Shall We Buy or Sell?
– a time series perspective of stock investment

Group 2: Bingyue Su, Yihao Fang, and Vicky Zhu

December 2021

1 Introduction

We are new to stock, and our group is entranced by the passion for exploring new investment opportunities. As an enthusiastic group in a science field, we are also amazed by the dynamics of stock trends. One of the reasons we question this is that every time there is a major event, the stock market responds with some significant ups and downs. For example, over the last decade, we noticed some big events such as the housing bubbles back in 2008, the elections in 2016, and the recent COVID-19 outbreaks have also changed the financial market considerably. Firms switched their marketing strategies accordingly while still attempting to attract new blood to their product line. This leaves much room for understanding the stock trends and their association with relevant events. How exciting if one can give an informed trade prediction before the closing approaches! Although "many have tried, but most have failed, to predict the stock market's ups and downs" [Marjanovic(2018)], we would like to give a try and use our class knowledge and beyond to build an informative statistical model.

In particular, we are interested in the Amazon and Walmart daily stock since they are closely related as retail firms but targeting different market (online versus physical stores). We obtained the data from 01/01/2006 to 01/01/2018 in the Kaggle DJIA 30 Stock Time Series competition in 2018. Hopefully our model can be useful for the investment field.

2 Method

We started our analysis by exploring Amazon and Walmart time series plots and built simple models that we learned from the class, then we explored other sophisticated methods beyond the scope of the classroom and refined our models.

2.1 ARIMA Modeling

Amazon Data: We see a constantly going up trend with the increasing variance. These are the signals of non-stationary. Although the seasonality is not

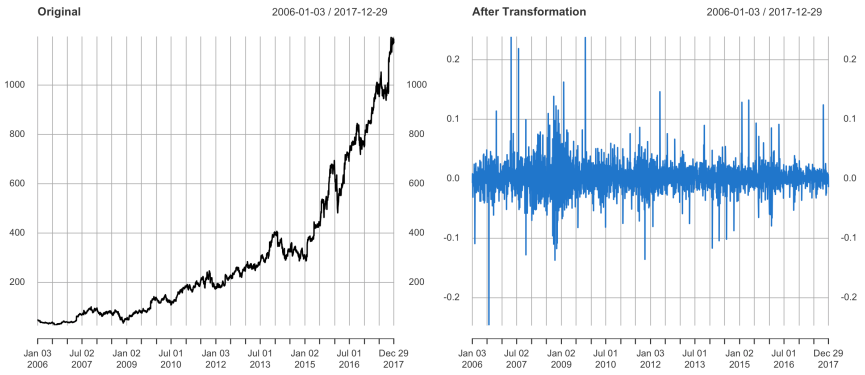

Figure 1: **Time series plots:** Left (non-stationary): original Amazon stock time series plot. Right (stationary): after taking a log transformation and first difference plot.

obvious, we decided to take the first difference after the log transformation to make it stationary (Figure1).

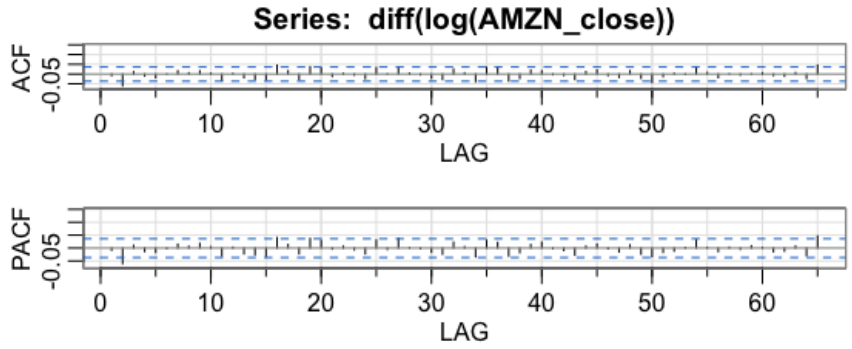

Figure 2: **ACF and PACF of the stationary AMZN data.** ACF (top) and PACF (bottom) indicated that the second lag is significant.

