Time series analysis on the stock price of Tesla Inc.

Wenhao Pan (3034946058) Ruojia Zhang (3034069393)
Mengzhu Sun (3034163682) Xiangxi Wang (3036138312)
Mingmao Sun (3033987480)

November 23, 2021

Contents

1 Abstract

In this report, we explore different univariate time series models and select one to fit and predict the daily close price of Tesla Inc. in the stock market. We collect the stock market data from Yahoo Finance. After careful model consideration and selection, we select $ARIMA(1,2,1)\times(0,0,0)[0]$ as our best mode. However, our best model still gives relatively inaccurate predictions. Our analysis demonstrates that using a purely statistical and mathematical tool to model the stock price data might be unrealistic. We provide suggestions for future research on modeling stock price data in the conclusion section.

2 Introduction

Due to the increasing focus on carbon neutrality, the trend of replacing non-sustainable energy with sustainable energy has boomed in the past few years. As relatively environment-friendly energy, electricity has been considered a replacement for traditional energy, such as gasoline and diesel. Among all those enterprises pursuing commercialized carbon neutrality, Tesla, as the largest electric car company, has been pioneering the fashion and aiming to transition the world to electric mobility. As a reflection of the public's belief, the stock price of Tesla has been sedentary for some time and has no evident increase until recent years. It is known that the stock price of a company is highly associated with its real-time status. The status of Tesla might be an adequate representation of the entire electric car industry. Thus, We are interested in building a univariate time series model to fit and predict the stock price of Tesla so that we can predict its future status. However, as we all know, predicting stock prices is an extremely challenging task, so our predictions might be inaccurate.

3 Data Description

The Tesla stock price comes from Yahoo Finance. The stock price dataset consists of open price, close price, high price, and low price of a trading day. We choose the close price to work on because it represents the will of the main forces more than other attributes. The whole volume of data, which contains 2791 data points, has a variance of 39768.49, max price of 1208.59, min price of 4.01, mean price of 112.4271.

4 Exploratory Data Analysis

To obtain a comprehensive understanding of the data, we first conduct explanatory data analysis (EDA). Figure 1(a) is the time series plot of all the given data points. We observe that the stock prices of Tesla before 2020 are averagely and considerably lower than those after 2020. The significantly different scales of different parts of the time series make it hard to visually examine the trend and seasonality pattern of the time series. Moreover, since the popularity of Tesla has only been significantly increasing recently, we do not have to analyze all the available data. Therefore, for the sake of interest and convenience, we decide only to analyze the last 300 time points, or trading days, which cover the period from `2020-08-26` to `2021-11-02` excluding weekends. Thus, whenever we use the word "data" in the following analysis, we implicitly mean the time series of the last three hundred time points.

Figure 1(b) is the time series plot of the close prices of Tesla in the last three hundred trading days before and including `2021-11-02`. We first observe that our data is roughly homoscedastic. To verify our observation, we try the square root and natural log transformations and judge whether they effectively stabilize the variance of the time series. Their plots are below in Figure 2.

We can see that both transformations unnecessarily increase the variance of the time series before mid-November in 2020 and do not change the variance of other time series data. Although both transformations shorten the vertical distance between the maximum and minimum of the time series after Oct. 2021, the spike

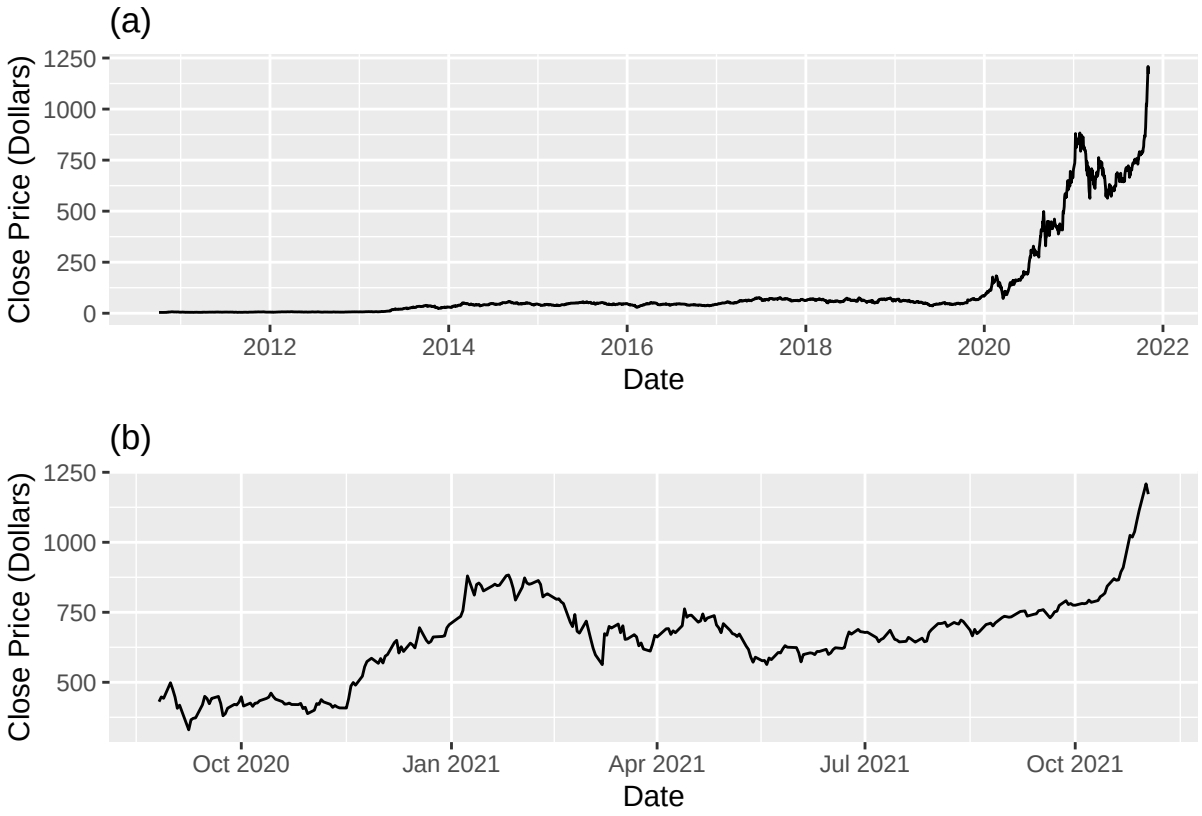

Figure 1: (a) Time series plot of all available trading days. (b) Time series plot of last 300 trading days

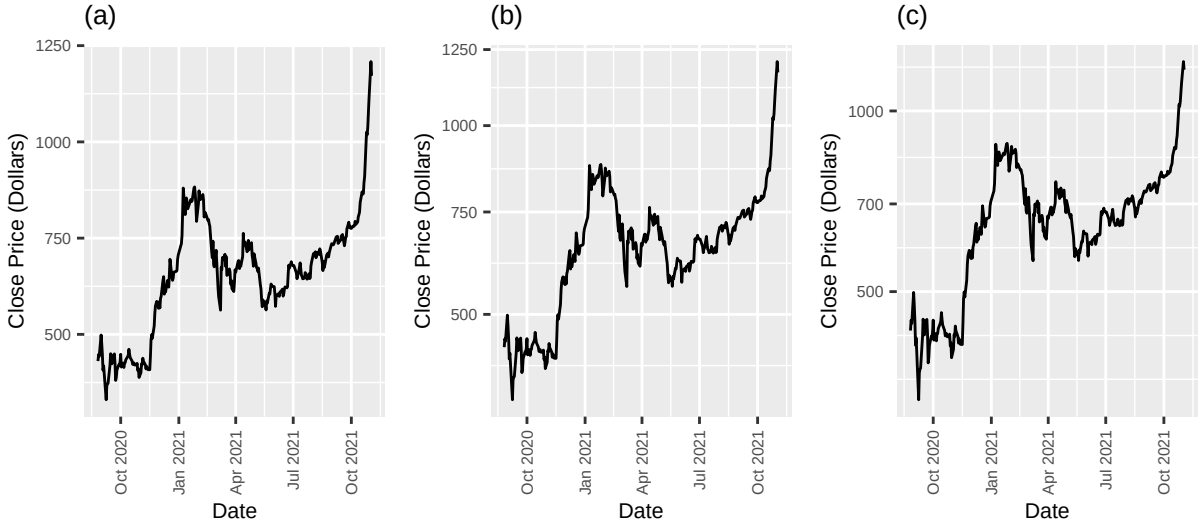

Figure 2: (a): Original time series. (b): Square root transfromed time seires. (c): Natural log transformed time series.

after Oct. 2021 is more like an increasing trend than a considerable fluctuation. In short, both transformations are redundant, and we do not need to use any variance stabilizing transformation.

