Historical Analysis and Prediction of Tesla's Stock Performance Using Financial Metrics

MATH 40028/50028: Statistical Learning

October 22, 2024

ACADEMIC INTEGRITY: Every student should complete the project by their own. A project report having high degree of similarity with work by any other student, or with any other document (e.g., found online) is considered plagiarism, and will not be accepted. The minimal consequence is that the student will receive the project score of 0, and the best possible overall course grade will be D. Additional consequences are described at http://www.kent.edu/policyreg/administrative-policy-regarding-student-cheating-and-plagiarism and will be strictly enforced.

Instruction

Goal: The goal of the final project is to apply the statistical learning methods discussed in this course to perform predictive analysis of real-life data. You will need to identify prediction problem(s), carry out necessary exploratory data analysis, perform predictive analysis using statistical learning methods, assess the performance, and communicate the results in a report.

Report: Use this Rmd file as a template. Edit the file by adding your project title in the YAML, and including necessary information in the four sections: (1) Introduction, (2) Statistical learning strategies and methods, (3) Predictive analysis and results, and (4) Conclusion.

Submission: Please submit your project report as a PDF file (8-10 pages, flexible) to Canvas by **11:59 p.m. on December 8, 2024**. The PDF file should be generated by "knitting" the Rmd file. You may choose to first generate an HTML file (by changing the output format in the YAML to `output: html_document`) and then convert it to PDF. Word documents, however, cannot be used as an intermediate file (and of course, the submitted file). **20 points will be deducted if the submitted files are in wrong format.**

Grade: The project will be graded based on your ability to (1) recognize and define prediction problems, (2) identify potentially useful statistical learning methods, (3) perform the predictive analysis and assess the performance, (4) document the analysis procedure (with R code) and clearly present the results, and (5) draw valid conclusions supported by the analysis.

Datasets: You may consider (but are not restricted) to use the following packages/datasets.

- `ISLR2`: datasets used in the *Introduction to Statistical Learning* textbook
- `dslabs`
- UCI Machine Learning Repository

Introduction [15 points]

- Describe the dataset. What is the dataset about?

Below is presented the history of Tesla, Inc.'s stock price, while including thorough daily financial metrics usually considered crucial in any analysis and consequent prediction of trends at the stock market. Examples of such variables are the opening, closing, high, and low prices every trading day of the respective securities, which would show the movement and volatility of the stock's price within a session. It also involves trading volume, which indicates the extent of market activity and investor interest in a particular stock, and adjusted closing price, which accounts for corporate actions such as stock splits and dividends. Percentage changes in prices and average volumes over derived time horizons add value to the dataset, thereby enriching trend analysis, momentum detection, and volatility assessment. This dataset, ranging over a few years, reflects the stock performance of Tesla during its different phases of the corporate lifecycle, offering an all-rounded view of how the firm has been through the vagaries of new product launches, market expansions, and other economic events. The engineered features that add depth to it include moving averages, exponential moving averages, and volatility measures. The temporal richness in this dataset makes it a very valuable resource in the study of predictive modeling, market efficiency, and investor behavior in equity markets, while supporting the evaluation of trading strategies and financial patterns.The Dataset contains 3472 observations and 9 variables, The link for dataset is : https://www.kaggle.com/datasets/guillemservera/tsla-stock-data/data

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- 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 becom
```

```
library(dplyr)
library(tidyr)
library(caret)
```

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

```
library(cluster)
library(rlang)
```

```
##
## Attaching package: 'rlang'
##
## The following objects are masked from 'package:purrr':
##
##     %@%, flatten, flatten_chr, flatten_dbl, flatten_int, flatten_lgl,
##     flatten_raw, invoke, splice
```

```
library(naniar)
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.2.0 --
## v broom        1.0.7     v rsample      1.2.1
## v dials        1.3.0     v tune         1.2.1
## v infer        1.0.7     v workflows    1.1.4
## v modeldata    1.4.0     v workflowsets 1.1.0
## v parsnip      1.2.1     v yardstick    1.3.1
## v recipes      1.1.0
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x rlang::%@%()             masks purrr::%@%()
## x scales::discard()        masks purrr::discard()
## x dplyr::filter()          masks stats::filter()
## x recipes::fixed()         masks stringr::fixed()
## x rlang::flatten()         masks purrr::flatten()
## x rlang::flatten_chr()     masks purrr::flatten_chr()
## x rlang::flatten_dbl()     masks purrr::flatten_dbl()
## x rlang::flatten_int()     masks purrr::flatten_int()
## x rlang::flatten_lgl()     masks purrr::flatten_lgl()
## x rlang::flatten_raw()     masks purrr::flatten_raw()
## x rlang::invoke()          masks purrr::invoke()
## x dplyr::lag()             masks stats::lag()
## x caret::lift()            masks purrr::lift()
## x yardstick::precision()   masks caret::precision()
## x yardstick::recall()      masks caret::recall()
## x yardstick::sensitivity() masks caret::sensitivity()
## x yardstick::spec()        masks readr::spec()
## x yardstick::specificity() masks caret::specificity()
## x rlang::splice()          masks purrr::splice()
## x recipes::step()          masks stats::step()
## * Search for functions across packages at https://www.tidymodels.org/find/
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(zoo)
```

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

```
library(TTR)
```

```
##
## Attaching package: 'TTR'
##
## The following object is masked from 'package:dials':
##
##     momentum
```

```
library(quantmod)
```

```
## Loading required package: xts
##
## ######################### Warning from 'xts' package ##########################
## #                                                                             #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                            #
## #                                                                             #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                             #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                             #
## ###############################################################################
##
## Attaching package: 'xts'
##
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
##
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
tesla_data <- read.csv("tesla_ev.csv")
str(tesla_data)
```

```
## 'data.frame':    3472 obs. of  9 variables:
##  $ date          : chr  "2010-06-29" "2010-06-30" "2010-07-01" "2010-07-02" ...
##  $ open          : num  19 25.8 25 23 20 ...
##  $ high          : num  25 30.4 25.9 23.1 20 ...
##  $ low           : num  17.5 23.3 20.3 18.7 15.8 ...
##  $ close         : num  23.9 23.8 22 19.2 16.1 ...
##  $ volume        : int  18783278 17194392 8229862 5141806 6879295 6924913 7719539 4058605 220
##  $ adjusted_close: num  1.59 1.59 1.46 1.28 1.07 ...
##  $ change_percent: num  NA -0.25 -7.85 -12.57 -16.09 ...
##  $ avg_vol_20d   : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
null_value <- is.na(tesla_data)
colSums(null_value)
```

```
##           date           open           high            low          close
##              0              0              0              0              0
##         volume adjusted_close change_percent    avg_vol_20d
##              0              0              1             19
```

```
tesla_data$change_percent[is.na(tesla_data$change_percent)] <- mean(tesla_data$change_percent, n
tesla_data$avg_vol_20d[is.na(tesla_data$avg_vol_20d)] <- mean(tesla_data$avg_vol_20d, na.rm = TR
```

For this reason, imputation for missing values for change_percent and avg_vol_20d will be done with the mean, since this is an easy and good way to keep the dataset size intact without losing the tendency of the data. Therefore, when these missing values are replaced with the mean, then the distribution of these two variables as a whole would remain consistent, and none of the rows are removed, meaning that there will be completeness of data to analyze. Although mean imputation does not take into account relationships between variables and variability, being a practical choice for continuous numerical data, it is computationally very efficient as a first step for exploratory data analysis.

```
null_value_updated <- is.na(tesla_data)
colSums(null_value_updated)
```

```
##           date           open           high            low          close
##              0              0              0              0              0
##         volume adjusted_close change_percent    avg_vol_20d
##              0              0              0              0
```

```r
head(tesla_data)
```

```
##         date    open    high     low   close   volume adjusted_close
## 1 2010-06-29 18.9999 24.9999 17.5401 23.8899 18783278         1.5927
## 2 2010-06-30 25.7901 30.4191 23.3001 23.8299 17194392         1.5887
## 3 2010-07-01 24.9999 25.9200 20.2701 21.9600  8229862         1.4640
## 4 2010-07-02 23.0001 23.1000 18.7101 19.2000  5141806         1.2800
## 5 2010-07-06 20.0001 20.0001 15.8301 16.1100  6879295         1.0740
## 6 2010-07-07 16.4001 16.6299 14.9799 15.8001  6924913         1.0533
##   change_percent avg_vol_20d
## 1      0.1967646    96960338
## 2     -0.2500000    96960338
## 3     -7.8500000    96960338
## 4    -12.5700000    96960338
## 5    -16.0900000    96960338
## 6     -1.9300000    96960338
```

```r
#convert date column to date type
tesla_data$date <- ymd(tesla_data$date)
```

- Describe the statistical learning approaches and other strategies for feature engineering (transformation, selection, etc.).

