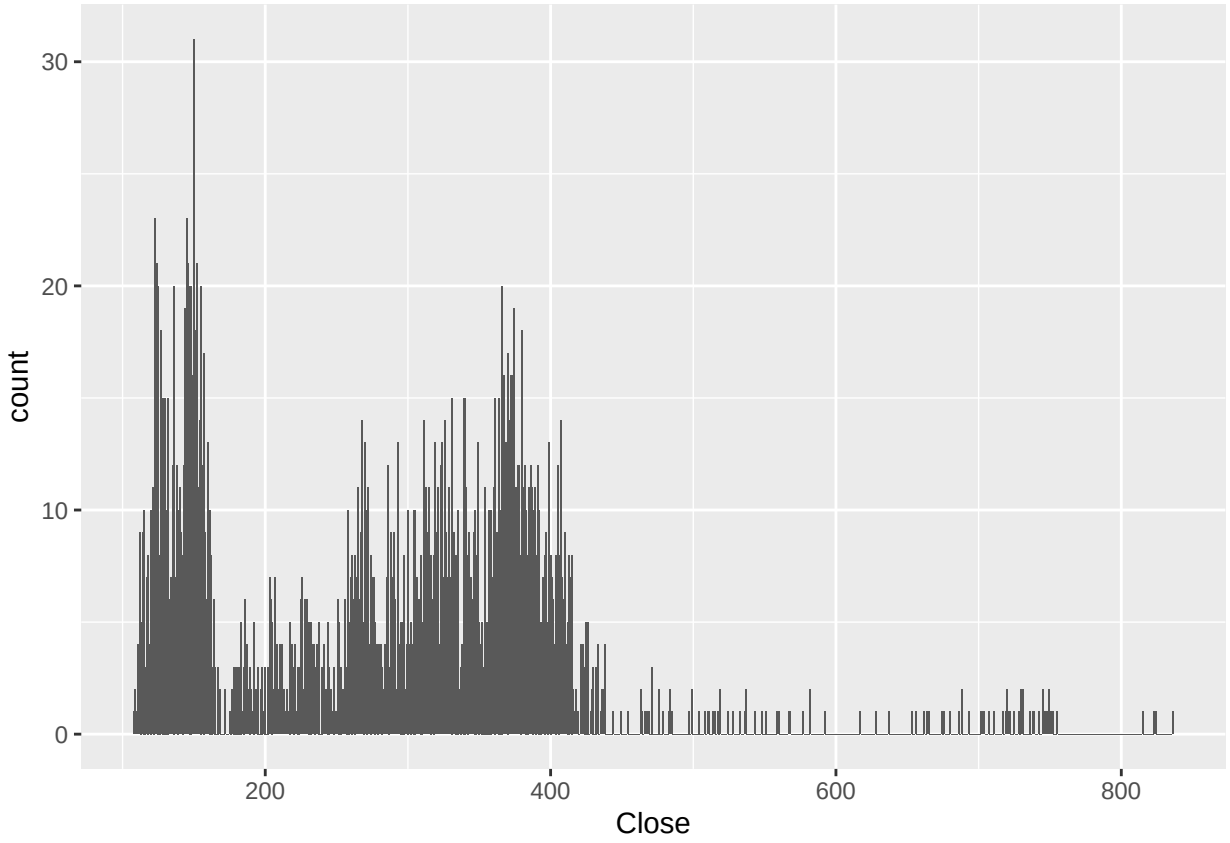
Scatterplot of year and close price

```
ggplot(df, aes(x = year, y = Close)) + geom_point()
```



Histogram for close price

```
ggplot(df, aes(x = Close)) + geom_histogram(binwidth = 1) #Line plot of open prices plot(df$Open,type='
```



Create a target column where 'Up' if Close > Open, 'Down' otherwise for SVM

```
df$target <- ifelse(df$Close > df$Open, "Up", "Down")
```