Back to Figure 1(b), intuitively, the data is not stationary because of a nonlinear and generally increasing trend. The trend first increases until around Feb. 2021 and then decreases until around mid-May. 2021. Finally, the trend increases again until the end of the time series. Nonetheless, we do not observe an obvious or significant seasonality pattern. It matches the intuition since the granularity of our data is day, and the structure of daily stock price data is too complicated to have a seasonality pattern.

In conclusion, based on all the previous discussions in EDA, we decide to construct possible models on the original time series data, including only the last three hundred time points.

5 Model Construction

With a comprehensive understanding of our data, we start to experiment and construct different time series models. We choose and build two non-parametric signal models of the trend and seasonality in our data. We aim to make the residuals obtained from removing the signalsapproximately weekly stationary. We do not consider any parametric trend model because we think the trend of the stock price data is too complicated to be modeled by a parametric model, such as a high-order polynomial. Indeed, we could use a 15 or 20 order polynomial, but it may overfit the training data and produce imprecise predictions. We do not consider a parametric seasonality model either because we do not find a clear seasonality pattern in our data by the EDA. Finally, based on each signal model, we provide two ARMA models or its extensions, such as SARMA or ARIMA, to whiten the residuals of the signal model. Thus, we have four candidate models, and we will explain how we select a final model among them in the next section.

5.1 Non-parametric Signal Model: exponential smoothing

In this signal model, we choose exponential smoothing with weight $\alpha = 0.9$ and lag $k = 10$ and a seasonal differencing with period $d = 5$.

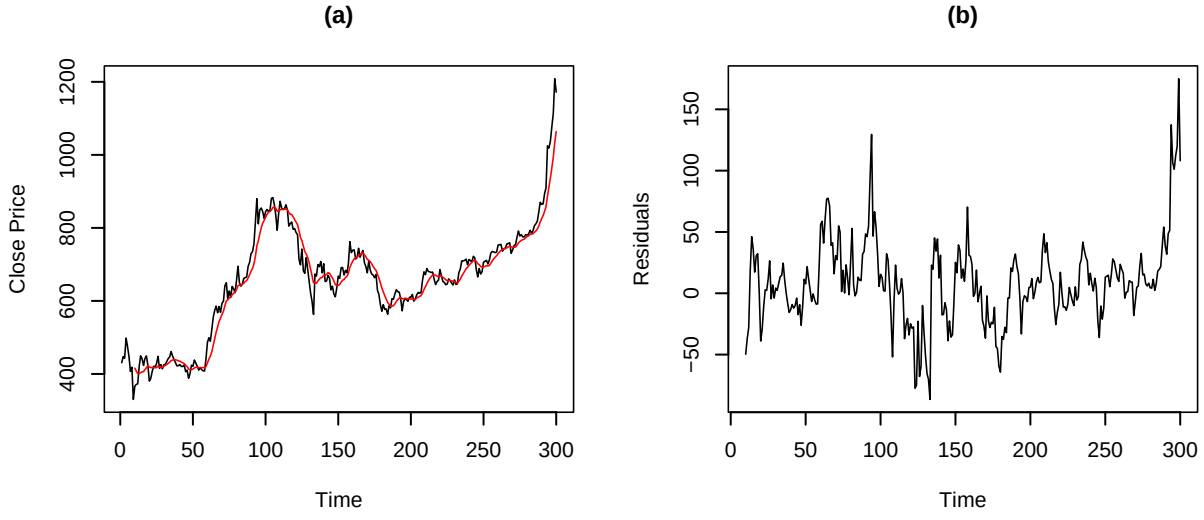

Figure 3: (a): Time series plot of the original data and fitted values. (b): The residual plot of exponential smoothing.

We experiment with different combinations of α and k with a careful consideration of overfitting issue. We choose $k = 10$ as the final value because we want only to use the past two weeks, which are ten days in our data, to forecast. We choose $\alpha = 0.9$ as the final value because we think it best balances the smoothing effect and the capture of trend pattern among $(0, 1)$. Indeed, the smoothing line in Figure 3(a) fits the data in the way we want. Note that we lose the first nine time points due to the computation process of the exponential smoothing.

However, the residual plot Figure 3(b) is fairly non-stationary, as it has a cycling fluctuation pattern and still a slightly nonlinear trend. It might be due to that we intentionally let exponential smoothing not fit the data perfectly. Next, we use the seasonal differencing with period $d = 5$, which is one week in our data, to make further the residuals more stationary.

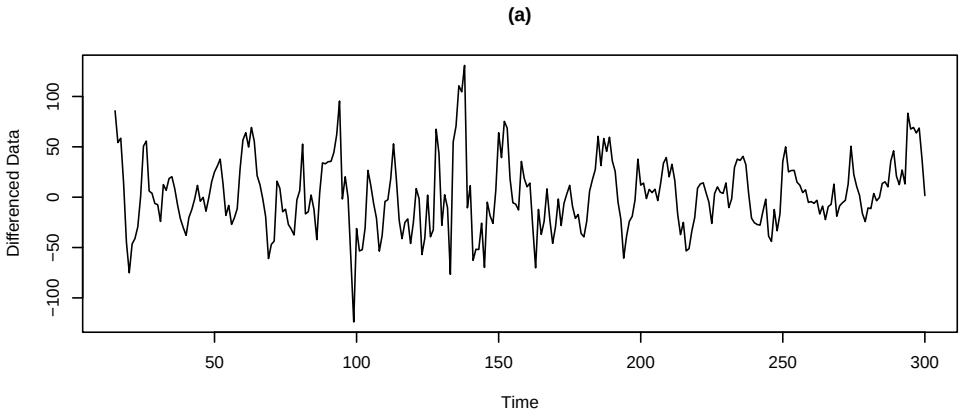

Figure 4: (a): Time series plot of the seasonal differenced (d = 5) residuals from the previous smoothing.

Indeed, now the differenced residuals have become more stationary. There seems to be a contradiction that recalling in EDA, we claim that there is not a clear seasonality in our data. However, the effect of the seasonal differencing here implies a possible seasonality with period $d = 5$. We think it might be that the seasonal differencing is removing the remaining trend left by the exponential smoothing instead of the seasonality. Nevertheless, We believe that the time series of the differenced residuals shown in Figure 4(a) is stationary enough for us to build ARMA models on it. For convenience, we use "residuals" to represent "the seasonal differenced residuals" in the following two subsections.

5.1.1 ARMA 1A: $ARIMA(1, 0, 3) \times (0, 0, 1)[5]$

After acquiring approximately stationary residuals from the signal model, we use the ARMA model to model the residuals further to predict future residuals reasonably. Then, summing the predicted residuals and signals gives us the prediction of the original time series, stock price. To obtain the intuition for choosing the ARMA model, we use the sample ACF and PACF plots of the residuals shown in Figure 5.

If we ignore lag 5 temporarily in the ACF plot, we observe an approximate cutoff after lag 3, so $MA(3)$ seems to be a reasonable choice. However, if we temporarily ignore lag multiples of 5, the PACF plot also seems to have a cutoff after lag 1. Then, a simple $MA(3)$ model becomes invalid. $AR(1)$ is also invalid for the same reason. To handle the spikes at lag 5 in ACF and lag multiples of 5 in PACF, we may consider seasonal MA or AR with period $d = 5$. To select one model, we will compute and compare AIC, AICc, BIC, and cross-validation errors of models that are ARMA(1,3) plus different orders of seasonal MA or AR. Then, the one that has best performance in terms of these four criterions will be chosen. After careful selection, we choose $ARIMA(1, 0, 3) \times (0, 0, 1)[5]$. See the code appendix for more details about the selection process.

