Homework Assignment4 Stats 123

Group F

2023-03-30

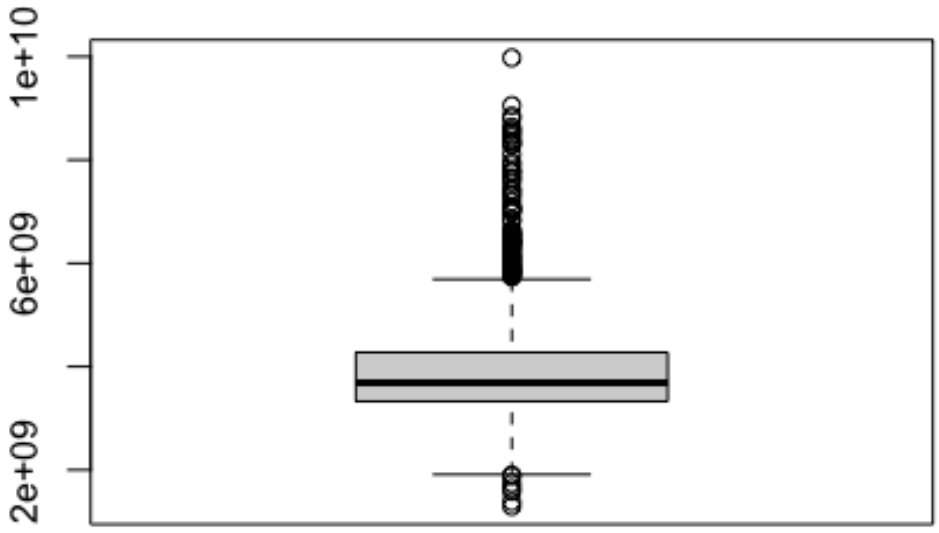
```
#Load the dataset in R and preprocess the data by removing any missing values
and normalizing the data.Data Cleaning and Preprocessing: Load the dataset
and remove any missing or irrelevant data. Preprocess the data by converting
any non numeric values to numeric.

# Library Loading
# Load the necessary libraries
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.1     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(lubridate)
library(dplyr)

# Data Loading and Formatting
# Load read the dataset GSPC.csv into a dataframe called data.
data <- read.csv("/Users/itagakikouki/stat123/project/^GSPC.csv")

# Print some information about the dataframe.
head(data)
```

```
##         Date    Open    High     Low   Close Adj.Close     Volume
## 1 2016-07-05 2095.05 2095.05 2080.86 2088.55   2088.55 3658380000
## 2 2016-07-06 2084.43 2100.72 2074.02 2099.73   2099.73 3909380000
## 3 2016-07-07 2100.42 2109.08 2089.39 2097.90   2097.90 3604550000
## 4 2016-07-08 2106.97 2131.71 2106.97 2129.90   2129.90 3607500000
## 5 2016-07-11 2131.72 2143.16 2131.72 2137.16   2137.16 3253340000
## 6 2016-07-12 2139.50 2155.40 2139.50 2152.14   2152.14 4097820000
```

```
dim(data)

## [1] 1274    7

# Convert the date column to the Date format
data$Date <- ymd(data$Date)

# Create new columns, Previous days closing price and price difference
data <- data %>% mutate(prev_close = lag(Adj.Close, default =
first(Adj.Close)))
data <- data %>% mutate(price_dif = Adj.Close - lag(Adj.Close, default =
first(Adj.Close)))

# Data Cleaning
# Take all the NA values out of the dataframe.
data <- drop_na(data)

# Boxplot for volume (before we do outlier clean).
boxplot(data$Volume)
```