Convert the target column to a factor

```
df$target <- as.factor(df$target)
```

Check the levels of the factor to make sure they are "Up" and "Down"

```
levels(df$target)
```

```
## [1] "Down" "Up"
```

Split the data into training and test sets

Set seed

```r
set.seed(123)
```

Training data

```r
train_indices <- sample(1:nrow(df), 0.8*nrow(df))
train <- df[train_indices, ]
```

Testing data

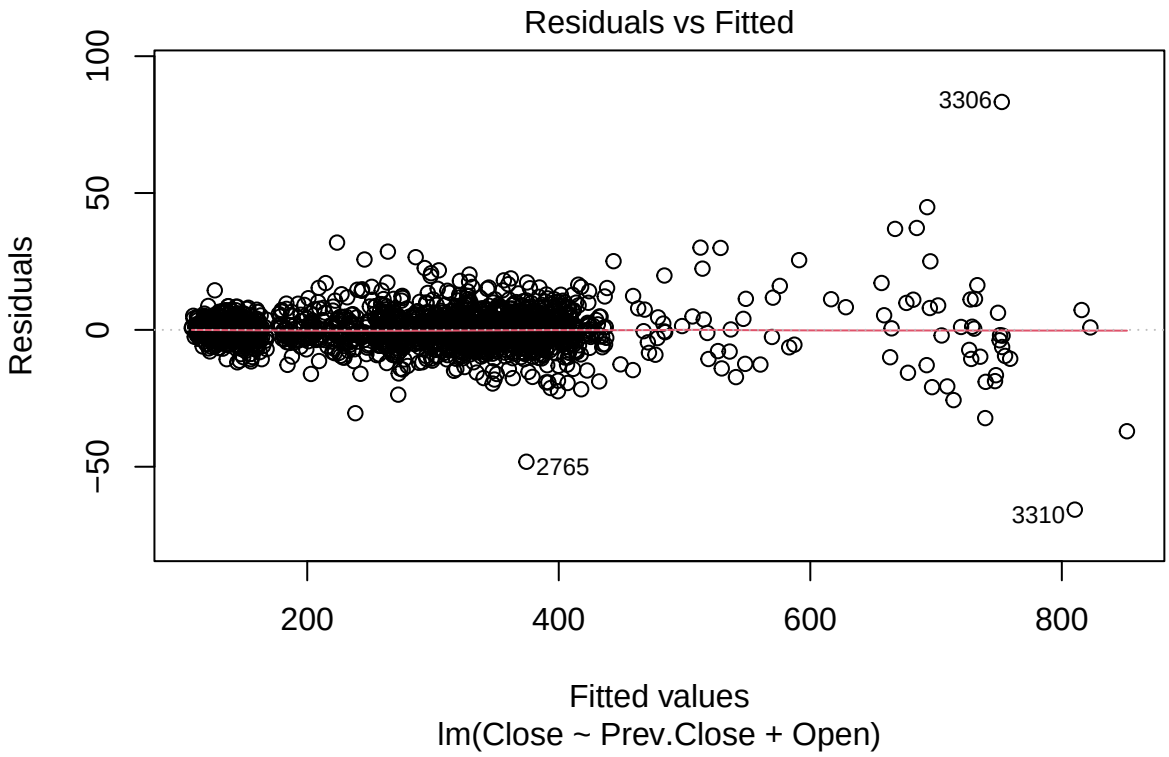
```r
test <- df[-train_indices, ]
```

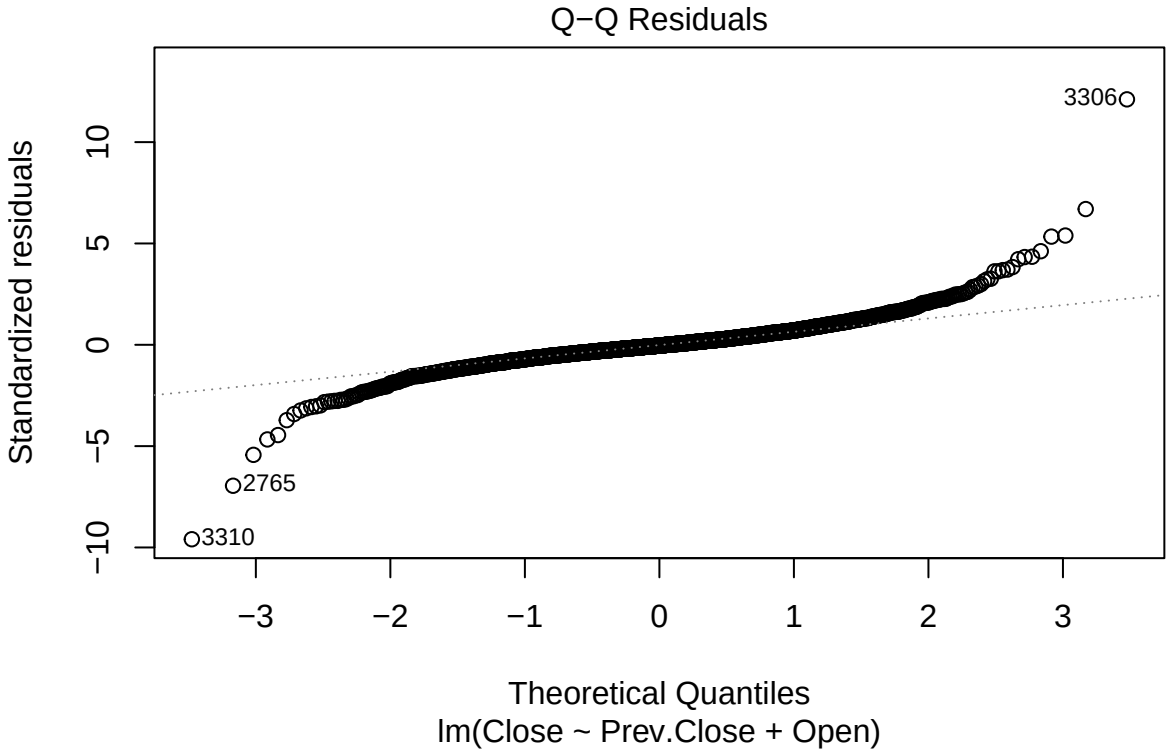
Fit the linear regression model for close based on previous close prev.close