To assess our model, we visualize different model diagnostics methods in Figure 6. The standardized residuals seem to have an i.i.d. standard normal distribution with a slightly larger variance. In the ACF plot of

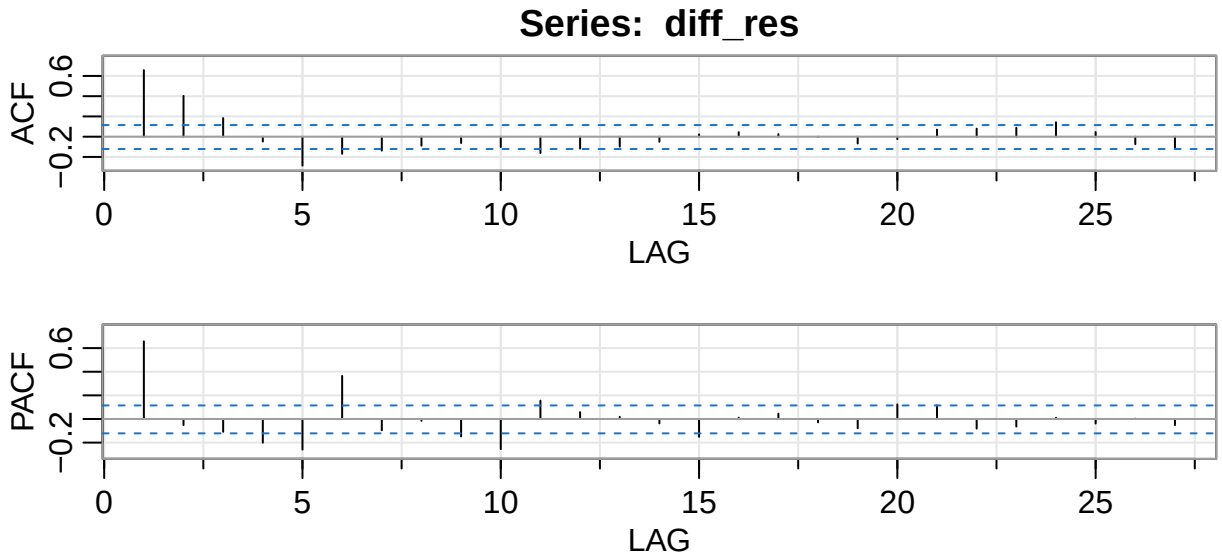

Figure 5: The sample ACF and PACF plots of the residuals from expo. smoothing

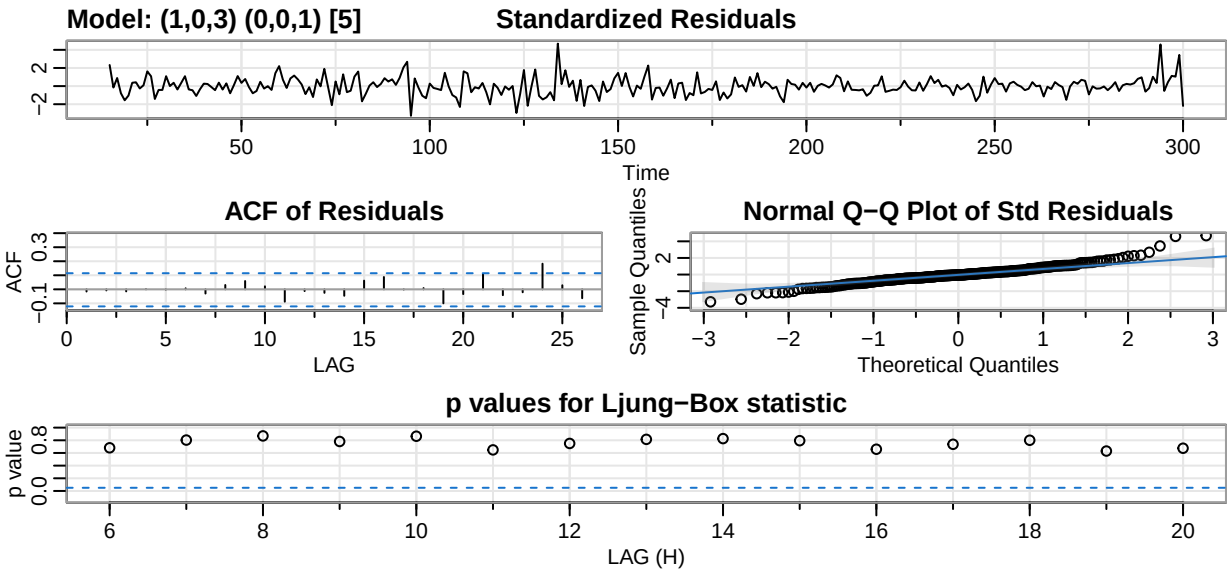

Figure 6: Model diagnostics plots of ARIMA(1,0,3)x(0,0,1)[5]

residuals, almost all the spikes lie within a 95% confidence interval. The normal q-q plot implies a slightly fatter tail distribution than the normal distribution, but we can still claim the validity of the normality assumption. The p-values of all the Ljung-Box statistics are above 0.05, so we fail to reject the null hypothesis that the data was generated from our candidate ARMA model. In summary, all the model diagnostics methods imply that our candidate ARMA model is considered reasonable. We call this model "ARMA 1A".

5.1.2 ARMA 1B: $ARIMA(1,0,1) \times (0,0,1)[5]$

Here we provide a different way to model the seasonal differenced residuals.

From the ACF and PACF plots of the residuals in Figure 5, we can observe a negative spike at lag = 5 for both plots. If we do not ignore the spikes at some lag = 5 in ACF plot, as we did in ARMA 1A, neither the ACF or the PACF plot has a reasonable cutoff. Thus, we need to choose both $p > 0$ and $q > 0$.

`auto.arima()` returns a $ARIMA(2,0,3) \times (0,0,0)[5]$ model but with bad diagnostics performance. Nonetheless, we can still treat it as a reference model and develop a better model based on it. We will use the same selection process for selecting ARMA 1A. See the code appendix for more details. We end up with $ARIMA(1,0,1) \times (0,0,1)[5]$ model.

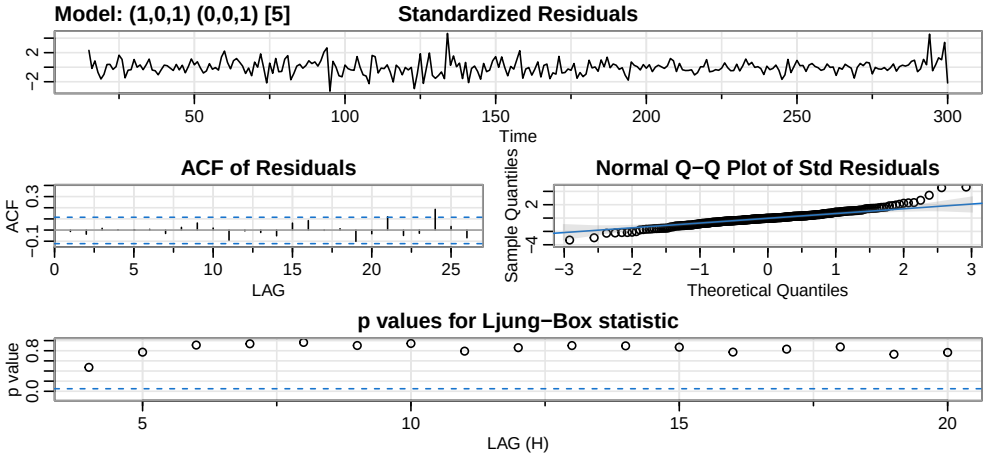

Figure 7: Model diagnostics plots of ARIMA(1,0,1)x(0,0,1)[5]

The diagnostics plots in Figure 7 provide highly similar information to those of ARMA 1A. For example, all the p-values of Ljung-Box statistics are large. Thus, we consider this model as a reasonable one and call it "ARMA 1B".