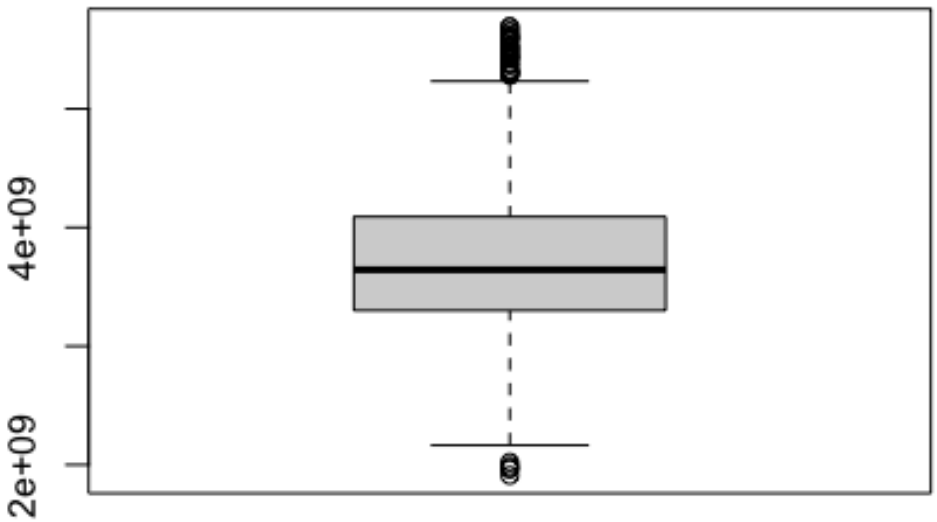



```
# Here we clean the outliers from volume.
data <- data[!data$Volume %in% boxplot(data$Volume, plot = FALSE)$out, ]
```

```
# Here we print the dimensions of the dataframe.
dim(data)
```

```
## [1] 1181    9
```

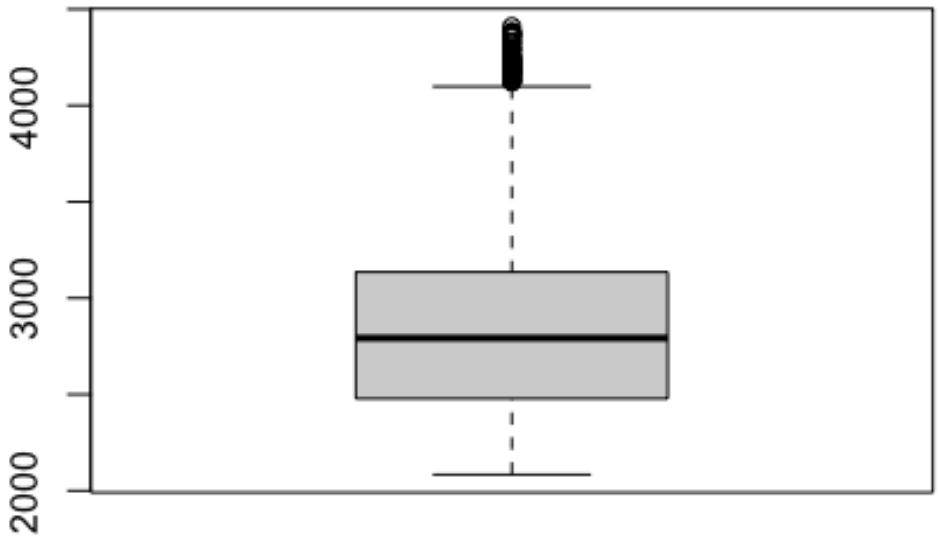
```
# Output a boxplot of volume after we have taken out the outliers.
boxplot(data$Volume)
```




```
# Boxplot for open (before we do outlier clean)
boxplot(data$Open)
# Here we clean the outlier values from our Open column.
data <- data[!data$Open %in% boxplot(data$Open, plot = FALSE), ]
# Print the dimensions of the dataframe after.
dim(data)
```