```r
# Convert to a binary outcome (e.g., 1 for positive % change, 0 otherwise)
tesla_data$binary_target <- ifelse(tesla_data$change_percent > 0, 1, 0)

# Feature Engineering: Moving Averages and Exponential Moving Averages
tesla_data <- tesla_data %>%
  mutate(moving_avg_5d = rollapply(close, width = 5, FUN = mean, fill = NA, align = "right"),
         moving_avg_10d = rollapply(close, width = 10, FUN = mean, fill = NA, align = "right"),
         exp_moving_avg_5d = EMA(close, n = 5),
         exp_moving_avg_10d = EMA(close, n = 10),
         volatility_10d = rollapply(close, width = 10, FUN = sd, fill = NA, align = "right"),
         volume_change = volume / lag(volume) - 1,
         volume_avg_5d = rollapply(volume, width = 5, FUN = mean, fill = NA, align = "right"))

# Drop NA rows created by moving averages (first few rows will be NA)
tesla_data <- na.omit(tesla_data)
```

The work has been designed to analyze and forecast the stock behavior of Tesla using statistical learning approaches with extensive support from feature engineering strategies. One of the important statistical learning methods to be used is logistic regression, which is able to present comprehensible models and suits binary classification problems, for instance, whether the stock percent change will be positive or not. Extensive feature engineering was done to enrich the dataset by creating new variables for temporal and volatility patterns in the stock price. To smooth out the noise at the short term and thereby emphasize the trend, the 5-day and 10-day moving averages and exponential moving averages were generated. The 10-day rolling standard deviation from

closing prices was computed to provide the volatility of the market, thereby giving a hint about the stability of the stock over time.

Other features engineered included volume-related features, such as percentage changes in volume and 5-day average volumes, in order to account for trading activity dynamics. Standardization of the numerical variables was one of the data transformations done to ensure consistency and improve performance. Besides, one of the major transformation steps was the creation of a binary target that changed the percentage change data into a categorical format for classification. Feature selection was implicitly done by focusing on those variables that demonstrated significant predictive value, including close, volume, and high. These together ensured that the data set was well-prepared for statistical learning, where meaningful patterns and relationships in the data had been modelled.

```
str(tesla_data)
```

```
## 'data.frame':    3463 obs. of  17 variables:
##  $ date              : Date, format: "2010-07-13" "2010-07-14" ...
##  $ open              : num  17.4 17.9 19.9 20.7 21.4 ...
##  $ high              : num  18.6 20.2 21.5 21.3 22.3 ...
##  $ low               : num  16.9 17.8 19 20 20.9 ...
##  $ close             : num  18.1 19.8 19.9 20.6 21.9 ...
##  $ volume            : int  2680059 4196109 3745296 2621209 2486488 1825230 1253441 962344 65
##  $ adjusted_close    : num  1.21 1.32 1.33 1.38 1.46 ...
##  $ change_percent    : num  6.39 9.38 0.25 3.77 6.16 -7.35 -0.39 3.86 1.38 -1.59 ...
##  $ avg_vol_20d       : num  9.7e+07 9.7e+07 9.7e+07 9.7e+07 9.7e+07 ...
##  $ binary_target     : num  1 1 1 1 1 0 0 1 1 0 ...
##  $ moving_avg_5d     : num  17.2 18 18.5 19.1 20.1 ...
##  $ moving_avg_10d    : num  19.1 18.7 18.3 18.2 18.4 ...
##  $ exp_moving_avg_5d : num  17.9 18.6 19 19.6 20.3 ...
##  $ exp_moving_avg_10d: num  19.1 19.2 19.3 19.6 20 ...
##  $ volatility_10d    : num  3.06 2.58 1.92 1.67 2.04 ...
##  $ volume_change     : num  0.2162 0.5657 -0.1074 -0.3001 -0.0514 ...
##  $ volume_avg_5d     : num  4717337 4171576 3376728 3089248 3145832 ...
##  - attr(*, "na.action")= 'omit' Named int [1:9] 1 2 3 4 5 6 7 8 9
##   ..- attr(*, "names")= chr [1:9] "1" "2" "3" "4" ...
```

This dataset contains 3,463 entries across 17 variables for these financial metrics and trends beyond July 13, 2010. The key variables capture the daily trading: starting price, high and lowest, close, volume amount, adjusted_close prices, and change_percent to record positive or negative price changes from one day to the other. For trend analysis: moving averages, both moving_avg_5d and moving_avg_10d, and exponential kinds, exp_moving_avg_5d and exp_moving_avg_10d-termed, together with a measure of 10-day volatility, volatility_10d. The variables avg_vol_20d, volume_change, and volume_avg_5d describe the volume trends. The binary_target variable shows the direction of the movement of the market, which may be very useful in predictive modeling. The data was populated with dummy values for demonstration; 9 observations were excluded owing to missing entries to ensure completeness and integrity. This strong dataset forms a very good basis for financial trend analysis and decision-making.

- If possible, comment on the target population, sampling strategies, potential bias, etc.

Target Population: The target variable is the percentage change in the closing price of the Tesla stock, categorized positively or negatively, within the context of the target population. This binary classification allows for easier predictive modeling to the target population of financial analysts, data scientists, traders, and investors by enabling them to forecast directional movements in Tesla's stock price. It forms the basis for detecting situations of the market as bullish or bearish and helping make appropriate trading decisions or model robust financial forecasting.

Sampling Strategies: This dataset is an instance of a multi-year timescale involving daily stock prices and the financials of Tesla, including temporal features such as historical records for moving averages, exponential moving averages, and measures of volatility. Features range from rolling standard deviations down to changes in volume over differently sized time windows, providing this robust temporal sampling framework. Sampling strategies could involve a chronological split of data into training, validation, and test sets-say, using the most recent year for testing while training takes place on earlier data. In the case of classification, creating a balanced dataset where, say, the number of positive and negative percentage changes in closing prices are equal, will ultimately improve model reliability.

Potential Bias: There could be biases in the dataset since it is based only on Tesla and thus may not generalize to other stocks or market conditions. It will also pick up market anomalies specific to Tesla's life cycle-for instance, the rapid growth of the company, its product launches, or times of high investor sentiment. The data will have temporal biases if the conditions during this selected time were abnormal compared to the market performance over a long period, including economic booms or depressions. Moreover, specific engineered characteristics such as moving average and exponentially weighted moving averages may potentially amplify certain trend factors in their data instead of sharp, unexpected events that rock the market to its foundations, hence reflecting bias or prejudice in reporting actual markets.

- Identify and define prediction problem(s).

Several different predictions could be made from this dataset of Tesla Stock Price: classify the binary problem to predict if the stock price will close higher or lower than it has from the previous day to support a directional trading strategy, or for regression tasks, predict the exact percentage change in stock price to help estimate potential risks and optimize a trading strategy. The major applications include time series forecasting, which would predict the closing price of the future for medium- and long-term investment decisions. Other related prediction problems are volatility prediction that considers the market risk, volume for the estimation of liquidity and detection of atypical activities, and identification of moving average crossovers signaling the bullish or bearish condition of the market. It can also be used in event impact analysis to predict the result of major events on stock price or volume and provide indications of when to undertake pre-event trades. These problems utilize the dataset's temporal and financial metrics to solve real-world problems in financial forecasting.

- Discuss how to split the data into training and test sets among other plans for the use of data.