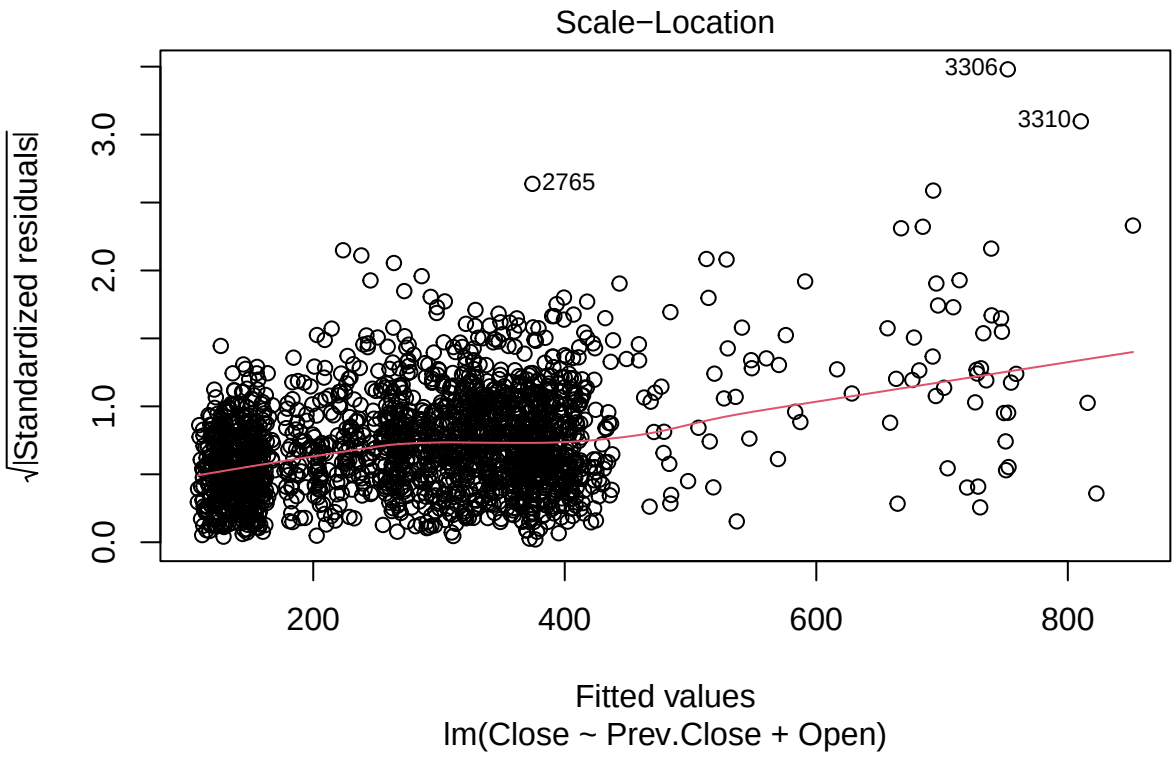
```r
model_fit <- lm(Close ~Prev.Close + Open , data = train)
close_predictions <- predict(model_fit, newdata = test)
summary(model_fit)
```

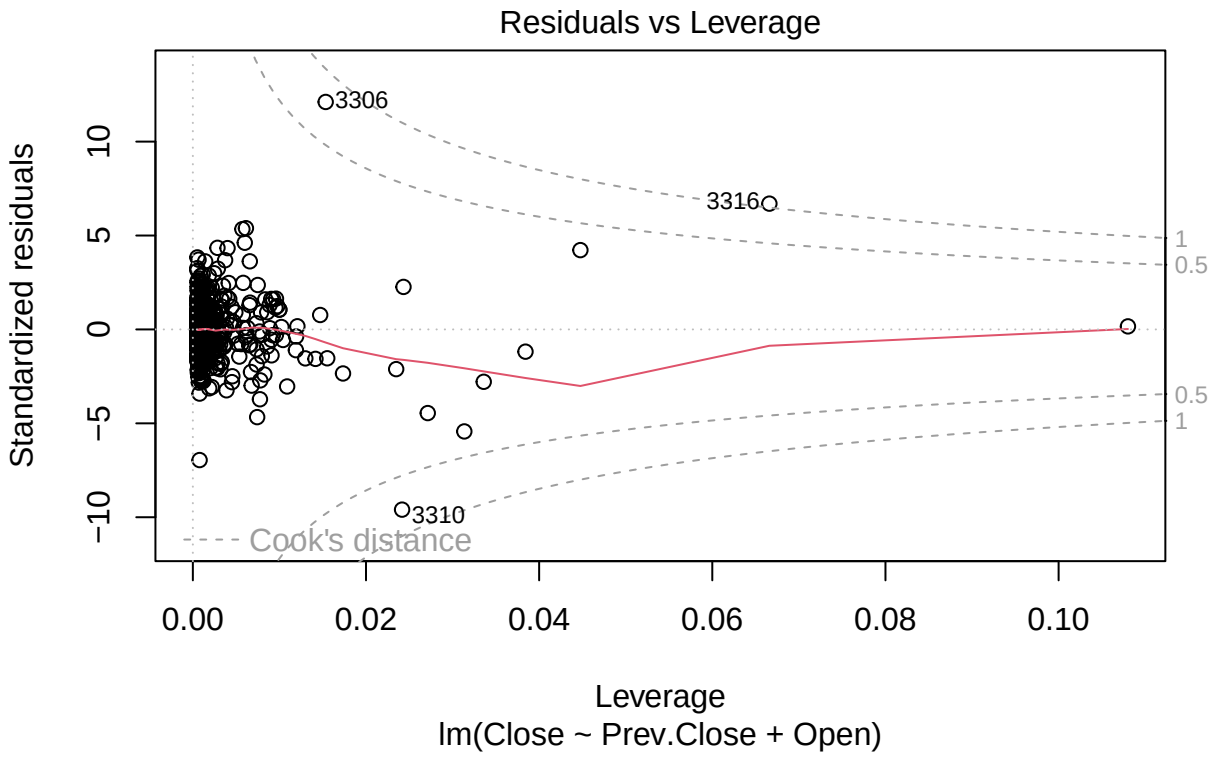
```
##
## Call:
## lm(formula = Close ~ Prev.Close + Open, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -65.674  -3.171  -0.259   2.974  83.275
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.54987    0.39354   1.397  0.16250
## Prev.Close   0.15665    0.05183   3.023  0.00254 **
## Open         0.84054    0.05166  16.271  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.929 on 1961 degrees of freedom
## Multiple R-squared:  0.9969, Adjusted R-squared:  0.9969
## F-statistic: 3.108e+05 on 2 and 1961 DF,  p-value: < 2.2e-16
```

```r
plot(model_fit)
```