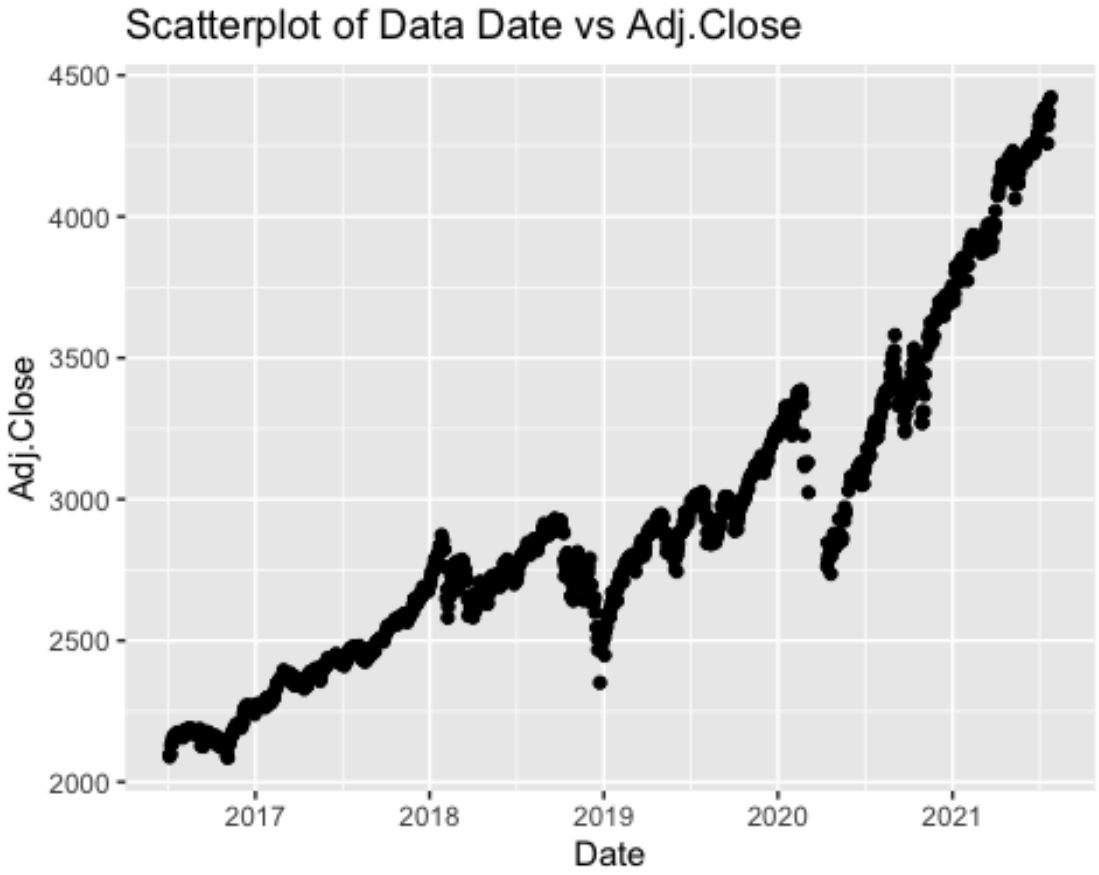
```
## [1] 1181    9
```

```
# Boxplot of the Open column after we have cleaned the outliers from it.
boxplot(data$Open)
```