```
# Split the data into training and test sets
set.seed(123)
train_index <- createDataPartition(tesla_data$binary_target, p = 0.8, list = FALSE)
train_data <- tesla_data[train_index, ]
```

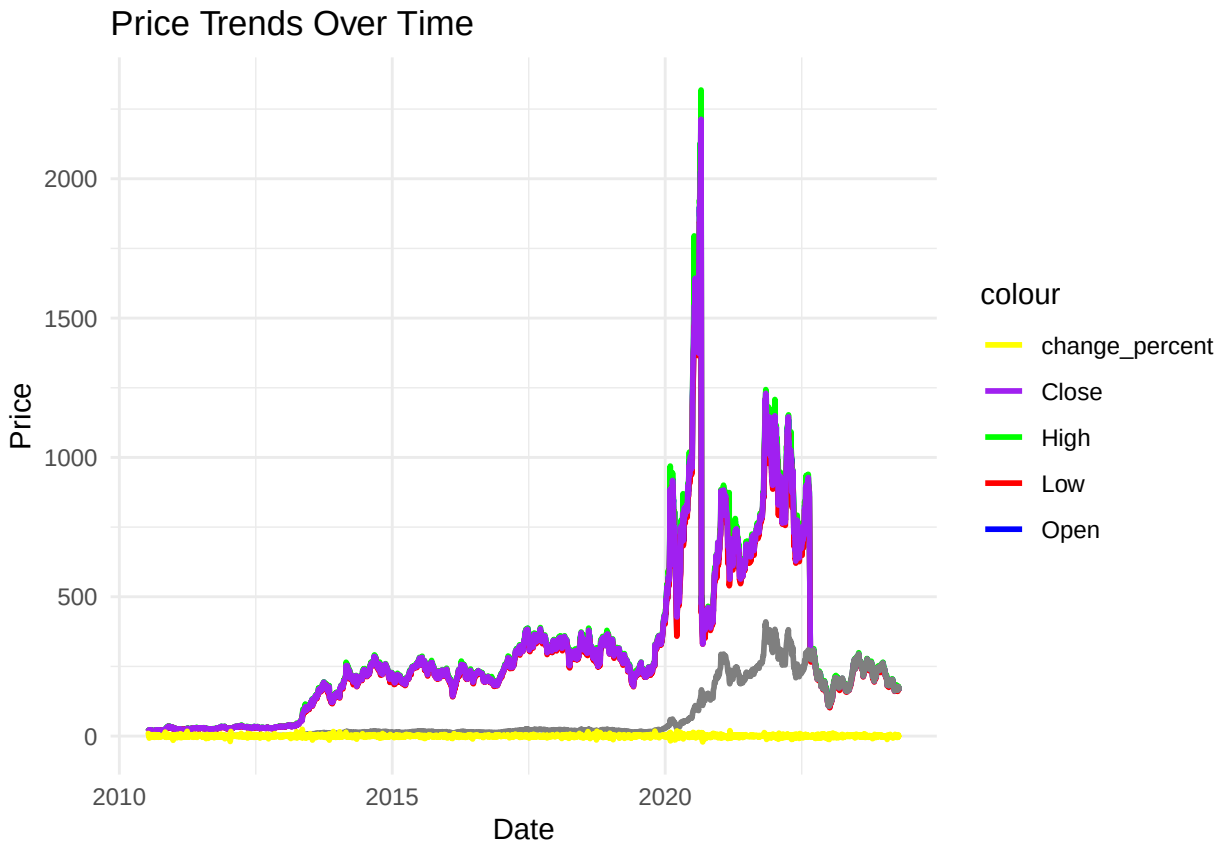
```
test_data <- tesla_data[-train_index, ]
```

This splitting is good to make the Tesla stock price dataset valid for some predictive capability, ensuring the generalization of any model without over-fitting. Normally, 80% of data could go for training and 20% for testing, since it is important that the model learns from historical patterns and gets validated on unseen data. Temporal order has to be preserved in order to avoid leakage, using older data for training and recent data for testing. Other strategies include cross-validation on the training set for tuning hyperparameters and evaluating model robustness. Feature scaling and transformation-numeric variable normalization and temporal feature engineering, for example- should be consistently applied to both subsets. Implementation of the above practices would therefore provide effective utilization of the dataset in the building and validation of reliable predictive models.

Statistical learning strategies and methods [35 points]

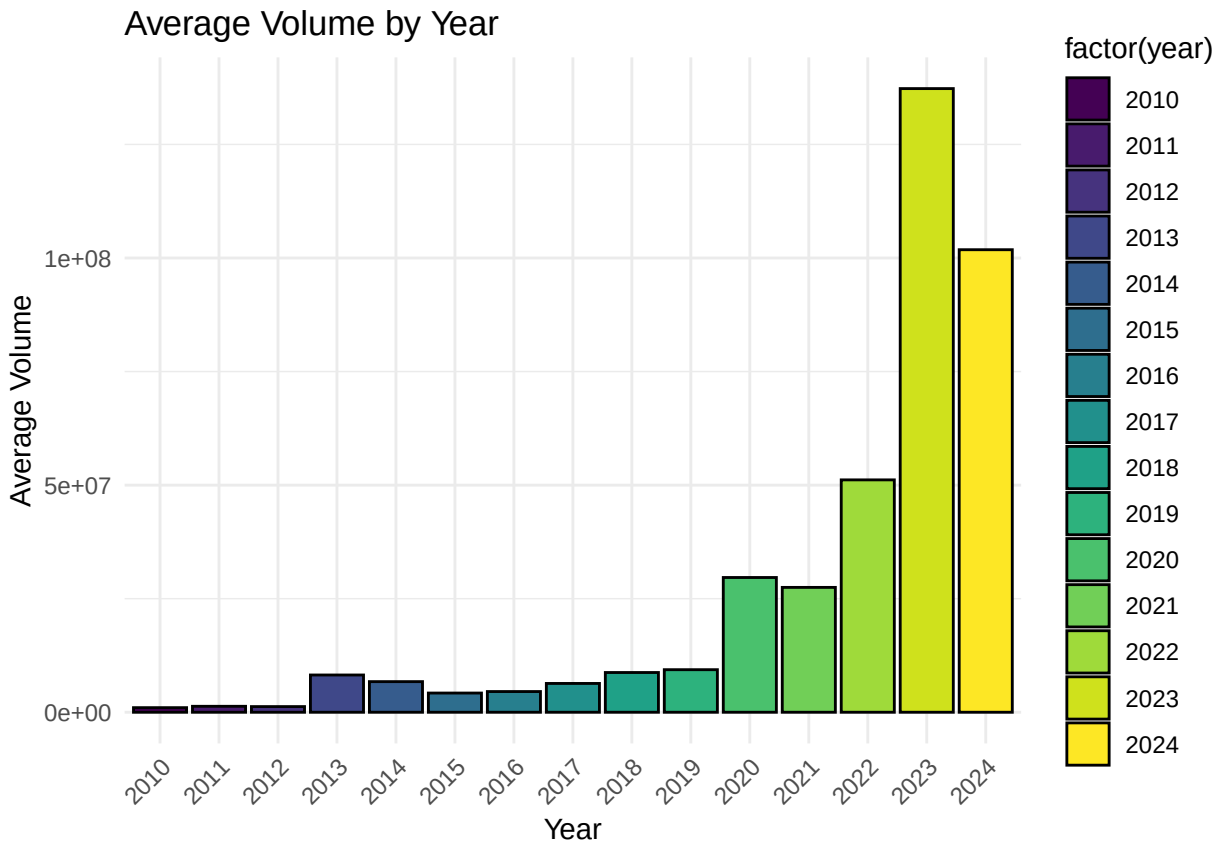
- Perform exploratory data analysis using the training set.