To identify a suitable model for the mean, we checked the auto correlation function (ACF) and partial ACF (PACF) plots (Figure2). Noticed the significance of the second lags, so we can try some standard auto-regressive integrated moving average (ARIMA) models such as ARIMA (2,1,0), ARIMA (0,1,2), and ARIMA (2,1,2) [Robert and Stoffer(2017)]. After a few attempts, we decided to fit it with an ARIMA(0,1,2). The diagnostic plots in Figure 3 indicate ARIMA(0,1,2) is a good model. Although the QQ plot showed a flat tail, due to the nature of financial data, it is common to have a t-distribution residual.

We reserved the last half year (07/01-01/01/2018) data as a validation test-

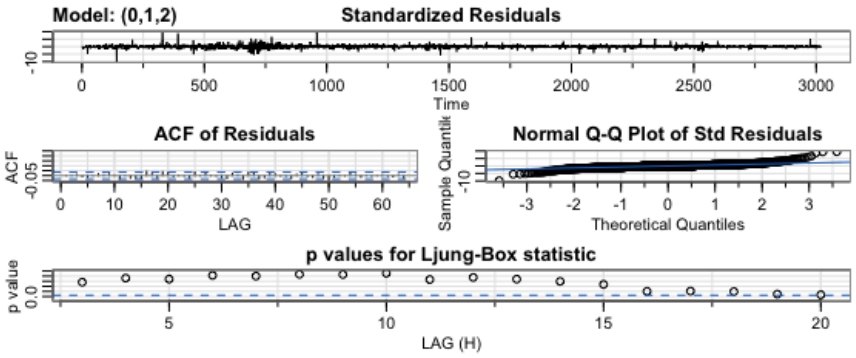

Figure 3: **ARIMA(0,1,2) Diagnostic Plot.**

ing set and compared ARIMA fit with double exponential smoothing (DES) method for the forecasts. We calculated prediction errors as the root of mean squared error, $RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$, where $m = 1, 2, \ldots, 126$ is the half year size, y is the prediction, and $\hat{y}$ is the true stock value. Noticed that ARIMA model has a smaller prediction error (58.43 *versus* 68.06), whereas the plot showed the confident bend width of DES is much smaller.

Figure 4: **Prediction Comparison.** Black: original time series from 09/01/2016 to 01/01/2018. Red: ARIMA (0,1,2) fit. Green: DES predictions.

Amazon stock data is rather a simple example in the realm of stocks, next we consider Walmart stock. As the competitor of Amazon, it has a more realistic setting since the stock has several periods of consistent going ups and down.

Walmart Data: Similar to the analysis in Amazon stock, we plotted Walmart's time series and its transformation in Figure 5. In addition to the overall increasing trend, we noticed other noteworthy patterns such as the appear-

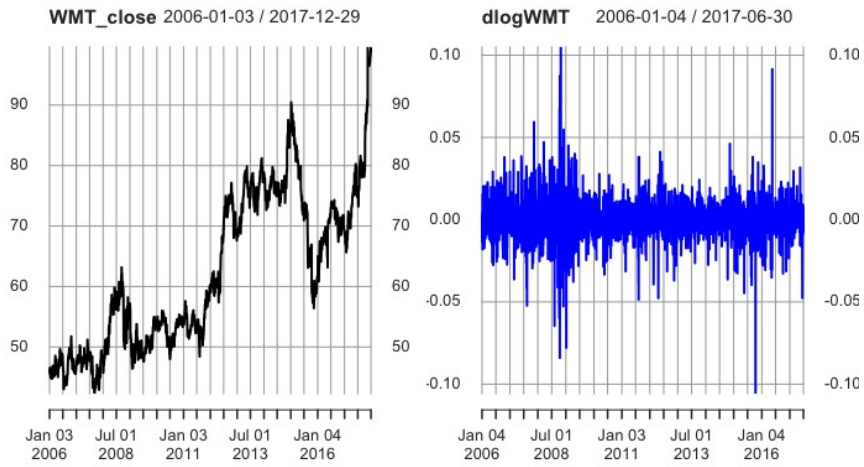

Figure 5: **Time series plots.** Same as in Figure 1, except it is Walmart stock.