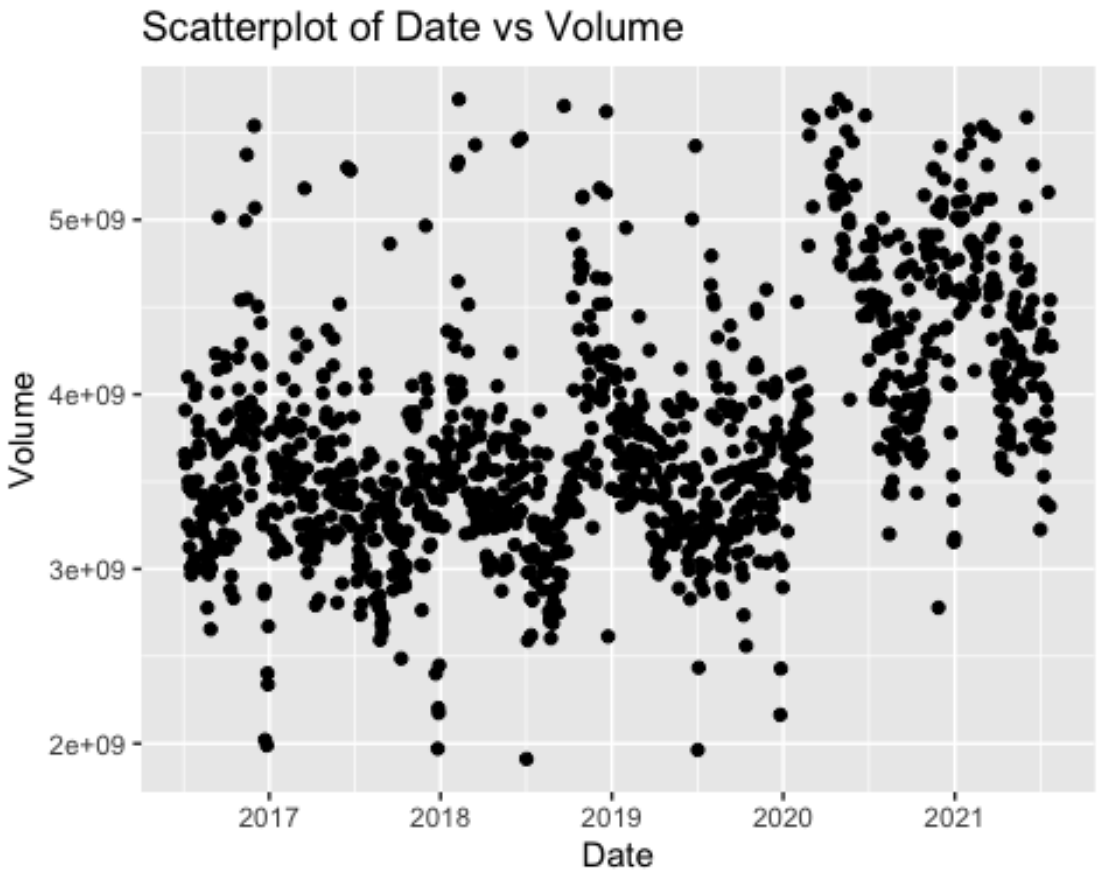



```
# There is a duplicate column, "Close" so we will remove it because it has
the same values listed in Adj.Close
data <- data[,c("Date","Open","High","Low",
"Adj.Close","Volume","prev_close","price_dif")]
head(data)

##         Date    Open    High     Low Adj.Close     Volume prev_close
price_dif
## 1 2016-07-05 2095.05 2095.05 2080.86   2088.55 3658380000    2088.55
0.000000
## 2 2016-07-06 2084.43 2100.72 2074.02   2099.73 3909380000    2088.55
11.179931
## 3 2016-07-07 2100.42 2109.08 2089.39   2097.90 3604550000    2099.73 -
1.830078
## 4 2016-07-08 2106.97 2131.71 2106.97   2129.90 3607500000    2097.90
32.000000
## 5 2016-07-11 2131.72 2143.16 2131.72   2137.16 3253340000    2129.90
7.260010
## 6 2016-07-12 2139.50 2155.40 2139.50   2152.14 4097820000    2137.16
14.979981

# Data Plot graphs
# Plot a scatterplot of Date vs Adj.Close.
```