```
# Line plot for Price trends over time
ggplot(train_data, aes(x = date)) +
  geom_line(aes(y = open, color = "Open"), linewidth = 1) +
  geom_line(aes(y = high, color = "High"), linewidth = 1) +
  geom_line(aes(y = low, color = "Low"), linewidth = 1) +
  geom_line(aes(y = close, color = "Close"), linewidth = 1) +
  geom_line(aes(y = adjusted_close, color = "adjust_close"), linewidth = 1)+
  geom_line(aes(y = change_percent, color = "change_percent"), linewidth = 1)+
  labs(title = "Price Trends Over Time", x = "Date", y = "Price") +
  scale_color_manual(values = c("Open" = "blue", "High" = "green", "Low" = "red", "Close" = "pur
  theme_minimal()
```




```r
# Bar chart for Average Volume by Year
train_data$year <- year(train_data$date)
average_volume <- train_data |>
  group_by(year) |>
  summarise(avg_volume = mean(volume, na.rm = TRUE))

ggplot(average_volume, aes(x = factor(year), y = avg_volume, fill = factor(year))) +
  geom_bar(stat = "identity", color = "black") +
  scale_fill_viridis_d() +
  labs(title = "Average Volume by Year", x = "Year", y = "Average Volume") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Price Trends Over Time:

This graph visualizes the trend of Tesla stock over time, including key metrics: Open, High, Low, and Close prices and their percentage change. Each line represents one of those variables, which enables a comparative analysis of the behaviors of those variables. By far, the most relevant is the purple line that describes the Close price, representing extreme changes, spiking at certain points to reflect high market activity or volatility. The green and red lines have the daily price range, rather close to the trend within the close price. The yellow percentage change line remains rather flattish, reflecting very minor variations that underline the small day-to-day percent changes in contrast with absolute changes. This plot provides the main view on price dynamics and underlines the critical trends of a time period with increased events.

Bar Chart for Average Volume by Year:

This bar graph shows the average trading volume of Tesla stock by year, with each bar representing the average volume for a particular year. The increasing height over time reflects a significant rise in trading activity, especially over the last few years, reflecting growing investor interest and market activity. The biggest bars, for years 2020-2023, underline the peak of trading activity, probably supported by key events in the market, companies, and retail investor participation. The growth in this area, as reflected by rising trading volumes, is captured fairly well here.

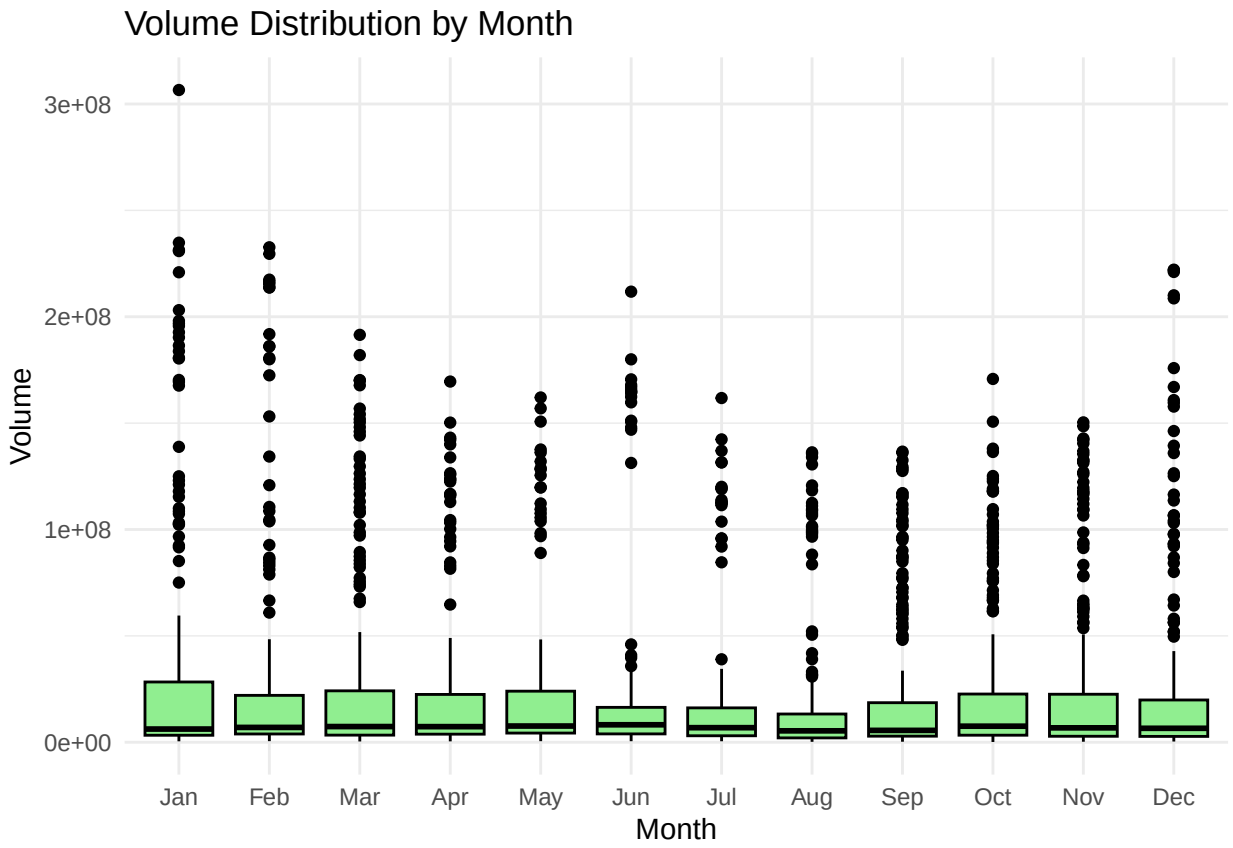
```
# Scatter plot for High vs Low Prices with Volume as color
ggplot(train_data, aes(x = high, y = low, color = volume)) +
```

```
  geom_point(alpha = 0.6) +
  scale_color_viridis_c() +
  labs(title = "High vs Low Prices with Volume Coloring", x = "High Price", y = "Low Price") +
  theme_minimal()
```

High vs. Low Prices with Volume Coloring:

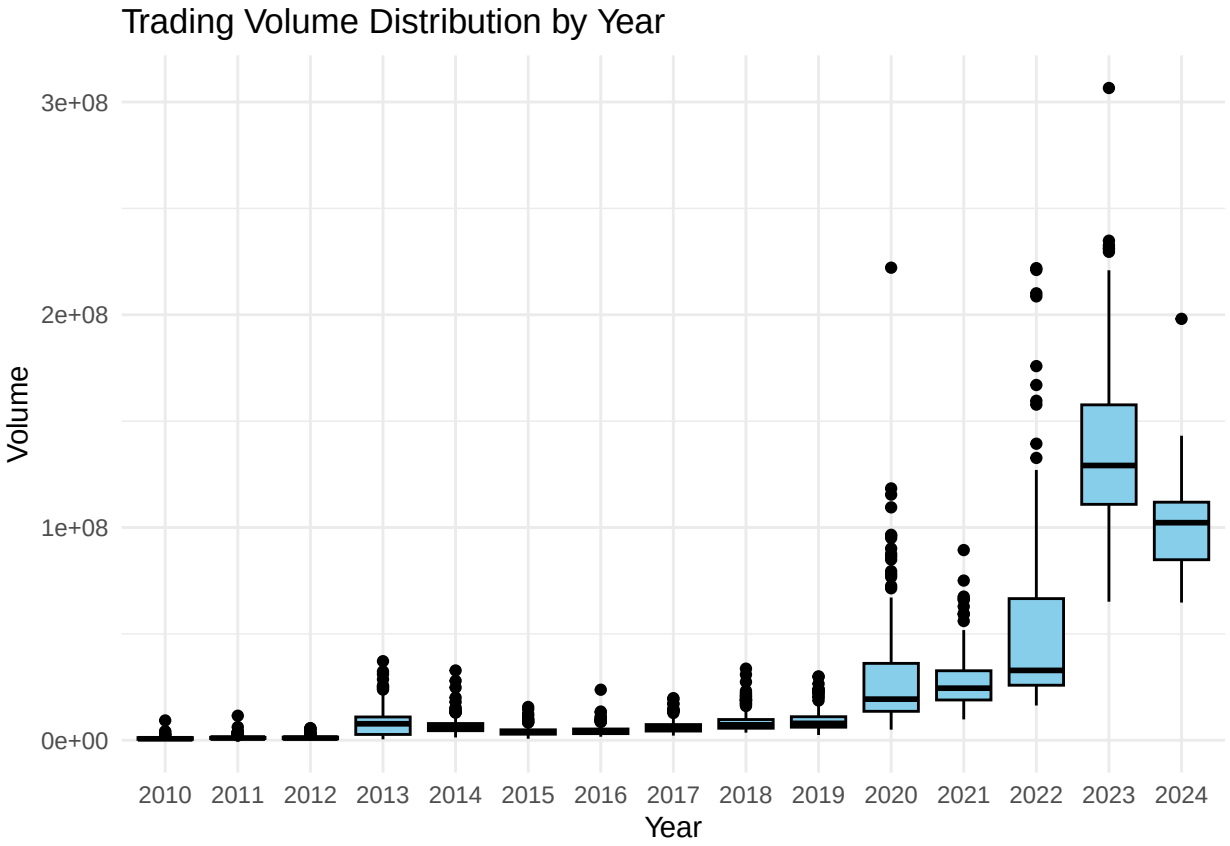
This scatter plot displays the relationship between the high and low stock prices of Tesla, where each point represents a trading day. The points follow the line very closely, which indicates that there is a strong positive correlation between the two metrics. Points are colored according to trading volume, showing variation in market activity. Darker points are low volume, lighter points are high volume. This means that wide ranges of prices, high to low, can happen at any level of volume. The following graph helps to very clearly illustrate the stability that exists between daily price range and trading volume dynamics.

```
# Boxplot for Volume by Month
train_data$month <- month(train_data$date, label = TRUE)
ggplot(train_data, aes(x = month, y = volume)) +
  geom_boxplot(fill = "lightgreen", color = "black") +
  labs(title = "Volume Distribution by Month", x = "Month", y = "Volume") +
  theme_minimal()
```