ance of heteroscedasticity and several abrupt changes at different time points. While the ACF and PACF plots did not give much information, we tried several

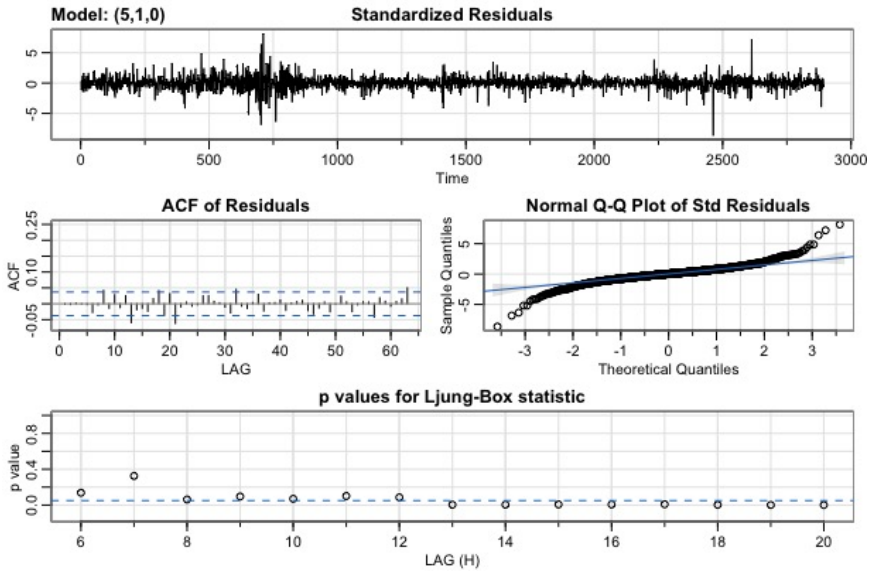

Figure 6: **Diagnostic plots of ARIMA(5,1,0)**

ARIMA models and decided to fit the mean with ARIMA (5,1,0). In Figure 6, we saw the residuals deviate from normal distribution and many p-values are on the rejection line. We believe residuals still contain information, which

enlightened us to explore generalized auto-regressive conditional heteroskedasticity (GARCH) models for the variance [Robert and Stoffer(2017)].

2.2 GARCH Modeling

Upon ARIMA(5,1,0) fit for the mean, we used GARCH(1,1) for the variance. The ACF and QQ plots in Figure 7 indicated a GARCH effect for our model. The residuals are uncorrelated and accord with t-distribution.This indeed reflects the characteristics of financial data. For the completeness, we fitted

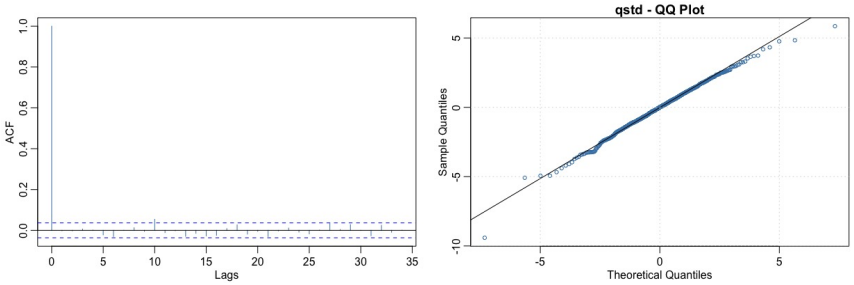

Figure 7: **GARCH fit for Walmart Stock.** ACF and QQ plots of Residuals.

GARCH model to Amazon data, but did not see reductions on prediction error (61.01), so the simple ARIMA is sufficient.