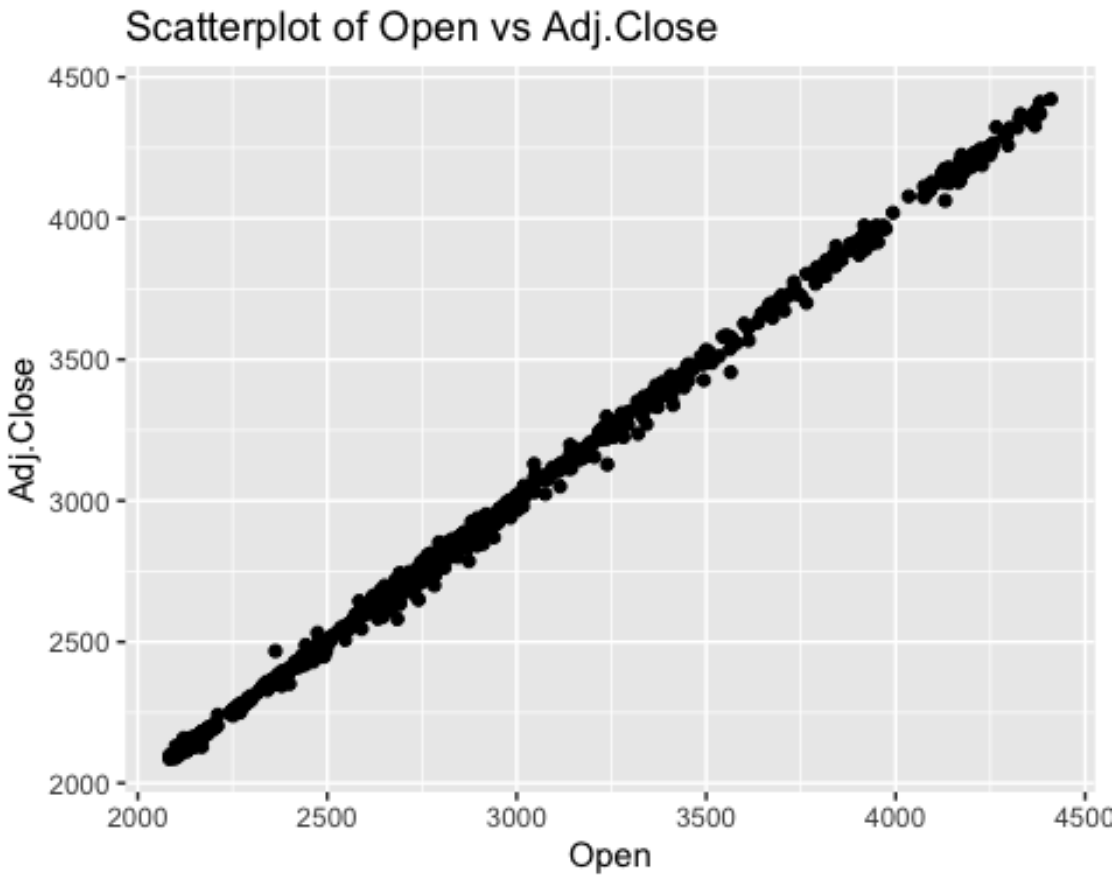
```
ggplot(data, aes(x = Date, y = Adj.Close)) + geom_point() + labs(x = "Date",
y = "Adj.Close", title = "Scatterplot of Data Date vs Adj.Close")
```




```
# Plot a scatterplot of Date vs Volume.
ggplot(data, aes(x = Date, y = Volume)) + geom_point() + labs(x = "Date", y =
"Volume", title = "Scatterplot of Date vs Volume")
```




```
# Plot a scatterplot of Open vs Adj.Close.
ggplot(data, aes(x = Open, y = Adj.Close)) + geom_point() + labs(x = "Open",
y = "Adj.Close", title = "Scatterplot of Open vs Adj.Close")
```




```
# Dataset Analysis
cor(data$Open, data$Adj.Close)

## [1] 0.9992989

# Analyze the data using the str() function.
str(data)

## 'data.frame':    1181 obs. of  8 variables:
##  $ Date      : Date, format: "2016-07-05" "2016-07-06" ...
##  $ Open      : num  2095 2084 2100 2107 2132 ...
##  $ High      : num  2095 2101 2109 2132 2143 ...
##  $ Low       : num  2081 2074 2089 2107 2132 ...
##  $ Adj.Close : num  2089 2100 2098 2130 2137 ...
##  $ Volume    : num  3.66e+09 3.91e+09 3.60e+09 3.61e+09 3.25e+09 ...
##  $ prev_close: num  2089 2089 2100 2098 2130 ...
##  $ price_dif : num  0 11.18 -1.83 32 7.26 ...

# Analyze the data using the summary() function.
summary(data)

##       Date                 Open           High           Low
##  Min.   :2016-07-05   Min.   :2084   Min.   :2095   Min.   :2074
##  1st Qu.:2017-09-12   1st Qu.:2478   1st Qu.:2489   1st Qu.:2472
```

```
##  Median :2018-11-15   Median :2792   Median :2805   Median :2779
##  Mean   :2018-12-09   Mean   :2889   Mean   :2902   Mean   :2875
##  3rd Qu.:2020-01-31   3rd Qu.:3135   3rd Qu.:3143   3rd Qu.:3116
##  Max.   :2021-07-26   Max.   :4410   Max.   :4423   Max.   :4405
##    Adj.Close        Volume            prev_close     price_dif
##  Min.   :2085   Min.   :1.911e+09   Min.   :2085   Min.   :-125.780
##  1st Qu.:2478   1st Qu.:3.301e+09   1st Qu.:2477   1st Qu.:  -6.920
##  Median :2794   Median :3.644e+09   Median :2793   Median :   2.120
##  Mean   :2890   Mean   :3.755e+09   Mean   :2887   Mean   :   2.239
##  3rd Qu.:3130   3rd Qu.:4.091e+09   3rd Qu.:3128   3rd Qu.:  14.340
##  Max.   :4422   Max.   :5.691e+09   Max.   :4412   Max.   : 126.750

# Train and Test Data Splitting
# Split training and test sets use sample() and subset()
set.seed(123)
train_index <- sample(nrow(data), size = round(0.8*nrow(data)), replace =
FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
# Output a sample of values from train_data and test_data.
head(train_data)

##            Date    Open    High     Low Adj.Close     Volume prev_close
## 423  2018-03-08 2732.75 2740.45 2722.65   2738.97 3206040000    2726.80
## 471  2018-05-16 2712.62 2727.76 2712.17   2722.46 3248480000    2711.45
## 183  2017-03-24 2350.42 2356.22 2335.74   2343.98 2978530000    2345.96
## 534  2018-08-15 2827.95 2827.95 2802.49   2818.37 3656680000    2839.96
## 199  2017-04-18 2342.53 2348.35 2334.54   2342.19 3272210000    2349.01
## 1002 2020-06-25 3046.60 3086.25 3024.01   3083.76 4847690000    3050.33
##       price_dif
## 423   12.169922
## 471   11.010010
## 183   -1.979981
## 534  -21.589844
## 199   -6.820069
## 1002  33.429932

head(test_data)