5.2 Non-parametric Signal Model: second-order differencing

In this model, we choose the second-order differencing to remove the trend. After the first-order differencing, we observe that there is still some trend pattern, such as the increasing one between 270 and 300, as shown by Figure 8(a). This matches our previous analysis that the trend of our data is nonlinear in EDA. Thus, we take another differencing and acquire the second-order differencing data shown in Figure 8(b).

The second-order differenced time series, denoted by Z_t, is more stationary than the first-order differenced time series. We can keep trying more higher-order differencings, but they may overfit our data. Therefore, we stop at the second-order differenced time series Z_t, which is already stationary enough for us to build an ARMA model on it.

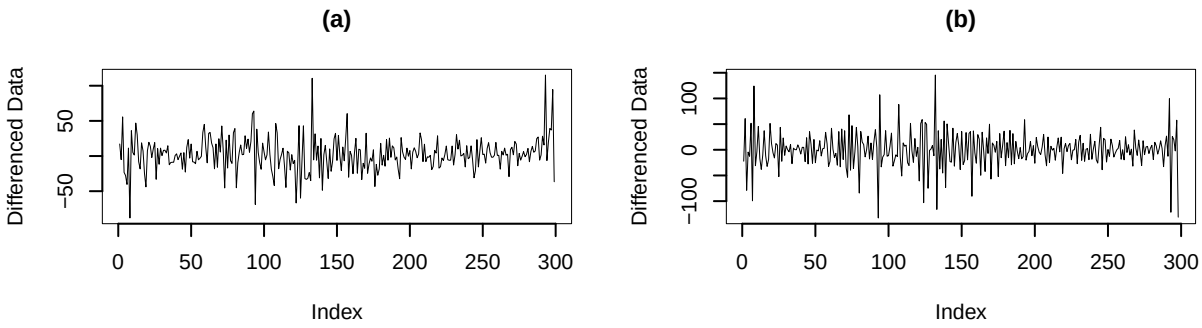

Figure 8: (a): The first-order differenced data. (b): The second-order differenced data.

5.2.1 ARMA 2A: $ARIMA(1,0,1) \times (0,0,0)[0]$

Based on Z_t, we continue our modeling by finding and fitting a suitable ARMA model on it. To obtain some intuition, we first plot the sample ACF and PACF of Z_t in Figure 9 as we did for the seasonal differenced residuals.

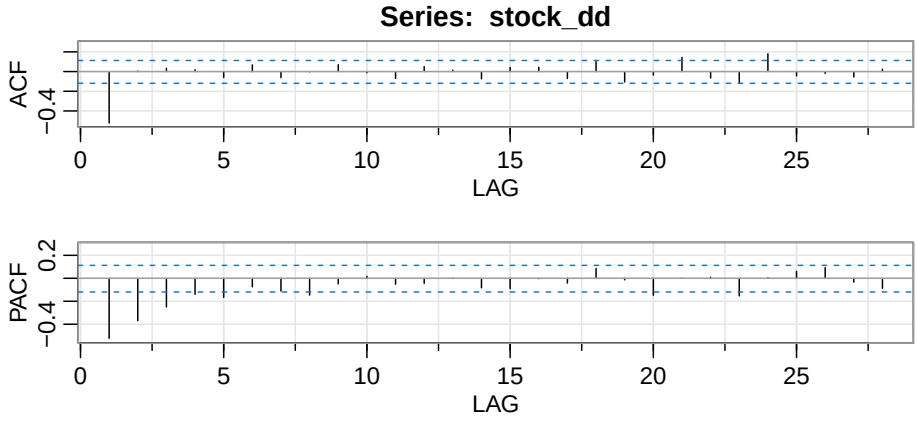

Figure 9: The sample ACF and PACF plots of Z_t

From the ACF plot of Z_t, we observe a rough cutoff after lag 1, so MA(1) seems to be a reasonable choice, but we are not completely sure. Moreover, we are not sure about the order of p. We use the `auto.arima()` to help us find the suitable p, and the function returns a $ARIMA(1,0,1) \times (0,0,0)[0]$ model on Z_t. We will accept this model.

Because the performance of the diagnostics plots in Figure 10 is similar to those of ARMA 1A and 1B models, our model choice here is reasonable.

5.2.2 ARMA 2B: $ARIMA(4,0,1) \times (0,0,0)[0]$

We provide a second way to model the second-ordered differenced time series Z_t.

In the PACF plot of Z_t in Figure 9, the magnitude of PACF is decreasing significantly from lag 1 to lag 5, so there might be a cutoff around lag 5. From the ACF plot of Z_t, we observe an evident cutoff of the magnitude of ACF occurs at lag 1. Therefore, we cannot use a simple MA or AR model. We use the same

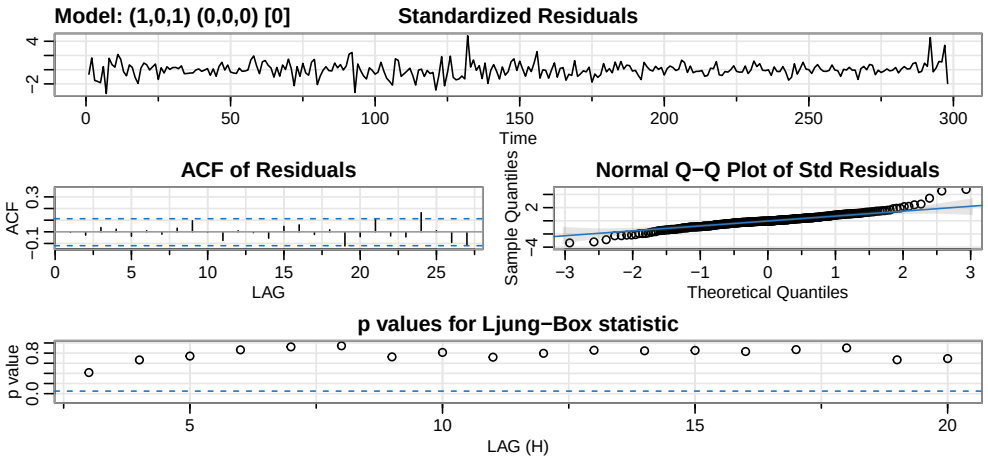

Figure 10: Model diagnostics plots of ARIMA(1,0,1)x(0,0,0)[0]

model selection process as before. See the code appendix for more details. We experiment with different combinations of $p \in \{4, 5, 6\}$ and of $q \in \{0, 1\}$. We end up with $ARIMA(4, 0, 1) \times (0, 0, 0)[0]$

Because the performance of the diagnostics plots is similar to those of the previous three models, which means our model choice here is reasonable, and we do not include them. See the code for plotting them.

6 Model Comparision and Selection

With four carefully chosen models, we need to select a final model that best fits the data. However, it is dangerous to blindly choose an overcomplicated model that can perfectly match the given data because the model may also fit the noise in the data. We only want our model to capture the true underlying pattern of the data to be generalized to unseen data well. Otherwise, our prediction might be too noisy and far away from the actual value. This issue is known as overfitting.

To lessen the overfitting issue as much as possible, we compute and compare each model's AIC, AICc, BIC, and cross-validation error. The first three criterion measures the fitness of a model on the in-sample data while penalizing the complexity of the model. The last one measures the predictability of a model on the out-of-sample data. Then, we will select the model that has the best performance with careful consideration of all four criterions. See the code appendix for more details about the selection process.