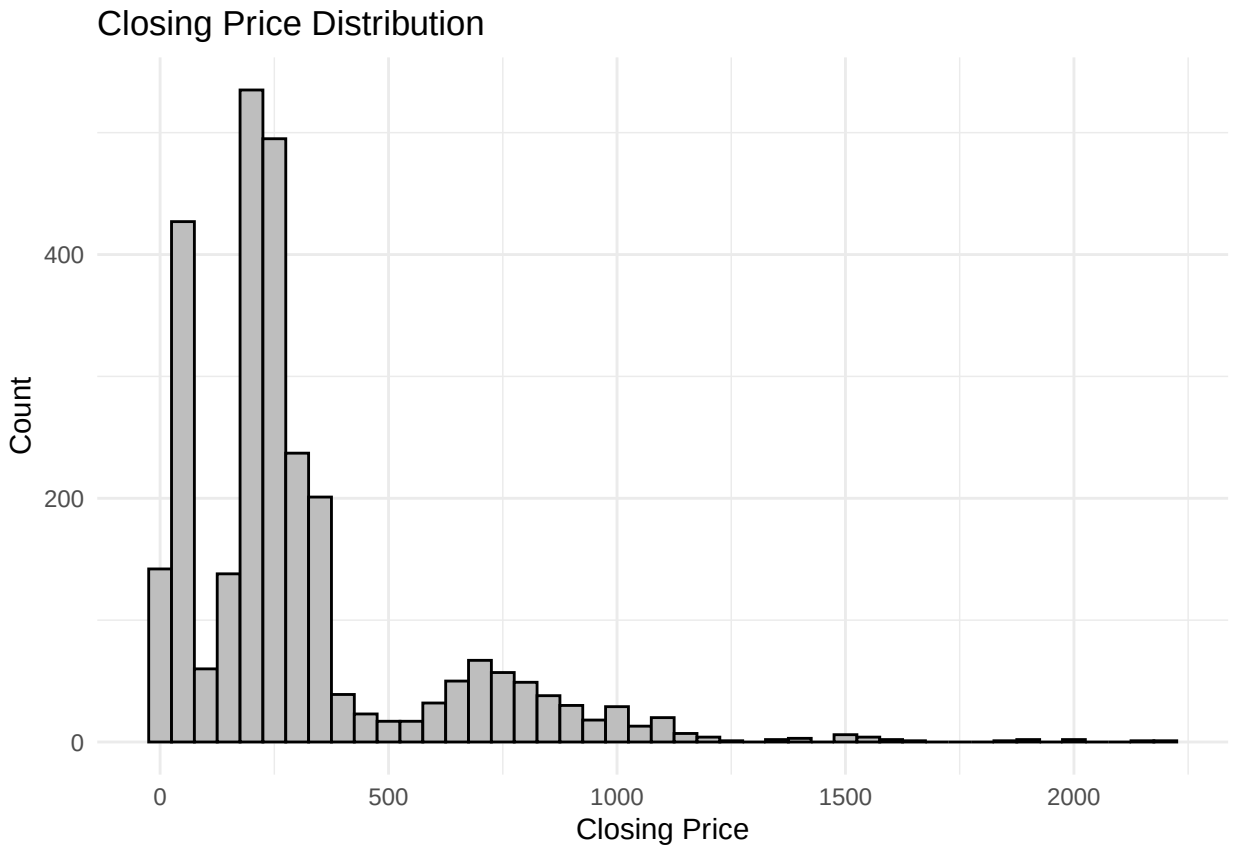

```r
# Boxplot for volume by year
train_data$year <- year(train_data$date)

ggplot(train_data, aes(x = factor(year), y = volume)) +
  geom_boxplot(fill = "skyblue", color = "black") +
  labs(title = "Trading Volume Distribution by Year", x = "Year", y = "Volume") +
  theme_minimal()
```



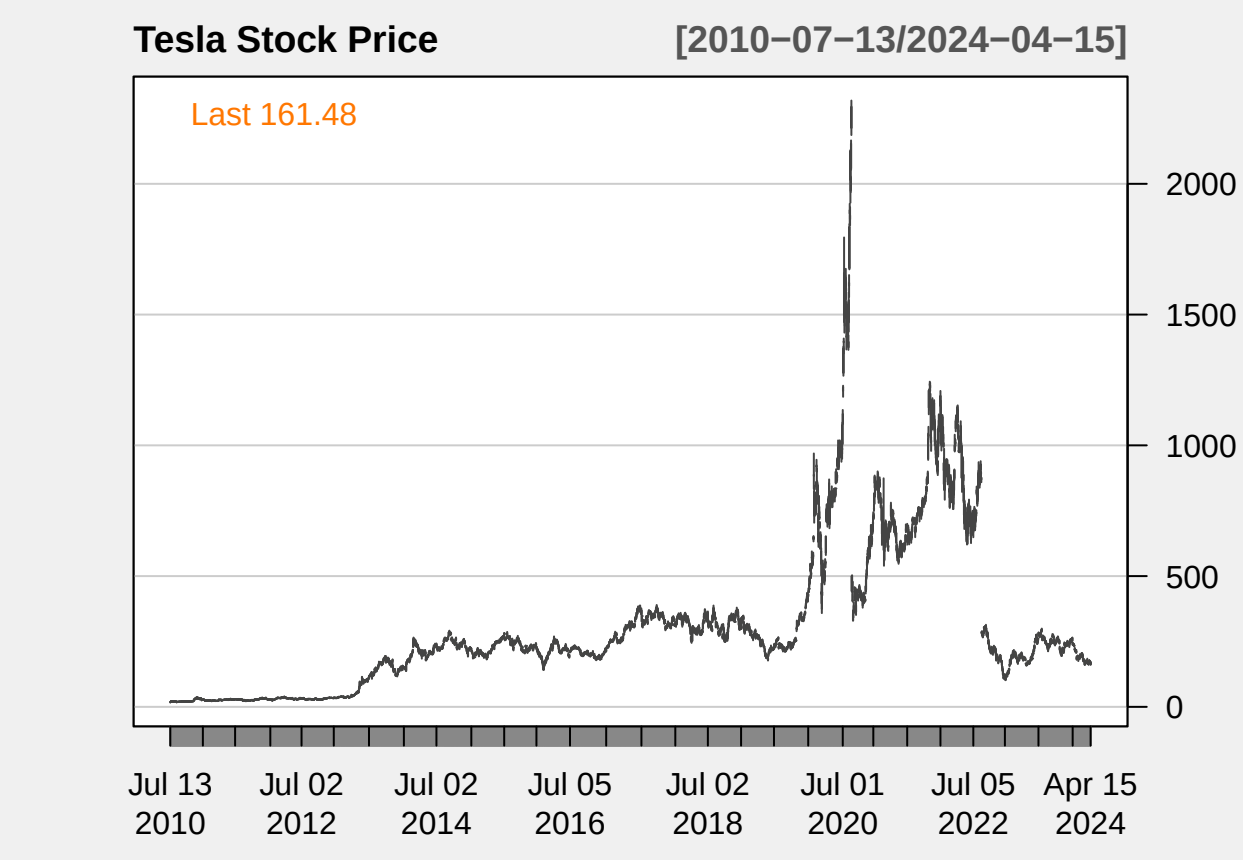
The first box plot visualizes the trading volume distribution by month in Tesla, showing dispersion and the median of trading volumes in each month across the dataset. Though most months show very similar volume distributions, a few outliers indicate days when trading was extremely high. The second boxplot in the series is for the trading volume distribution by year. There is a definite increasing trend of volumes over the years, especially post-2020, coinciding with the major market events of Tesla. An increasing interquartile range and higher median volumes in recent years imply increased market activity and interest in Tesla's stock. Taken together, these plots both show seasonal consistency and long-term growth in trading volumes.

```
# Histogram for Closing Price Distribution
ggplot(train_data, aes(x = close)) +
  geom_histogram(binwidth = 50, fill = "grey", color = "black") +
  labs(title = "Closing Price Distribution", x = "Closing Price", y = "Count") +
  theme_minimal()
```



