Code appendix

1 Abstract

2 Introduction

3 Data Description

```
# 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

```
# 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
```

```
# 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))

# 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

```
# 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)
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
# 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: $ARIMA(1,0,1) \times (0,0,1)[5]$

```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
```

```r
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 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

```r
# 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-ord
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")
```

7 Final Model

7.1 Model Expression

7.2 Prediction

```
# Code for plotting Figure 11

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)
```