We summarize the criterion values of all the models into Table 1:

Table 1: Criterion values of each model

Index	Model	CV.error	AIC	AICc	BIC
1	Expo. smoothing + ARMA 1A	5492.341	8.898896	8.899949	8.988378
2	Expo. smoothing + ARMA 1B	5521.305	8.886808	8.887305	8.950724
3	2nd-order Diff. + ARMA 2A	4507.891	9.231862	9.232136	9.281488
4	2nd-order Diff. + ARMA 2B	4413.298	9.249888	9.250857	9.336733

Interestingly, the models using exponential smoothing have higher cross-validation errors but smaller AIC, AICc, and BIC than those using second-order differencing. It could be that our exponential smoothing model is more overfitting than the second-order differencing model. Because the average difference between cross-validation errors of second-order differencing models and exponential smoothing models is much larger than the the average difference between AIC, AICc, and BIC of two models, we will select one model from two second-order differencing models. Now, since the criterions of model 4 and 5 are very close to each other, we will choose model 3 because it has fewer parameters than model 4 to avoid potential overfitting. The choice here is somewhat arbitrary.

There is a caveat that we extract the AIC, BIC, and AICc of these four models directly from their `sarima()` fitted results. However, the exponential smoothing cannot be directly involved in `sarima()` while the second-order differencing can be. This discrepancy might bias the measured criterions. For the simplicity of this report, we ignore this discrepancy, although we recommend further exploration of it.

7 Final Model

From the model selection process, we choose second-order differencing with ARMA 2A as our final model. We will give the specific formula of our model and its prediction for future ten time points.

7.1 Model Expression

To write out the formula, we denote the original time series, close price, by $\{X_t\}$. Then, our model has the form

$$(1 - \phi_1 B)(\nabla^2 X_t - \mu_t) = (1 + \theta_1 B)W_t$$

where $W_t \sim WN(0, \sigma^2)$ and μ_t is a constant. The estimations of the parameters are in the Table 2.

Table 2: The estimates of the model parameters

Parameters	Estimate	SE
ϕ_1	-0.1085	0.0632
θ_1	-0.9520	0.0269
μ_t	0.0576	0.0677

In Figure 11, we plot the sample and theoretical ACF/PACF of our model. We can see that the sample ACF's are bit off from their theoretical values for all lags except 1, but it might be due the randomness. No sample ACF surpasses the blue band which is the 95% confidence interval. We still think the sample and theoretical ACF match well. Similarly, the sample and theoretical PACF match well.

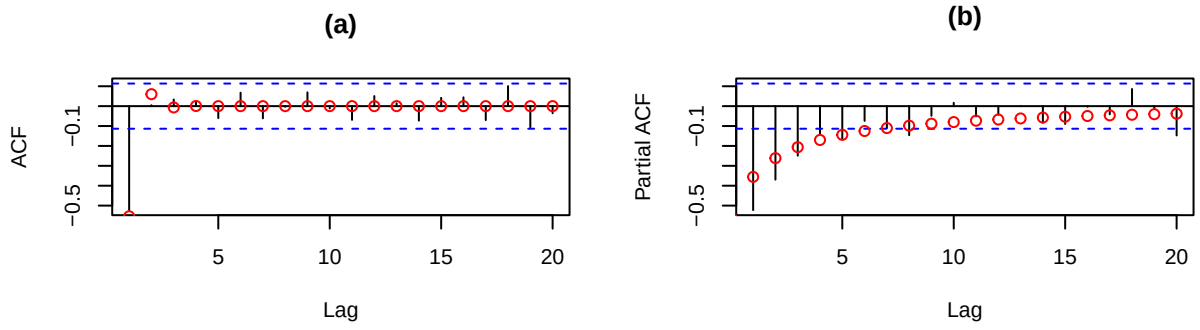

Figure 11: (a) The sample and theoretical ACF's (b) The sample and theoretical PACF's

7.2 Prediction

The time series plot in Figure 12(a) shows the predicted value (red points) of TSLA stock price for the next ten trading days from November 3 and November 16, 2021. Our model predicts a monotonically increasing

trend probably because X_t is monotonically increasing during the last few days as well. The gray areas are 68% and 95% prediction intervals respectively.

To further specify the uncertainty and error of the prediction, we collect the actual close price of Tesla on these ten days and plot them with the predictions and their prediction intervals in Figure 12(b).

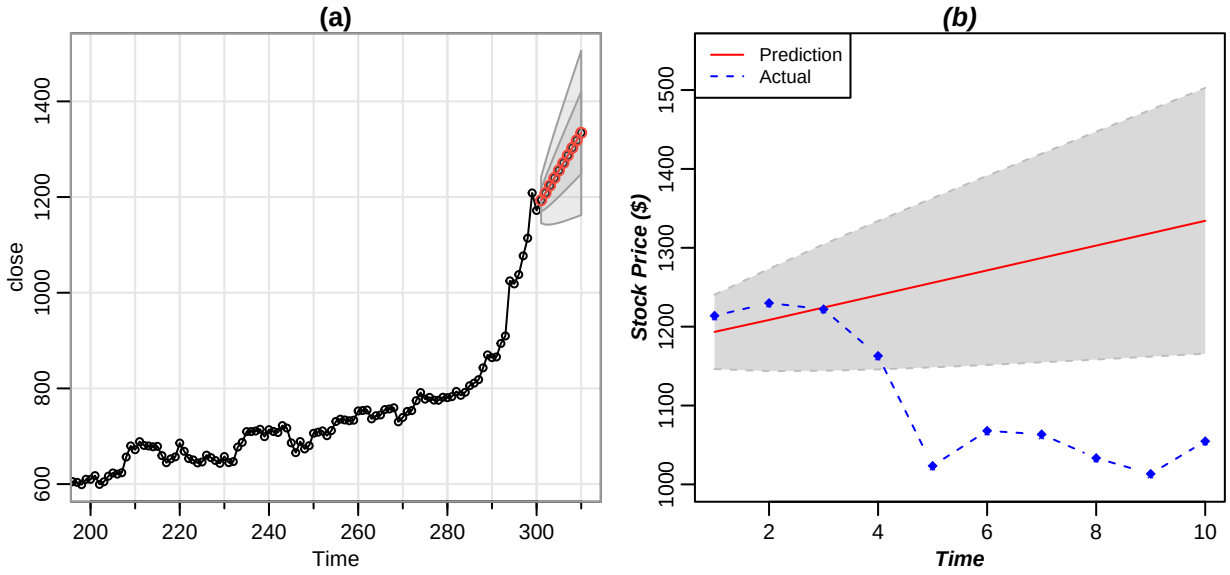

Figure 12: (a): The historical and predicted closed price (b): Predicted and actual close price

Figure 12(b) above displays the comparison between the forecasted and actual values of Tesla stock price in the analyzed period. The 95% prediction interval captures the first four days well. However, our predictions significantly deviate from the actual values because the actual close price sharply decreased from day three to day five and did not increase to the original level. We find out that Elon Mask announced his interest to sell ten percent of his holdings on the weekend of Nov. 6th. It might be the reason for the decrease.

8 Conclusion

While the $ARIMA(1,2,1) \times (0,0,0)[0]$ model works the best among all the models considered, there is no guarantee that this is the true model underlying this dataset. Moreover, as we mentioned in the introduction, our model indeed generates inaccurate predictions. Comparing our model prediction results to the actual stock price changes, we observe that the model has failed to capture the trend and fluctuations in Tesla stock price across time: in stark contrast to the overall increasing trend the model predicts, the stock price dropped after Nov. 4th, 2021, and had only started to go up a week ago. Even if our model is ideal from the point of view of statistical performance, it has failed to take the human factor into account (in this case, Tesla stock prices slump after Elon Musk announced selling 10% of his holdings in the company earlier this month). These conclusions from our time series analysis of Tesla could also apply to the overall stock market, where statistical probability and human behavior are intertwined. Therefore, it is probably an unwise idea to predict the stock price solely and blindly based on a purely statistical and mathematical model.

To design a better methodology of modeling the stock price, we can consider a multivariate time series analysis. As mentioned before, we need to add in the human factors, which might be hard to quantified. We can also add the company's features that are outside of the stock market. For example, we can add the daily number of tweets related to the company in some way. We exepct a multivariate time series model will perform better than a univairate time series model.