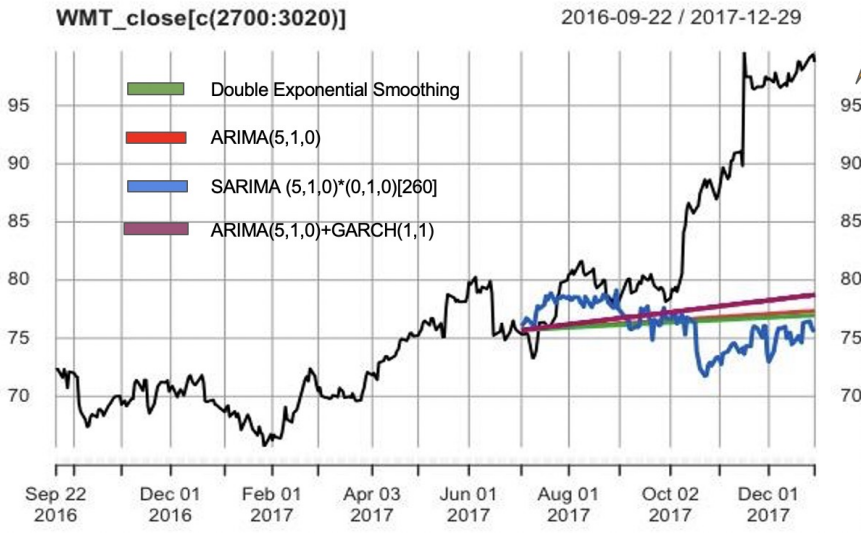

Figure 8: **Prediction.** Prediction of ARIMA, GARCH, SARIMA and DES.

In comparison of the prediction performance, ARIMA, GARCH and DES

model in Figure 8 can only capture the overall increasing prediction performance, but not the abrupt increments. We further assumed a period of 260 (52 weeks $\times$ 5 weekdays) and fitted a SARIMA model. Although this model can capture some dynamics, it could not forecast the skyrocket moment in 2017. In stock data, some significant changes may be affected by other complex factors that are difficult to predict based on the previous patterns. Among all the traditional models, The ARIMA(5,1,0)+GARCH(1,1) model gives the best accuracy (RMSE at 10.79). For the prediction in a detailed and microscope level, we dig into other methods beyond the wall of our classroom.

2.3 GAM Modeling

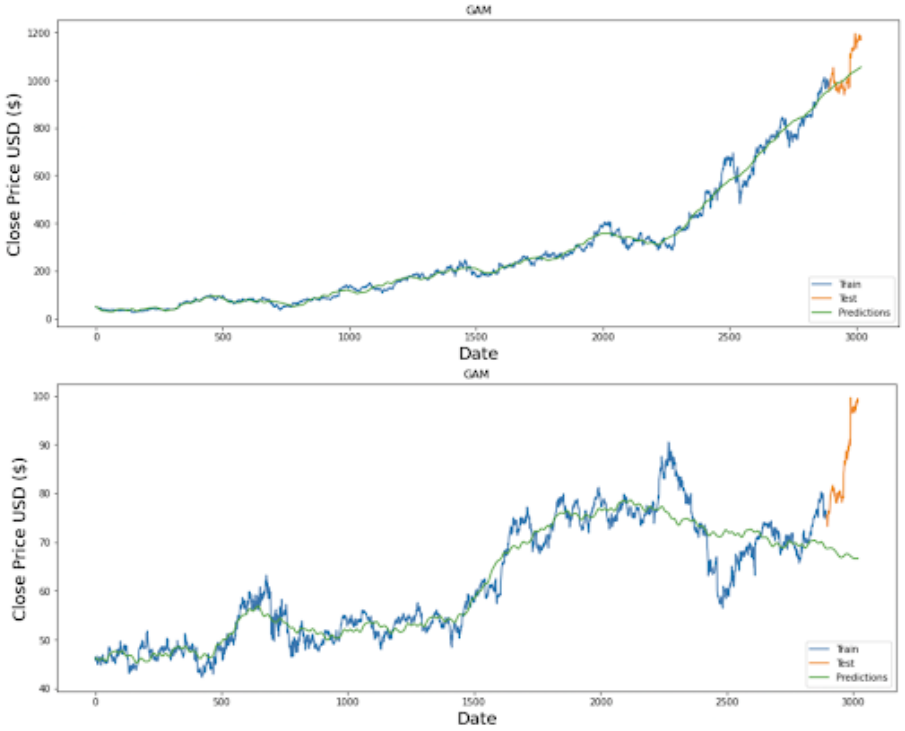

Figure 9: **GAM in Amazon and Walmart stocks forecasting.** Blue: training set. Orange curve: test set. Green: predictions.