The above histogram displays the distribution of closing stock prices of Tesla within the period of observation. Most of the closing prices fall within the range of the lower values, around 100 to 500, with the highest frequency being around 600. The distribution has a long right tail, reflecting that the number of occurrences at higher levels of price is much less frequent, with some outliers even going over 1500. This pattern suggests that the price of Tesla stock remained quite stable for most of the time within a certain range but showed occasional jumps during times of rapid growth or high volatility in the stock market. This graph skilfully highlights the skewness and overall spread of the closing prices.

```
# Create a candlestick chart
chartSeries(
  xts(
    train_data[, c("open", "high", "low", "close")],
    order.by = train_data$date
  ),
  type = "candlesticks",
  theme = chartTheme("white"),
  name = "Tesla Stock Price"
)
```

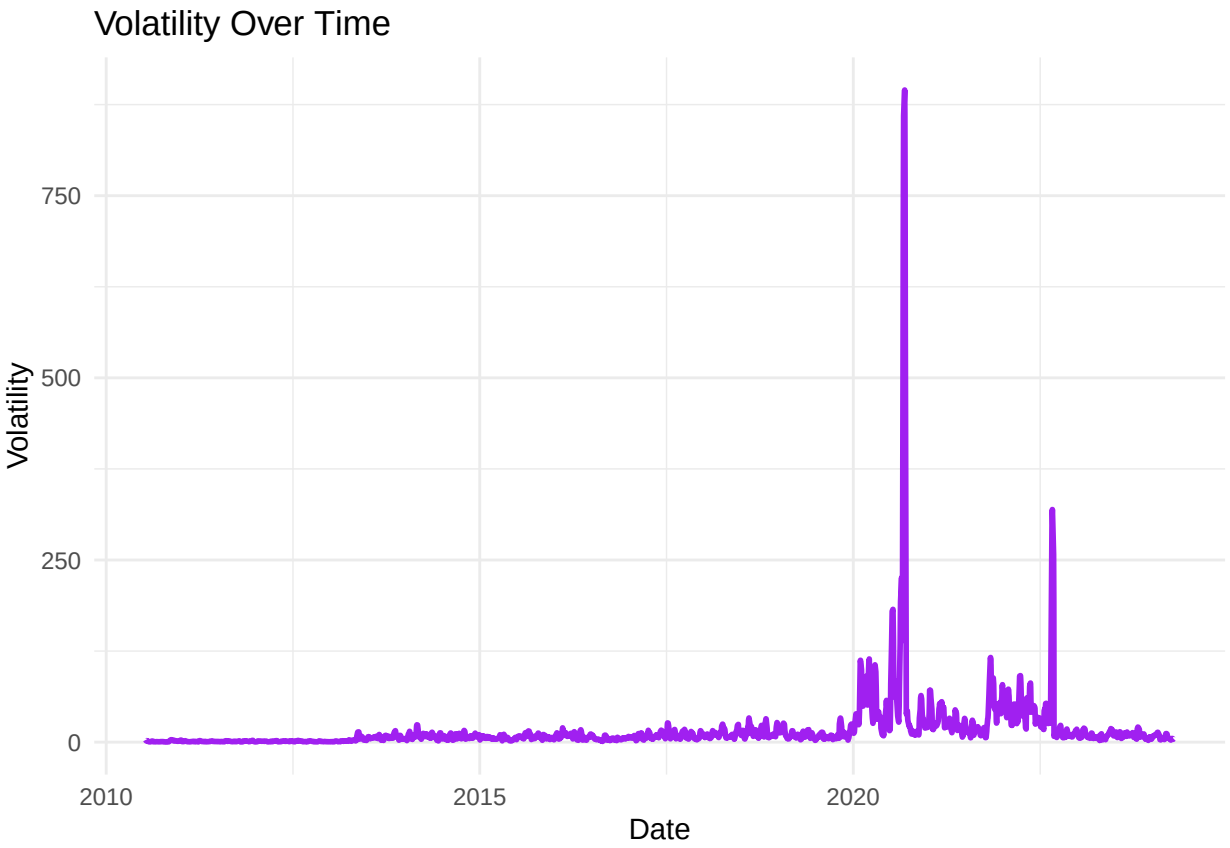


This line graph represents the trends of Tesla's stock prices from July 2010 to April 2024. The stock price showed some ups and downs at relatively low values until 2020. Beyond that, a tremendous upsurge was observed, peaking above 2000. In other words, this graph signals explosive growth in this period due to certain market activities or the publication of milestones by the firm. After the peak, the stock price experiences volatility, with notable fluctuations and a downward adjustment in the following years. The current price, as of the last data point, is 161.48, indicating a substantial correction from its peak. This visualization effectively captures Tesla's dramatic price evolution and the volatility inherent in its trading history.

```
# Trend of High Price over the Years
ggplot(train_data, aes(x = year, y = high)) +
  geom_line(stat = "summary", fun = "mean", color = "red", linewidth = 1) +
  labs(title = "Trend of High Price Over the Years", x = "Year", y = "Average High Price") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

This is a line chart showing the trend in the average high prices over the years of Tesla. From 2010 to 2014, prices remain relatively stable at lower levels, reflecting an early growth stage of the company, while between 2015 and 2019, the increase is linear and steady, driven by growing market confidence. Notice the huge price increase in 2020-2021, even over 800, possibly because of events like Tesla's inclusion into the S&P 500 and an overall increased demand for electric vehicles. After 2021, the prices drop rapidly to show the correction and stabilization of the market. Overall, this chart captures Tesla's fast rise and subsequent adjustment in stock performance.

```
ggplot(train_data, aes(x = date, y = volatility_10d)) +
  geom_line(color = "purple", linewidth = 1) +
  labs(title = "Volatility Over Time", x = "Date", y = "Volatility") +
  theme_minimal()
```



This line graph depicts volatility in the stock price set of Tesla over time. From 2010 until around 2019, this volatility consistently remained low and reflected relatively stable price movements. In the year 2020, there is a huge rise in it, which reflects sharp spikes in both market activity and price fluctuations. Events such as Tesla entering into the S&P 500, among broader market dynamics with respect to the pandemic, align with these observed spikes. It serves well to underline how much more volatile Tesla's stock became in its rapid growth phase and the risks and opportunities of high market interest.

- Based on the conditions assumed by the statistical learning methods, discuss their applicability to the prediction problem.

The applicability of statistical learning methods, including logistic regression and stepwise regression, relies on the solvability of the binary classification problem of predicting Tesla stock performance. Logistic regression therefore fits well, since logistic regression models the probability given independent variables of a binary outcome. Because it considers the engineered feature using moving averages and standardized values, the assumptions of linearity in the relationship of the log-odds of the target with respect to the predictors are relatively satisfied. Similarly, given the independence of the observations concerning daily trading data, this technique assumes that the observations are non-autocorrelated.

Complementary to logistic regression, stepwise regression works on the optimization of feature selection so that only the most predictive variables are included. It iteratively compares different predictors with Akaike Information Criterion, which balances model complexity and performance. This approach proves to be very useful when handling multiple engineered features like changes

in volatility and volume since it reduces redundancy and focuses only on the significant predictors.

These methods would be more appropriate to the problem due to their simplicity and interpretability. Though logistic regression works fine in instances of linear relationships, stepwise regression enhances the model's efficiency and cuts down the possibility of overfitting. The possible shortcoming with them is that they cannot depict complex nonlinear relationships that may further involve advanced models.

Predictive analysis and results [35 points]

- Apply and document the statistical learning procedure for the predictive analysis.