##          Date    Open    High     Low Adj.Close     Volume prev_close
price_dif
## 1  2016-07-05 2095.05 2095.05 2080.86   2088.55 3658380000    2088.55
0.000000
## 7  2016-07-13 2153.81 2156.45 2146.21   2152.43 3502320000    2152.14
0.290039
## 12 2016-07-20 2166.10 2175.63 2164.89   2173.02 3211860000    2163.78
9.239991
## 14 2016-07-22 2166.47 2175.11 2163.24   2175.03 3023280000    2165.17
9.860107
## 15 2016-07-25 2173.71 2173.71 2161.95   2168.48 3057240000    2175.03 -
6.550049
```

```
## 22 2016-08-03 2156.81 2163.79 2152.56   2163.79 3786530000    2157.03
6.760010

# Model Creation and Refinement
# Build a linear regression model to predict price difference
model <- lm(price_dif ~ Open + High + Low + Volume, data = train_data)
model

##
## Call:
## lm(formula = price_dif ~ Open + High + Low + Volume, data = train_data)
##
## Coefficients:
## (Intercept)         Open         High          Low       Volume
##  -2.475e+00   -1.868e+00    9.301e-01    9.396e-01    2.640e-10

# Use the summary() function to output information about the model.
summary(model)

##
## Call:
## lm(formula = price_dif ~ Open + High + Low + Volume, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -77.406  -5.557  -0.016   5.712  69.623
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.475e+00  2.874e+00  -0.861    0.389
## Open        -1.868e+00  3.635e-02 -51.389   <2e-16 ***
## High         9.301e-01  3.546e-02  26.227   <2e-16 ***
## Low          9.396e-01  2.861e-02  32.840   <2e-16 ***
## Volume       2.640e-10  7.612e-10   0.347    0.729
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.4 on 940 degrees of freedom
## Multiple R-squared:  0.753,  Adjusted R-squared:  0.7519
## F-statistic: 716.3 on 4 and 940 DF,  p-value: < 2.2e-16

# We see Volume is not significant so we remake the model without it.

# Build a linear regression model to predict price difference without
# Volume.
model <- lm(price_dif ~ Open + High + Low, data = train_data)
model

##
## Call:
## lm(formula = price_dif ~ Open + High + Low, data = train_data)
```

```
##
## Coefficients:
## (Intercept)         Open         High          Low
##     -1.8234      -1.8679       0.9343       0.9352
```

```
# Use the summary() function to output information about the model.
summary(model)
```

```
##
## Call:
## lm(formula = price_dif ~ Open + High + Low, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -77.230  -5.569   0.013   5.703  69.614
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.82336    2.17372  -0.839    0.402
## Open        -1.86788    0.03633 -51.414   <2e-16 ***
## High         0.93429    0.03329  28.062   <2e-16 ***
## Low          0.93522    0.02571  36.380   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.39 on 941 degrees of freedom
## Multiple R-squared:  0.7529, Adjusted R-squared:  0.7521
## F-statistic: 955.9 on 3 and 941 DF,  p-value: < 2.2e-16
```

```
# Model Testing
# Predict the price difference for the test set
pred_diff <- predict(model, newdata = test_data)
head(pred_diff)
```

```
##          1          7         12         14         15         22
## -11.675754  -2.950507   9.482660   6.763140  -9.274894   4.242368
```