Generalized additive model (GAM) [Taylor and Letham(2018)] has the regression form in time t,

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t.$$

Here $g(t)$ is a trend function which models non-periodic changes, $s(t)$ represents periodicity, and $h(t)$ captures the effects of potentially irregular schedules such

as holiday. The error term ϵ_t represents any idiosyncratic changes which are not accommodated by the model. Implementing GAM in python using Prophet package, we picked a piece-wise linear function $g(t)$ to capture the trend, Fourier series $s(t)$ to describe the seasonality, and identity function to grab the holiday effects. With the assumption of normal ϵ_t, we used the default priors and hyper-parameters in the model. To achieve the maximum likelihood function, we evaluated in a L-BFGS optimization method.

As shown in Figure 9, GAM is able to capture the overall trend in Amazon data. The predictions look linear because the seasonal part is not significant within Amazon stock. However, the GAM predictions fail in Walmart stock since the decreasing forecasting values go in the opposite direction in comparing to the truly increasing trend. We hypothesized that this is related to the previous training trend period that our piece-wise linear function detects, in which GAM predicts consistent decreasing values from the last period of the decreasing trend in 2015.

2.4 LSTM Modeling

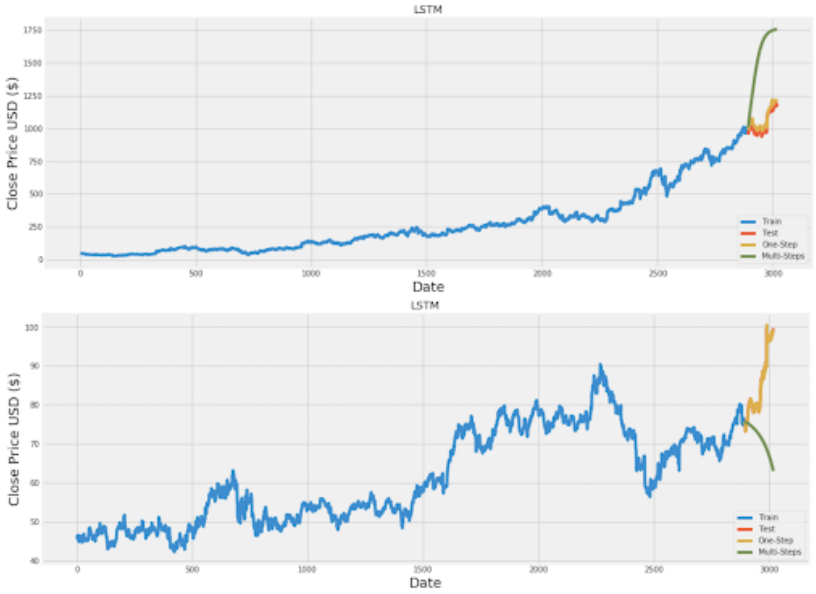

Figure 10: **LSTM model in Amazon and Walmart stocks forecasting**. Blue: training set. Orange: test set. Green: predictions.

We also attempted a long short term memory (LSTM) model in a recurrent neural network (RNN) [Sak et al.(2014)Sak, Senior, and Beaufays], where each layer in the model shares the same structure and takes the inputs x_t sequentially. For example, in $i-th$ chunk, we have a hidden state h_{i-1} and a cell state C_{i-1} inheriting from the previous chunk. To update current h_i and C_i, we combined

the previous hidden state h_{t-1} and current observation x_i in different gates (i.e. the input, the forget, and the outcome gate). After update, the information proceeds to the next chunk. We employed a Pytorch package in Python and established a LSTM model with two layers. We also set up the window length to be 60, which forecasts the current price using previous 60 values.

We predicted the stock prices in one-step and multi-steps shown in Figure 10. The one-step prediction is considerably close to the true value, especially for Walmart data, while the multi-steps predictions are far away from the true values due to error accumulation. This indicates that our LSTM model is powerful to forecast a short period of time window but powerless for a longer time.