Code appendix

```
# Template source: https://github.com/alexpghayes/rmarkdown_homework_template
knitr::opts_chunk$set(
  echo = TRUE, # show code
  eval = FALSE, # don't eval code
  warning = FALSE, # don't show warnings
  message = FALSE, # don't show messages (less serious warnings)
  cache = FALSE, # set to TRUE to save results from last compilation
  fig.align = "center", # center figures
  fig.height = 4
)
library(ggplot2)
library(patchwork)
library(astsa)
library(TSA)
```

```
##
## Attaching package: 'TSA'

## The following objects are masked from 'package:stats':
##
##     acf, arima

## The following object is masked from 'package:utils':
##
##     tar
```

```
library(forecast)
```

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

## Registered S3 methods overwritten by 'forecast':
##   method       from
##   fitted.Arima TSA
##   plot.Arima   TSA

##
## Attaching package: 'forecast'

## The following object is masked from 'package:astsa':
##
##     gas
```

```r
set.seed(153) # make random results reproducible
```

1 Abstract

2 Introduction

3 Data Description

```r
# load data
stock = read.csv("data/TSLA.csv")
stock$Date = as.Date(stock$Date)

# Extract last 300 days
n = 300
t = 1:n
stock_300 = tail(stock, n)
stock_300 = stock_300[c('Date','Close')]
stock_300$t = t
```

4 Exploratory Data Analysis

```r
# Figure 1

# Line plot of all the data
full_data <- ggplot(data = stock, aes(x = Date, y = Close)) +
  geom_line() +
  ylab("Close Price (Dollars)") +
  ggtitle("(a)")

# Line plot of last 300 data
last_300 <- ggplot(data = stock_300, aes(x = Date, y = Close)) +
  geom_line() +
  ylab("Close Price (Dollars)") +
  ggtitle("(b)")

full_data / last_300
```

```r
# Figure 2

# Line plot of last 300 data
last_300 <- ggplot(data = stock_300, aes(x = Date, y = Close)) +
  geom_line() +
  ylab("Close Price (Dollars)") +
  ggtitle("(a)") +
  theme(text = element_text(size = 8)) +
  guides(x =  guide_axis(angle = 90))
```

```r
# Line plot of last 300 data with square root transformation
last_300_sqrt <- ggplot(data = stock_300, aes(x = Date, y = Close)) +
  geom_line() +
  scale_y_sqrt() +
  ylab("Close Price (Dollars)") +
  ggtitle("(b)") +
  theme(text = element_text(size = 8)) +
  guides(x =  guide_axis(angle = 90))

# Line plot of last 300 data with natural log transformation
last_300_log <- ggplot(data = stock_300, aes(x = Date, y = Close)) +
  geom_line() +
  scale_y_log10() +
  ylab("Close Price (Dollars)") +
  ggtitle("(c)") +
  theme(text = element_text(size = 8)) +
  guides(x =  guide_axis(angle = 90))

last_300 + last_300_sqrt + last_300_log
```

5 Model Construction

5.1 Non-parametric Signal Model: exponential smoothing

```r
# Figure 3

par(mfrow = c(1, 2), cex = 0.65, lwd = 0.8)

# exponential filter
alpha = 0.9
lag = 10
filter_weights = alpha^(1:lag)
filter_weights = filter_weights/sum(filter_weights)
filtered_stock = filter(stock_300$Close, filter_weights, sides = 1)

# Plot the original data and fitted values
plot(stock_300$t, stock_300$Close, type = 'l', main = "(a)",
     xlab = "Time", ylab = "Close Price")
lines(stock_300$t, filtered_stock, col = 'red')

filtered_stock = na.omit(filtered_stock)
log_stock = stock_300$Close[-1:-(lag - 1)]
res = log_stock - filtered_stock
plot(res, main = "(b)", xlab = "Time", ylab = "Residuals")
```

```r
# Figure 4

# seasonal differencing
diff_res = diff(res, lag = 5)
```

```r
par(cex = 0.65)
plot(diff_res, main = "(a)", ylab = "Differenced Data")
```

5.1.1 ARMA 1A: $ARIMA(1,0,3) \times (0,0,1)[5]$

```r
# Figure 5

acf2(diff_res)
```

```r
# Model selection process for ARMA 1A

# six attemps will be enought since we don't want an overcomplicated model
attempt1 <- sarima(diff_res, p = 1, d = 0, q = 3, P = 0, D = 0, Q = 0, S = 5)
attempt2 <- sarima(diff_res, p = 1, d = 0, q = 3, P = 1, D = 0, Q = 0, S = 5)
attempt3 <- sarima(diff_res, p = 1, d = 0, q = 3, P = 0, D = 0, Q = 1, S = 5)
attempt4 <- sarima(diff_res, p = 1, d = 0, q = 3, P = 1, D = 0, Q = 1, S = 5)
attempt5 <- sarima(diff_res, p = 1, d = 0, q = 3, P = 2, D = 0, Q = 1, S = 5)
attempt6 <- sarima(diff_res, p = 1, d = 0, q = 3, P = 1, D = 0, Q = 2, S = 5)


start <- 186
cv_1 <- 0
cv_2 <- 0
cv_3 <- 0
cv_4 <- 0
cv_5 <- 0
cv_6 <- 0

# cross-validation using last 100 time points
for (i in 0:9) {
  # train-val set split
  train <- window(diff_res, end = start + i * 10)
  val <- window(diff_res, start = start + i * 10 + 1, end = start + (i + 1) * 10)

  # ARMA prediction
  arma_1_pred <- sarima.for(train, n.ahead = 10,  p = 1, d = 0, q = 3,
                            P = 0, D = 0, Q = 0, S = 5, plot = FALSE)$pred
  arma_2_pred <- sarima.for(train, n.ahead = 10,  p = 1, d = 0, q = 3,
                            P = 1, D = 0, Q = 0, S = 5, plot = FALSE)$pred
  arma_3_pred <- sarima.for(train, n.ahead = 10,  p = 1, d = 0, q = 3,
                            P = 0, D = 0, Q = 1, S = 5, plot = FALSE)$pred
  arma_4_pred <- sarima.for(train, n.ahead = 10,  p = 1, d = 0, q = 3,
                            P = 1, D = 0, Q = 1, S = 5, plot = FALSE)$pred
  arma_5_pred <- sarima.for(train, n.ahead = 10,  p = 1, d = 0, q = 3,
                            P = 2, D = 0, Q = 1, S = 5, plot = FALSE)$pred
  arma_6_pred <- sarima.for(train, n.ahead = 10,  p = 1, d = 0, q = 3,
                            P = 1, D = 0, Q = 2, S = 5, plot = FALSE)$pred

  # compute errors
  cv_1 <- cv_1 + mean((arma_1_pred - val)^2)
  cv_2 <- cv_2 + mean((arma_2_pred - val)^2)
```

```
  cv_3 <- cv_3 + mean((arma_3_pred - val)^2)
  cv_4 <- cv_4 + mean((arma_4_pred - val)^2)
  cv_5 <- cv_5 + mean((arma_5_pred - val)^2)
  cv_6 <- cv_6 + mean((arma_6_pred - val)^2)
}

cv_1 = cv_1 / 10
cv_2 = cv_2 / 10
cv_3 = cv_3 / 10
cv_4 = cv_4 / 10
cv_5 = cv_5 / 10
cv_6 = cv_6 / 10

AIC = c(attempt1$AIC,attempt2$AIC,attempt3$AIC,attempt4$AIC,attempt5$AIC,attempt6$AIC)
BIC = c(attempt1$BIC,attempt2$BIC,attempt3$BIC,attempt4$BIC,attempt5$BIC,attempt6$BIC)
AICc = c(attempt1$AICc,attempt2$AICc,attempt3$AICc,
         attempt4$AICc,attempt5$AICc,attempt6$AICc)
cv = c(cv_1,cv_2,cv_3,cv_4,cv_5,cv_6)

which.min(AIC)
which.min(BIC)
which.min(AICc)
which.min(cv)

# Attempt 3 has lowest AIC,BIC,AICc, and CV errors
```

```
# Figure 6
model_1A <- sarima(diff_res, p = 1, d = 0, q = 3, P = 0, D = 0, Q = 1, S = 5)
```

5.1.2 ARMA 1B: $ARIMA(1,0,1) \times (0,0,1)[5]$