```
# Scale numerical variables for training and test data
numeric_vars <- c("volume", "avg_vol_20d", "change_percent", "open", "high", "low", "close", "mo
train_data_scaled <- train_data
train_data_scaled[, numeric_vars] <- scale(train_data[, numeric_vars])
test_data_scaled <- test_data
test_data_scaled[, numeric_vars] <- scale(test_data[, numeric_vars])


# Ensure the target variable is a factor
train_data_scaled$binary_target <- factor(train_data_scaled$binary_target, levels = c(0, 1), lab
test_data_scaled$binary_target <- factor(test_data_scaled$binary_target, levels = c(0, 1), label

# Set up cross-validation parameters
cv_control <- trainControl(method = "cv",    # Cross-validation
                           number = 5,      # Number of folds
                           savePredictions = "final",  # Save predictions
                           classProbs = TRUE)          # Enable class probabilities

# Train logistic regression model with cross-validation
log_model_cv <- train(binary_target ~ open + high + low + close + volume + adjusted_close + avg_
                        moving_avg_5d + moving_avg_10d + volatility_10d + volume_avg_5d,
                      data = train_data_scaled,
                      method = "glm",
                      family = "binomial",
                      trControl = cv_control)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# View cross-validation results
print(log_model_cv)
```

```
## Generalized Linear Model
##
## 2771 samples
##   11 predictor
##    2 classes: 'Class0', 'Class1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2216, 2217, 2217, 2217, 2217
## Resampling results:
##
##   Accuracy   Kappa
##   0.8505975  0.7004503
```

The output provides the results of a cross-validated logistic regression model using 5-fold cross-validation on a dataset of 2,771 samples with 11 predictors and two target classes (Class0 and Class1). The model achieved an average accuracy of 84.52%, indicating that it correctly classifies approximately 85% of the samples in the validation folds. The Kappa statistic is 0.6960, suggesting a substantial level of agreement between the predicted and actual classifications, accounting for chance agreement. This indicates that the model performs well in distinguishing between the two classes.

```
# Load necessary library
library(MASS)

# Define the null model (intercept only) and the full model (all predictors)
null_model <- glm(binary_target ~ 1, data = train_data_scaled, family = binomial)
full_model <- glm(binary_target ~ open + high + low + close + volume + adjusted_close +
                    avg_vol_20d + moving_avg_5d + moving_avg_10d + volatility_10d + volume_avg_5
                  data = train_data_scaled, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# Perform stepwise regression
stepwise_model <- stepAIC(null_model,
                          scope = list(lower = null_model, upper = full_model),
                          direction = "both",
                          trace = TRUE)
```

```
## Start:  AIC=3837.23
## binary_target ~ 1
##
##                  Df Deviance    AIC
## + close           1   3831.5 3835.5
## + low             1   3833.1 3837.1
## <none>                3835.2 3837.2
## + high            1   3833.5 3837.5
## + avg_vol_20d     1   3833.6 3837.6
```

```
## + volume_avg_5d   1   3833.9 3837.9
## + adjusted_close  1   3834.0 3838.0
## + volume          1   3834.3 3838.3
## + open            1   3834.4 3838.4
## + moving_avg_5d   1   3834.8 3838.8
## + moving_avg_10d  1   3835.1 3839.1
## + volatility_10d  1   3835.2 3839.2
##
## Step:  AIC=3835.49
## binary_target ~ close

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##                   Df Deviance    AIC
## + open             1   2887.9 2893.9
## + high             1   3290.4 3296.4
## + moving_avg_5d    1   3396.4 3402.4
## + low              1   3484.8 3490.8
## + moving_avg_10d   1   3723.7 3729.7
## <none>                 3831.5 3835.5
## + volume_avg_5d    1   3830.4 3836.4
## + volatility_10d   1   3830.8 3836.8
## + volume           1   3830.8 3836.8
## + avg_vol_20d      1   3831.2 3837.2
## - close            1   3835.2 3837.2
## + adjusted_close   1   3831.5 3837.5

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##
## Step:  AIC=2893.89
## binary_target ~ close + open

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##                   Df Deviance    AIC
## + moving_avg_5d    1   2732.7 2740.7
```

```
## + moving_avg_10d  1   2844.8 2852.8
## + low             1   2873.0 2881.0
## + high            1   2878.5 2886.5
## <none>                2887.9 2893.9
## + volatility_10d  1   2886.5 2894.5
## + avg_vol_20d     1   2887.5 2895.5
## + volume_avg_5d   1   2887.5 2895.5
## + volume          1   2887.6 2895.6
## + adjusted_close  1   2887.7 2895.7
## - open            1   3831.5 3835.5
## - close           1   3834.4 3838.4

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##
## Step:  AIC=2740.67
## binary_target ~ close + open + moving_avg_5d

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##                  Df Deviance    AIC
## + moving_avg_10d  1   2706.0 2716.0
## + low             1   2717.8 2727.8
## + volatility_10d  1   2726.5 2736.5
## <none>                2732.7 2740.7
## + volume          1   2731.3 2741.3
## + volume_avg_5d   1   2731.3 2741.3
## + high            1   2731.5 2741.5
## + adjusted_close  1   2732.1 2742.1
## + avg_vol_20d     1   2732.3 2742.3
## - moving_avg_5d   1   2887.9 2893.9
## - open            1   3396.4 3402.4
## - close           1   3832.1 3838.1

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##
## Step:  AIC=2715.98
```

```
## binary_target ~ close + open + moving_avg_5d + moving_avg_10d

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##                  Df Deviance    AIC
## + low             1   2696.4 2708.4
## <none>                2706.0 2716.0
## + high            1   2704.1 2716.1
## + volume          1   2705.2 2717.2
## + volume_avg_5d   1   2705.3 2717.3
## + adjusted_close  1   2705.8 2717.8
## + avg_vol_20d     1   2706.1 2718.1
## + volatility_10d  1   2714.9 2726.9
## - moving_avg_10d  1   2732.7 2740.7
## - moving_avg_5d   1   2844.8 2852.8
## - open            1   3353.7 3361.7
## - close           1   3828.3 3836.3

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##
## Step:  AIC=2708.42
## binary_target ~ close + open + moving_avg_5d + moving_avg_10d +
##     low

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##                  Df Deviance    AIC
```

```
## + high             1   2693.7 2707.7
## <none>                 2696.4 2708.4
## + volume_avg_5d    1   2696.3 2710.3
## + volume           1   2696.3 2710.3
## + avg_vol_20d      1   2696.4 2710.4
## + adjusted_close   1   2696.6 2710.6
## - low              1   2706.0 2716.0
## + volatility_10d   1   2702.0 2716.0
## - moving_avg_10d   1   2717.8 2727.8
## - moving_avg_5d    1   2824.5 2834.5
## - open             1   3062.5 3072.5
## - close            1   3448.0 3458.0

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##
## Step:  AIC=2707.73
## binary_target ~ close + open + moving_avg_5d + moving_avg_10d +
##     low + high

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##                  Df Deviance    AIC
## <none>                 2693.7 2707.7
## - high             1   2696.4 2708.4
## + volume           1   2693.4 2709.4
## + volume_avg_5d    1   2693.5 2709.5
## + adjusted_close   1   2693.8 2709.8
## + avg_vol_20d      1   2693.8 2709.8
## - low              1   2704.1 2716.1
## + volatility_10d   1   2701.2 2717.2
## - moving_avg_10d   1   2715.1 2727.1
## - moving_avg_5d    1   2817.6 2829.6
## - open             1   2837.2 2849.2
## - close            1   3277.8 3289.8
```

```r
# Display the summary of the selected model
summary(stepwise_model)
```

```
##
## Call:
## glm(formula = binary_target ~ close + open + moving_avg_5d +
##     moving_avg_10d + low + high, family = binomial, data = train_data_scaled)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      0.17090    0.06677   2.560 0.010477 *
## close           97.85266    5.74598  17.030  < 2e-16 ***
## open           -55.84564    4.49450 -12.425  < 2e-16 ***
## moving_avg_5d  -18.26748    2.21691  -8.240  < 2e-16 ***
## moving_avg_10d   2.76136    0.87218   3.166 0.001545 **
## low            -17.06562    4.40741  -3.872 0.000108 ***
## high            -9.21045    5.00161  -1.841 0.065549 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3835.2  on 2770  degrees of freedom
## Residual deviance: 2693.7  on 2764  degrees of freedom
## AIC: 2707.7
##
## Number of Fisher Scoring iterations: 8
```

- Estimate the performance of the statistical learning approaches on test data, using resampling methods or other measures.