3 Discussion

Amazon stock has a pretty consistent increasing trend that we can use a simple ARIMA model for prediction, so I will probably keep this stock for the long-term return; whereas Walmart stock has a general increasing trend but with a big variability under several influences. Although for the neighboring points, we could use SARIMA or LSTM(fewer-step) for a better predictions, if considering longer window of the time, none of the methods can hold its accuracy! I will probably better off by doing some short-term management for this stock.

Fun Fact: in 2015, Walmart has its technology reformed. it rebuilt their e-commerce and delivery system. It also increased 1.5 billion in labour wage spending. [Taylor and Letham(2018)]. As a consequence of large spending, the earning decreased in the next following years. Such strategy of moving towards more online stores does give them the stock skyrockets phenomenon in 2017. Similarly, Amazon also came up with its strategy of moving more offline stores such as the purchase of Whole Foods in 2017, opened up Amazon go, fresh, book, etc.

In our opinions, in responding to the market's condition and people's need, while these two firms are monopolies in their own territories, they are more and more alike and acting like retail empires that keep expending and invading each others' boundaries. So the retail war will continue remaining, and this competition is good to break the monopoly capitalism and keeps our market in a healthier space.

Future Direction: for sophisticated stocks like WAMT, we may consider other models such as applying an intervention model by setting several different time thresholds, discovering other correlated elements that can be incorporated into ARIMAX model, and redeveloping a deeper level of neural networks or reconsidering other activation function and smoothing techniques for the refinement.