```
# Model Selection Process for ARMA 1B
# model selected by auto.arima
auto.arima(diff_res)

attempt1 <- sarima(diff_res, p = 2, d = 0, q = 2, P = 0, D = 0, Q = 0, S = 5)
# does not pass the Ljung-box test

attempt2 <- sarima(diff_res, p = 2, d = 0, q = 2, P = 1, D = 0, Q = 0, S = 5)
# does not pass the Ljung-box test

attempt3 <- sarima(diff_res, p = 2, d = 0, q = 2, P = 0, D = 0, Q = 1, S = 5)
attempt4 <- sarima(diff_res, p = 1, d = 0, q = 2, P = 0, D = 0, Q = 1, S = 5)
attempt5 <- sarima(diff_res, p = 1, d = 0, q = 1, P = 0, D = 0, Q = 1, S = 5)
attempt6 <- sarima(diff_res, p = 0, d = 0, q = 1, P = 0, D = 0, Q = 1, S = 5)


start <- 186
cv_1 <- 0
cv_2 <- 0
cv_3 <- 0
```

```r
cv_4 <- 0
cv_5 <- 0
cv_6 <- 0

# cross-validation using last 100 time points
for (i in 0:9) {
  # train-val set split
  train <- window(diff_res, end = start + i * 10)
  val <- window(diff_res, start = start + i * 10 + 1, end = start + (i + 1) * 10)

  # ARMA prediction
  arma_1_pred <- sarima.for(train, n.ahead = 10,  p = 2, d = 0, q = 2,
                            P = 0, D = 0, Q = 0, S = 5, plot = FALSE)$pred
  arma_2_pred <- sarima.for(train, n.ahead = 10,  p = 2, d = 0, q = 2,
                            P = 1, D = 0, Q = 0, S = 5, plot = FALSE)$pred
  arma_3_pred <- sarima.for(train, n.ahead = 10,  p = 2, d = 0, q = 2,
                            P = 0, D = 0, Q = 1, S = 5, plot = FALSE)$pred
  arma_4_pred <- sarima.for(train, n.ahead = 10,  p = 1, d = 0, q = 2,
                            P = 0, D = 0, Q = 1, S = 5, plot = FALSE)$pred
  arma_5_pred <- sarima.for(train, n.ahead = 10,  p = 1, d = 0, q = 1,
                            P = 0, D = 0, Q = 1, S = 5, plot = FALSE)$pred
  arma_6_pred <- sarima.for(train, n.ahead = 10,  p = 0, d = 0, q = 0,
                            P = 0, D = 0, Q = 1, S = 5, plot = FALSE)$pred

  # compute errors
  cv_1 <- cv_1 + mean((arma_1_pred - val)^2)
  cv_2 <- cv_2 + mean((arma_2_pred - val)^2)
  cv_3 <- cv_3 + mean((arma_3_pred - val)^2)
  cv_4 <- cv_4 + mean((arma_4_pred - val)^2)
  cv_5 <- cv_5 + mean((arma_5_pred - val)^2)
  cv_6 <- cv_6 + mean((arma_6_pred - val)^2)
}

cv_1 = cv_1 / 10
cv_2 = cv_2 / 10
cv_3 = cv_3 / 10
cv_4 = cv_4 / 10
cv_5 = cv_5 / 10
cv_6 = cv_6 / 10

AIC = c(attempt1$AIC,attempt2$AIC,attempt3$AIC,attempt4$AIC,attempt5$AIC,attempt6$AIC)
BIC = c(attempt1$BIC,attempt2$BIC,attempt3$BIC,attempt4$BIC,attempt5$BIC,attempt6$BIC)
AICc = c(attempt1$AICc,attempt2$AICc,attempt3$AICc,
         attempt4$AICc,attempt5$AICc,attempt6$AICc)
cv = c(cv_1,cv_2,cv_3,cv_4,cv_5,cv_6)

which.min(AIC)
which.min(BIC)
which.min(AICc)
which.min(cv)

# Attempt 5 have lowest AIC, BIC, and AICc and Attempt 3 has lowest CV error. However,
# the difference between the CV errors of attempt 3 and 5 is only 0.2, so we choose
```

```r
# attempt 5
```

```r
# Figure 7

model_1B <- sarima(diff_res, p = 1, d = 0, q = 1, P = 0, D = 0, Q = 1, S = 5)
```

5.2 Non-parametric Signal Model: second-order differencing

```r
# Figure 8

par(mfrow = c(1, 2), cex = 0.65, lwd = 0.5)

# first order differencing
stock_d = diff(stock_300$Close)
# second order differencing
stock_dd = diff(stock_d)

plot(stock_d, type = 'l', main = "(a)", ylab = "Differenced Data")
plot(stock_dd, type = 'l', main = "(b)", ylab = "Differenced Data")
```

5.2.1 ARMA 2A: $ARIMA(1,0,1) \times (0,0,0)[0]$

```r
# Figure 9

acf2(stock_dd)
```

```r
# Figure 10

# model diagnostics plots of ARMA 2A
auto.arima(stock_dd, start.q = 1)

model_2A <- sarima(stock_dd, p=1, d=0, q=1, P=0, D=0, Q=0, S=0)
```

5.2.2 ARMA 2B:

```r
# Model selection process for ARMA 2B

attempt1 <- sarima(stock_dd, p = 4, d = 0, q = 0, P = 0, D = 0, Q = 0, S = 0)
# does not pass the Ljung-box test

attempt2 <- sarima(stock_dd, p = 5, d = 0, q = 0, P = 0, D = 0, Q = 0, S = 0)
# does not pass the Ljung-box test

attempt3 <- sarima(stock_dd, p = 6, d = 0, q = 0, P = 0, D = 0, Q = 0, S = 0)
attempt4 <- sarima(stock_dd, p = 4, d = 0, q = 1, P = 0, D = 0, Q = 0, S = 0)
attempt5 <- sarima(stock_dd, p = 5, d = 0, q = 1, P = 0, D = 0, Q = 0, S = 0)
attempt6 <- sarima(stock_dd, p = 6, d = 0, q = 1, P = 0, D = 0, Q = 0, S = 0)
```

```r
start <- 200
cv_1 <- 0
cv_2 <- 0
cv_3 <- 0
cv_4 <- 0
cv_5 <- 0
cv_6 <- 0

# cross-validation using last 100 time points
for (i in 0:9) {
  # train-val set split
  train <- window(diff_res, end = start + i * 10)
  val <- window(diff_res, start = start + i * 10 + 1, end = start + (i + 1) * 10)

  # ARMA prediction
  arma_1_pred <- sarima.for(train, n.ahead = 10,  p = 4, d = 0, q = 0,
                            P = 0, D = 0, Q = 0, S = 0, plot = FALSE)$pred
  arma_2_pred <- sarima.for(train, n.ahead = 10,  p = 5, d = 0, q = 0,
                            P = 0, D = 0, Q = 0, S = 0, plot = FALSE)$pred
  arma_3_pred <- sarima.for(train, n.ahead = 10,  p = 6, d = 0, q = 0,
                            P = 0, D = 0, Q = 0, S = 0, plot = FALSE)$pred
  arma_4_pred <- sarima.for(train, n.ahead = 10,  p = 4, d = 0, q = 1,
                            P = 0, D = 0, Q = 0, S = 0, plot = FALSE)$pred
  arma_5_pred <- sarima.for(train, n.ahead = 10,  p = 5, d = 0, q = 1,
                            P = 0, D = 0, Q = 0, S = 0, plot = FALSE)$pred
  arma_6_pred <- sarima.for(train, n.ahead = 10,  p = 6, d = 0, q = 1,
                            P = 0, D = 0, Q = 0, S = 0, plot = FALSE)$pred