```
# Use the trained cross-validated model to predict test data
predicted_classes_cv <- predict(log_model_cv, newdata = test_data_scaled)
conf_matrix_cv <- confusionMatrix(predicted_classes_cv, test_data_scaled$binary_target)
print(conf_matrix_cv)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Class0 Class1
##     Class0    234     46
##     Class1    116    296
##
##                Accuracy : 0.7659
##                  95% CI : (0.7325, 0.797)
##     No Information Rate : 0.5058
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5328
##
```

```
##  Mcnemar's Test P-Value : 5.922e-08
##
##             Sensitivity : 0.6686
##             Specificity : 0.8655
##          Pos Pred Value : 0.8357
##          Neg Pred Value : 0.7184
##              Prevalence : 0.5058
##          Detection Rate : 0.3382
##    Detection Prevalence : 0.4046
##       Balanced Accuracy : 0.7670
##
##        'Positive' Class : Class0
##
```

```
# Extract metrics
precision <- conf_matrix_cv$byClass["Pos Pred Value"]  # Precision
recall <- conf_matrix_cv$byClass["Sensitivity"]        # Recall

# Calculate F1 Score
f1_score <- 2 * (precision * recall) / (precision + recall)
# Print the F1 Score
cat("F1 Score for Logistic Regression Model:", round(f1_score, 4), "\n")
```

```
## F1 Score for Logistic Regression Model: 0.7429
```

- Evaluate the performance on the test data.

```
# Plot ROC Curve for the test set
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

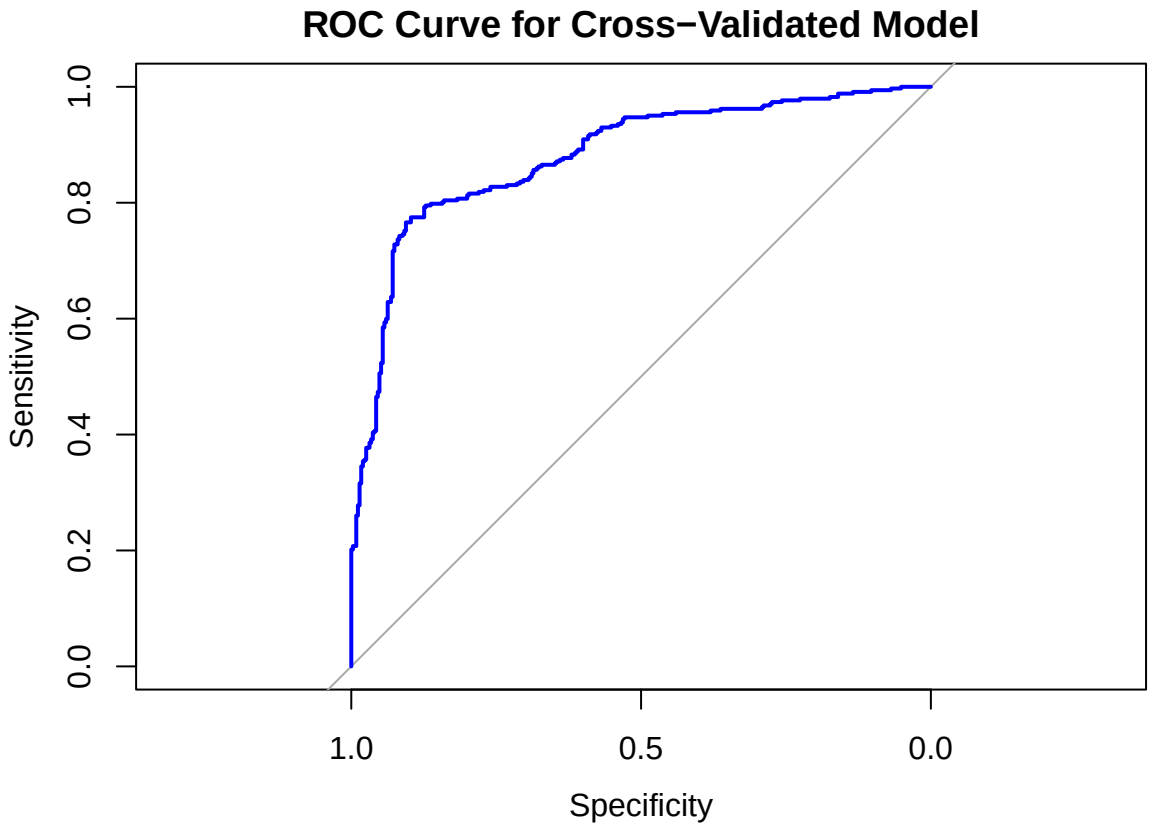
```
predicted_probs_cv <- predict(log_model_cv, newdata = test_data_scaled, type = "prob")

roc_curve_cv <- roc(test_data_scaled$binary_target, predicted_probs_cv[, 2])
```

```
## Setting levels: control = Class0, case = Class1
```

```
## Setting direction: controls < cases
```

```
plot(roc_curve_cv, col = "blue", lwd = 2, main = "ROC Curve for Cross-Validated Model")
```




```
# Calculate AUC
auc_value_cv <- auc(roc_curve_cv)
cat("AUC for Cross-Validated Model:", auc_value_cv, "\n")
```

```
## AUC for Cross-Validated Model: 0.8804177
```

- Discuss the results.

The logistic regression model works in producing the results of Tesla's performance of stocks and hence providing a more interpretable and statistically valid classification framework. For the proposed model, its cross-validation accuracy was 76.59%, sensitivity was 66.86%, and specificity was 86.55%, thus showing the overall power of this model in the balance between prediction classes, as can also be obtained from the confusion matrix. An AUC score from the ROC curve of 0.88 underpins the fact that this indeed has high discriminatory power and hence, will be reliable in making out the trends of the stocks while minimizing the trade-offs between false positives and negatives.

However, logistic regression has the limitation of capturing the complex, nonlinear relationship commonly seen in financial data due to its linear nature. Supplementing with more advanced techniques such as decision trees, random forests, or other ensemble models may be able to improve predictive accuracy by capturing complex patterns and interactions among features. Nevertheless, this model of logistic regression gives robust and interpretable baselines and is really valuable for understanding the stock dynamics of Tesla and forming a basis for exploring more advanced approaches. Using cross-validation also enhances its reliability due to the insurance of good performance on a variety of data splits.

Conclusion [15 points]

- Discuss the scope and generalizability of the predictive analysis.

Tesla stock prediction with data illustrates poignantly how, generally speaking, the statistical learning approach is effective to resolve the binary classification problem in the estimation of stocks with the use of logistic regression. The model turned out to be generalizable with the given dataset- as was deduced from different metrics like accuracy and AUC. Coupled with feature engineering, such as moving averages, volatility, and changes in volume, the addition of depth in understanding stock behavior adds to this model's potential for generalizability of its findings to other datasets for financial markets. However, the model has limitations. Logistic regression assumes linearity between predictors and the target variable, which may not fully capture complex, non-linear relationships inherent in financial data. Additionally, the scope of the analysis is restricted to the features and time period available in the dataset, limiting its generalizability to other stocks or markets without further validation. The model's reliance on historical data also means that it may not adapt well to abrupt changes in market conditions, such as those caused by unforeseen events.

- Discuss potential limitations and possibilities for improvement.

This can be further improved by incorporating state-of-the-art models for feature engineering, such as decision trees, random forests, or neural networks, which will leverage nonlinear relationships and interactions between features better. Further supplementation with exogenous factors economic indicators or even characteristics of sentiment can further increase higher predictive power and a wider scope of generalization out of sample. It would then test better in the out-of-sample process or cross-validated model for different periods with a higher degree of reliability as to external validity. These enhancements may make a huge difference in the performance of the model and, more importantly, its utility in real-world financial decision-making.