Residuals vs Fitted
Residuals
100
50
0
−50
3306
2765
3310
200
400
600
800
Fitted values
lm(Close ~ Prev.Close + Open)


Q−Q Residuals
Standardized residuals
10
5
0
−5
−10
3306
2765
3310
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(Close ~ Prev.Close + Open)


Scale−Location
√|Standardized residuals|
0.0
1.0
2.0
3.0
200
400
600
800
3306
3310
2765
Fitted values
lm(Close ~ Prev.Close + Open)



Evaluate the model using RMSE and R-squared

```
mse <- mean((test$Close - close_predictions)^2)

rmse <- sqrt(mse)

mae <- mean(abs(test$Close - close_predictions))

r_squared <- cor(test$Close, close_predictions)^2

mse <- mean((test$Close - close_predictions)^2)

rmse <- sqrt(mse)

mae <- mean(abs(test$Close - close_predictions))

r_squared <- cor(test$Close, close_predictions)^2
```

Print results

```r
print(paste("MSE for closing price:", mse))
```

```
## [1] "MSE for closing price: 40.9712553764558"
```

```r
print(paste("RMSE for closing price:", rmse))
```

```
## [1] "RMSE for closing price: 6.40087926588651"
```

```r
print(paste("MAE for closing price:", mae))
```