  # compute errors
  cv_1 <- cv_1 + mean((arma_1_pred - val)^2)
  cv_2 <- cv_2 + mean((arma_2_pred - val)^2)
  cv_3 <- cv_3 + mean((arma_3_pred - val)^2)
  cv_4 <- cv_4 + mean((arma_4_pred - val)^2)
  cv_5 <- cv_5 + mean((arma_5_pred - val)^2)
  cv_6 <- cv_6 + mean((arma_6_pred - val)^2)
}

cv_1 = cv_1 / 10
cv_2 = cv_2 / 10
cv_3 = cv_3 / 10
cv_4 = cv_4 / 10
cv_5 = cv_5 / 10
cv_6 = cv_6 / 10

AIC = c(attempt1$AIC,attempt2$AIC,attempt3$AIC,attempt4$AIC,attempt5$AIC,attempt6$AIC)
BIC = c(attempt1$BIC,attempt2$BIC,attempt3$BIC,attempt4$BIC,attempt5$BIC,attempt6$BIC)
AICc = c(attempt1$AICc,attempt2$AICc,attempt3$AICc,
         attempt4$AICc,attempt5$AICc,attempt6$AICc)
cv = c(cv_1,cv_2,cv_3,cv_4,cv_5,cv_6)

which.min(AIC)
which.min(BIC)
```

```
which.min(AICc)
which.min(cv)

# Attempt 4 have lowest AIC, BIC, and AICc and Attempt 1 has lowest CV error. However,
# Attempt 1 has a very bad performance of Ljung-box statistic. Thus, we will choose attempt 4
```

```
# model diagnostics plots of ARMA 2B
model_2B <- sarima(stock_dd, p=4, d=0, q=1, P=0, D=0, Q=0, S=0)
```

6 Model Comparision and Selection

```
# Here we compute the cross-validation error of each model. We only use the last one hundred
# data points, which is 1/3 of the entire data, to select the validation set. There are
# ten validation sets, and each of them contains ten data points. The whole process is
# very similar to the classical k-fold cross-validation. In the first iteration, the training
# set contains the time points 1:200, and the validation set contains the time points 201:210.
# Then, in the second iteration, the training set contains the time points 1:210, and
# the validation set contains the time points 211:220. We repeat the similar process, and in
# the last iteration, the training set contains the time points 1:290, and the validation set
# contains the time points 291:300.
start <- 200
cv_1A <- 0
cv_1B <- 0
cv_2A <- 0
cv_2B <- 0

for (i in 0:9) {
  # train-val set split
  train <- window(stock_300$Close, end = start + i * 10)
  val <- window(stock_300$Close, start = start + i * 10 + 1, end = start + (i + 1) * 10)

  # get stationary residuals
  filtered_train = filter(train, filter_weights, sides = 1)
  filtered_train = na.omit(filtered_train)
  cut_train <- train[-1:-(lag - 1)]
  exp_res = cut_train - filtered_train

  # exponential smoothing prediction
  exp_pred = numeric(10)
  values = tail(train, 10) # store the values for prediction
  for (i in 1:10){
    exp_pred[i] = sum(tail(values, 10) * rev(filter_weights))
    values <- append(values, exp_pred[i])
  }

  # ARMA prediction
  arma_1A_pred <- sarima.for(exp_res, n.ahead = 10,  p = 1, d = 0, q = 3,
                             P = 0, D = 0, Q = 1, S = 5, plot = FALSE)$pred
  arma_1B_pred <- sarima.for(exp_res, n.ahead = 10,  p = 1, d = 0, q = 1,
                             P = 0, D = 0, Q = 1, S = 5, plot = FALSE)$pred
```

```
  # total prediction
  pred_1A <- exp_pred + arma_1A_pred
  pred_1B <- exp_pred + arma_1B_pred
  pred_2A <- sarima.for(train, n.ahead = 10,  p = 1, d = 2, q = 1,
                        P = 0, D = 0, Q = 0, S = 0, plot = FALSE)$pred
  pred_2B <- sarima.for(train, n.ahead = 10,  p = 4, d = 2, q = 1,
                        P = 0, D = 0, Q = 0, S = 0, plot = FALSE)$pred

  # compute errors
  cv_1A <- cv_1A + mean((pred_1A - val)^2)
  cv_1B <- cv_1B + mean((pred_1B - val)^2)
  cv_2A <- cv_2A + mean((pred_2A - val)^2)
  cv_2B <- cv_2B + mean((pred_2B - val)^2)
}

cv_1A = cv_1A / 10
cv_1B = cv_1B / 10
cv_2A = cv_2A / 10
cv_2B = cv_2B / 10
```

```
# Table 1

index = c(1, 2, 3, 4)
model = c("Expo. smoothing + ARMA 1A", "Expo. smoothing + ARMA 1B",
          "2nd-order Diff. + ARMA 2A","2nd-order Diff. + ARMA 2B")
cves = c(cv_1A, cv_1B, cv_2A, cv_2B)
aic = c(model_1A$AIC,model_1B$AIC,model_2A$AIC,model_2B$AIC)
aicc = c(model_1A$AICc,model_1B$AICc,model_2A$AICc,model_2B$AICc)
bic = c(model_1A$BIC,model_1B$BIC,model_2A$BIC,model_2B$BIC)
s <- data.frame(Index = index, "Model" = model, "CV error"=cves,
                "AIC"=aic,"AICc"=aicc,"BIC"=bic)
knitr::kable(s, caption = "Criterion values of each model")
```

```
# Figure 11

par(mfrow = c(1, 2), cex = 0.7)
model_2A.acf=ARMAacf(ar=c(-0.1085), ma=c(-0.9520), lag.max = 20, pacf=FALSE)
acf(as.vector(stock_dd), lag.max=20, main = "(a)")
points(0:20, model_2A.acf, col='red')

model_2A.pacf=ARMAacf(ar=c(-0.1085), ma=c(-0.9520), lag.max = 21, pacf=TRUE)
pacf(as.vector(stock_dd), lag.max=20, main = "(b)")
points(0:20, model_2A.pacf, col='red')
```

7 Final Model

7.1 Model Expression

```
# Table 2
model_2A$ttable
```

7.2 Prediction

```
# Code for plotting Figure 12

par(mfrow = c(1, 2), cex = 0.7)
close = stock_300$Close
model_forecast <- sarima.for(close, n.ahead = 10, p = 1, d = 2, q = 1,
                        P = 0, D = 0, Q = 0, S = 0)
title(main = "(a)")

# compute the prediction intervals
model_pred <- model_forecast$pred
model_se <- model_forecast$se
lower <- c()
upper <- c()
for (i in 1:10){
  lower[i] <- model_pred[i] - 1.96 * model_se[i]
  upper[i] <- model_pred[i] + 1.96 * model_se[i]
}

# collect actual data
stock_new = read.csv("data/TSLA_new.csv")
stock_new_true <- tail(stock_new, 10)
stock_new_true = stock_new_true[c('Date','Close')]
df <- data.frame(stock_new_true$Date, model_pred, stock_new_true$Close, lower, upper)
colnames(df)=c("Time", "Prediction", "True", "Lower", "Upper")

# plot the prediction, actual value, and prediction interval
plot(1:10, xlab = "Time", ylab = "Stock Price ($)", ylim=c(1000,1550), main = "(b)",
     sub = "(Nov 3, 2021 - Nov 16, 2021)",
  # Titles in italic and bold
  font.main=4, font.lab=4, font.sub=4,
  # Change font size
  cex.main=1.2, cex.lab=1, cex.sub=0.75)
polygon(c(rev(1:10), 1:10), c(rev(df[ ,4]), df[ ,5]), col = 'grey85', border = NA)
lines(1:10, df[ ,4], lty = 'dashed', col = 'gray')
lines(1:10, df[ ,5], lty = 'dashed', col = 'gray')
lines(1:10, df[ ,2], type = "l", pch = 19, col = "red")
points(1:10, stock_new_true$Close, pch = "*", col = "blue")
lines(1:10, stock_new_true$Close, pch = 18, col = "blue", type = "b", lty = 2)
legend("topleft", legend=c("Prediction", "Actual"),
       col=c("red", "blue"), lty = 1:2, cex=0.8)
```