```
# Calculate the predicted closing price for the test set
pred_close <- test_data$Adj.Close + pred_diff
head(pred_close)
```

```
##        1        7       12       14       15       22
## 2076.874 2149.479 2182.503 2181.793 2159.205 2168.032
```

```
# Model Evaluation
# Calculate the mean absolute error
mae <- mean(abs(test_data$Adj.Close - pred_close))
mae
```

```
## [1] 15.98838
```

```r
# Calculate the mean percentage error
mpe <- mean((pred_close - test_data$Adj.Close) / test_data$Adj.Close) * 100
mpe
```

```
## [1] 0.04031531
```

```r
# Here we classify all the predictions of our model as Increase or Decrease
# and save all of them to predicted.
predicted <- ifelse(pred_diff > 0, "Increase", "Decrease")
# Here we classify all the predictions of our model as Increase or Decrease
# and save all of them to actual.
actual <- ifelse(test_data$price_dif > 0, "Increase","Decrease")
# Output a sample of the values in predicted and actual.
head(predicted)
```

```
##          1          7         12         14         15         22
## "Decrease" "Decrease" "Increase" "Increase" "Decrease" "Increase"
```

```r
head(actual)
```

```
## [1] "Decrease" "Increase" "Increase" "Increase" "Decrease" "Increase"
```

```r
# Create a confusion matrix off of predicted and actual using the table
# function.
confusion_matrix <- table(predicted, actual)
# Output the confusion matrix we created.
confusion_matrix
```

```
##           actual
## predicted  Decrease Increase
##   Decrease       79       20
##   Increase       24      113
```

```r
# Evaluation Metrics
# Calculate the accuracy of the model using the confusion matrix.
accuracy <- sum((confusion_matrix[1,1]) + (confusion_matrix[2,2])) /
(sum(confusion_matrix))
# Output the calculated accuracy.
accuracy
```

```
## [1] 0.8135593
```

```r
# Calculate the sensitivity of the predicted model using the confusion
matrix.
sensitivity <- confusion_matrix[1,1]/ (confusion_matrix[1,1] +
confusion_matrix[2,1])
# Output the calculated sensitivity.
sensitivity
```

```
## [1] 0.7669903
```

```r
# Calculate the precision of the predicted model using the confusion matrix.
precision <- confusion_matrix[1,1]/(confusion_matrix[1,1] +
confusion_matrix[1,2])
# Output the calculated precision.
precision
```

```
## [1] 0.7979798
```

```r
# Calculate recall of the predicted model using the confusion matrix.
recall <- confusion_matrix[1,1] / (confusion_matrix[1,1] +
confusion_matrix[2,1])
# Output the calculated recall.
recall
```

```
## [1] 0.7669903
```

```r
# Use the calculated recall and precision to calculate the F1 Score for our
# model.
f1_score <- 2 * precision * recall / (precision + recall)
# Output the calculated F1 Score.
f1_score
```

```
## [1] 0.7821782
```

```r
# Final Model Evaluation
# Print the evaluation metrics
cat("Sensitivity:", sensitivity, "\n")
```

```
## Sensitivity: 0.7669903
```

```r
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.8135593
```

```r
cat("Precision:", precision, "\n")
```

```
## Precision: 0.7979798
```

```r
cat("Recall:", recall, "\n")
```

```
## Recall: 0.7669903
```

```r
cat("F1-score:", f1_score, "\n")
```

```
## F1-score: 0.7821782
```