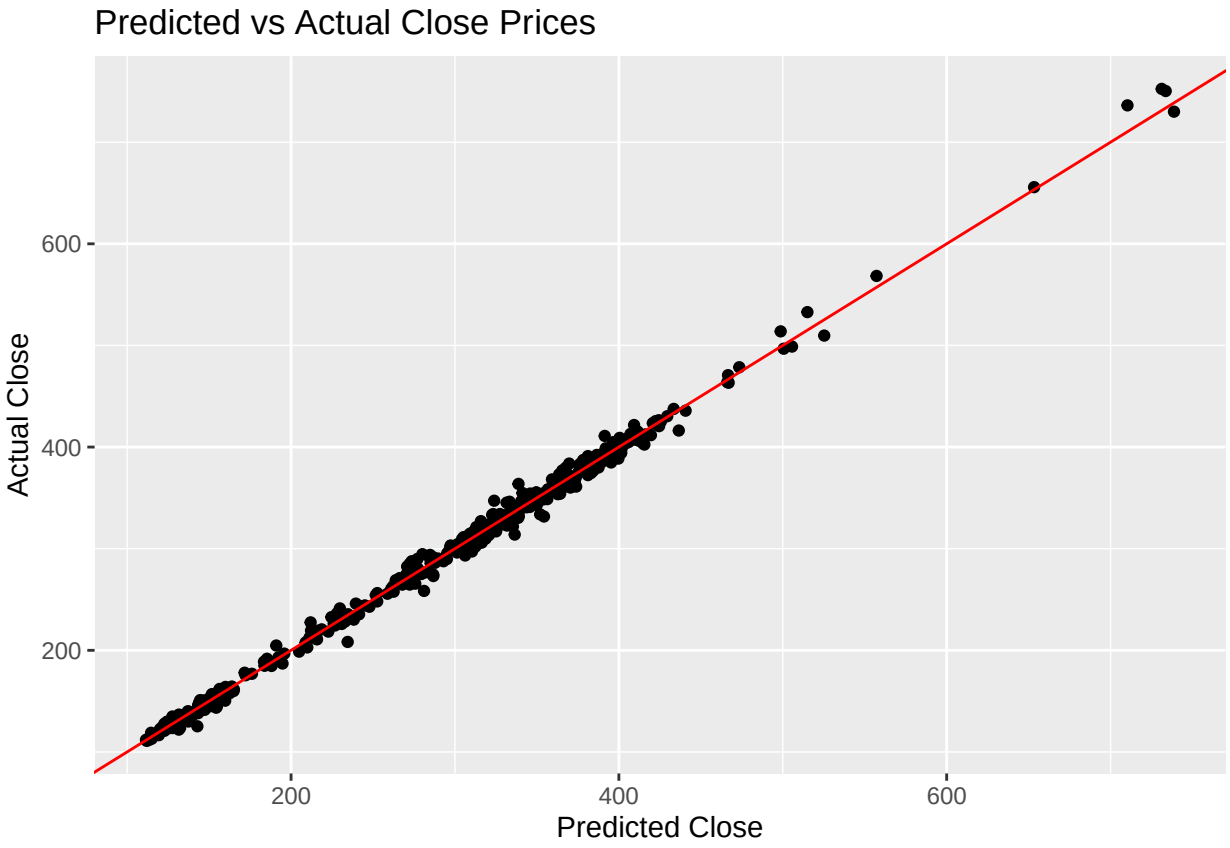
```
## [1] "MAE for closing price: 4.60259732369209"
```

```r
print(paste("R-squared for closing price:", r_squared))
```

```
## [1] "R-squared for closing price: 0.996854728114382"
```

Plot actual vs predicted close prices

```r
ggplot(data = test, aes(x = close_predictions, y = Close)) + geom_point() + geom_abline(slope = 1, inte
```



Train the SVM model using the 'open' and 'close' variables as predictors to see if it can predict price going up or down

***Not a very good model, could keep tweaking but would rather try others**

***Tried to optimize, not very successful**

```
train_control <- trainControl(method = "cv", number = 10)

tune_grid <- expand.grid(C = seq(0.01, 1, by = 0.1))
```

***Cross-validation before it was guessing at .5, randomly, .5 after**

SVM Model

```
svm_model <- train(target ~ Open + Prev.Close, data = train, method = "svmLinear", trControl = train_c
```

Make predictions on the test set using the trained model

```
predictions <- predict(svm_model, newdata = test)
```

Evaluate the model's performance using a confusion matrix

```
confusion_matrix <- confusionMatrix(predictions, test$target)
print(confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down  Up
##       Down  246 246
##       Up      0   0
##
##                Accuracy : 0.5
##                  95% CI : (0.4549, 0.5451)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.518
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
```

```
##
##             Sensitivity : 1.0
##             Specificity : 0.0
##          Pos Pred Value : 0.5
##          Neg Pred Value : NaN
##              Prevalence : 0.5
##          Detection Rate : 0.5
##    Detection Prevalence : 1.0
##       Balanced Accuracy : 0.5
##
##        'Positive' Class : Down
##
```