#This R code generates a model that predicts future prices of the S&P 500 Index using historical data. The script is divided into several segments.

###Firstly the necessary libraries for the script are loaded, including tidyverse, lubridate, and dplyr.

###Secondly the script loads and formats the data. The script reads in the GSPC.csv dataset, and converts all the values in the Date column to the date type. It then creates two new columns, prev_close and price_dif, based on data from the dataframe.

###Thirdly the script performs data cleaning. It removes all the NA values, creates a boxplot of the Volume column to visualize outliers, and removes the outliers from the Volume and Open columns. The Close column is removed from the dataframe since it has the same values as Adj.Close.

###Fourthly the script plots three separate scatterplots of the columns in the dataframe, namely Date vs Adj.Close, Date vs Volume, and Open vs Adj.Close.

###Fifthly the script analyzes the dataset by calculating the correlation coefficient between the Open and Adj.Close columns of the dataframe, and analyzing the dataframe with the str() and summary() functions.

###Sixthly the script splits the data into a random 80% for training and 20% for testing. It then outputs the first few values from each of them.

###Seventhly the script creates a linear regression model based on the Open, High, Low, and Volume columns of the dataframe. It prints out a summary of the model, and removes Volume since it is not significant. It then creates a new model without Volume, leaving the intercept as is. The new model shows that the most important factors affecting stock price are Open, High, and Low.

###Eighthly the script tests the model on the test data using the predict() function to predict the difference in stock price. It then uses this predicted difference to calculate the model's predicted closing price of the stocks in the test set.

###Ninthly the script evaluates the model. It calculates and outputs the mean absolute error and the mean percentage error of the model compared with the actual values for the test data. It also classifies the differences in the predicted differences and the actual differences as either Increase or Decrease depending on what they say the stock will do, saving the classifications. It outputs a few values of the classifications and creates a confusion matrix using the table() function.

###Tenthly the script calculates accuracy, sensitivity, recall, precision, and F1 score using the confusion matrix.

###Finally the script outputs each of the final evaluation metrics.