4 Appendix

```
library(xts)
library(astsa)
library(forecast)
########### Amazon Stock ###########
AMZN=read.table("AMZN_2006-01-01_to_2018-01-01.csv",
                sep=',',head=TRUE)
par(mfrow=c(1,2))
AMZN_close=as.xts(AMZN$Close,as.Date(AMZN$Date))
plot(AMZN_close) # increasing variance
WMT=read.table("WMT_2006-01-01_to_2018-01-01.csv",
                sep=',',head=TRUE)
WMT_close=as.xts(WMT$Close,as.Date(WMT$Date))
plot(WMT_close)

acf2(AMZN_close)
######################################## end of first time series plot
## TS Plot and Cleaning
par(mfrow=c(1,2))
plot(AMZN_close, main = "Original")
plot(diff(log(AMZN_close)), col=4, main = "After Transformation")

par(mfrow=c(1,1))
plot(log(AMZN_close))
acf2(log(AMZN_close)) # need to take a first order difference

plot(diff(diff(log(AMZN_close))),260)
######################################## end of second time series plot and cleaning

acf2(diff(log(AMZN_close)))

auto.arima(log(AMZN_close))

AMZN_arima_fit101<-sarima(log(AMZN_close),0,1,0)

AMZN_arima_fit011<-sarima(log(AMZN_close),0,1,1)
AMZN_arima_fit110<-sarima(log(AMZN_close),1,1,0)
AMZN_arima_fit111<-sarima(log(AMZN_close),1,1,1)

AMZN_arima_fit012<-sarima(log(AMZN_close),0,1,2)
AMZN_arima_fit210<-sarima(log(AMZN_close),2,1,0)
AMZN_arima_fit112<-sarima(log(AMZN_close),1,1,2)
AMZN_arima_fit211<-sarima(log(AMZN_close),2,1,1)
AMZN_arima_fit212<-sarima(log(AMZN_close),2,1,2)
```

```
######################################### end of acf analysis

sarima(log(AMZN_close),1,1,1)
AMZN_arima_fit111$ttable

sarima(log(AMZN_close),0,1,2)
AMZN_arima_fit012$ttable

######################################### end of model comparisons
par(mfrow=c(1,1))
#AMZN_pred111<-sarima.for(log(AMZN_close), n.ahead = 10, 1,1,1)
logAMZN_pred012<-sarima.for(log(AMZN_close[c(1:2893),]), n.ahead = 127, 0,1,2)
plot(exp(logAMZN_pred012$pred))

### putting them into one plot AND present a partial plot
time_pred=as.Date(WMT$Date[c(2893:3019)])
AMZN_fit2=as.xts(as.vector(logAMZN_pred012$pred), time_pred)
AMZN_fit2_up=as.xts(as.vector(logAMZN_pred012$pred+logAMZN_pred012$se), time_pred)
AMZN_fit2_down=as.xts(as.vector(logAMZN_pred012$pred-logAMZN_pred012$se), time_pred)

plot(AMZN_close[c(2700:3019)], type = "l", main = "amzon-close")
lines(exp(AMZN_fit2),col=2, lwd = 3)
lines(AMZN_fit1,col=3, lwd = 3)
lines(exp(AMZN_fit2_up),col=2)

# Double exponential prediction
DoubleES <-  HoltWinters(AMZN_close[c(1:2893),], gamma = FALSE)
predictedAMZN <- forecast(DoubleES, 127)
plot(predictedAMZN)

AMZN_fit1=as.xts(as.vector(predictedAMZN$mean), time_pred)
AMZN_fit1_up=as.xts(as.vector(predictedAMZN$upper[,2]), time_pred)
AMZN_fit1_down=as.xts(as.vector(predictedAMZN$lower[,2]), time_pred)
plot(AMZN_fit1)

### putting them into one plot AND present a partial plot
plot(AMZN_close[c(2700:3019)], type = "l", main = "amzon-close")
lines(AMZN_fit1,col=2, lwd = 3)
lines(AMZN_fit1_up,col=4)
lines(AMZN_fit1_down,col=4)


### putting everything into one plot
plot(AMZN_close[c(2700:3019)], type = "l", main = "amzon-close")
lines(exp(AMZN_fit2),col=2, lwd = 3)
lines(AMZN_fit1,col=3, lwd = 3)
```

```
lines(exp(AMZN_fit2_up),col=2)
lines(exp(AMZN_fit2_down),col=2)
lines(AMZN_fit1_up,col=3)
lines(AMZN_fit1_down,col=3)

### Prediction errors:
true_value = AMZN$Close[c(2893:3019)]
error1 = sqrt(sum(as.numeric(exp(logAMZN_pred012$pred)-true_value)^2)/127)
error2 = sqrt(sum((predictedAMZN$mean-true_value)^2)/127)
######################################## end of model forcastings

AMZN_res = AMZN_arima_fit012$fit$residuals
par(mfrow=c(1,2))
acf2(AMZN_res, max.lag = 20)
acf2(AMZN_res^2, max.lag = 20) # indicate
library(FinTS)
ArchTest(diff(log(AMZN_close))) # significant

dlogAMZN = diff(log((AMZN$Close)))
time_AMZN<- index(dlogAMZN)

Garch_AMZN=garchFit(~arma(0,2)+garch(1,1),cond.dist = "std",data=dlogAMZN)
summary(Garch_AMZN)

# the fitted values
par(mfrow=c(1,1))
fitted_value <- Garch_AMZN@fitted
plot(time_AMZN, dlogAMZN, type = "l", main = "Amazon-return")
lines(time_AMZN, fitted_value, col="red")

garch_AMZN1<-predict(Garch_AMZN, n.ahead = 127)
#garch_AMZN2<-forecast(Garch_AMZN, n.ahead = 127)

d = garch_AMZN1$meanForecast
exp(diffinv(d, xi = 1))

###another garch model
library(rugarch)
spec <- ugarchspec(variance.model=list(modl = "sGARCH",
garchOrder = c(1, 1)),
mean.model = list(armaOrder = c(0,2)),
distribution.model = "std")
# without fixed parameters here
fit <- ugarchfit(spec, data = dlogAMZN)
pred_garch=ugarchforecast(fit,dlogAMZN,n.ahead=127)
pred_mean=pred_garch@forecast$seriesFor
```

```
pred_sd=pred_garch@forecast$sigmaFor


pred_AMZN=rep(0,126)
pred_upper=rep(0,126)
pred_lower=rep(0,126)
for(i in 1:127){
  if(i==1){
    pred_AMZN[i]=log(AMZN$Close[2893])+pred_mean[i]
    pred_upper[i]=log(AMZN$Close[2893])+pred_mean[i]+pred_sd[i]
    pred_lower[i]=log(AMZN$Close[2893])+pred_mean[i]-pred_sd[i]
  }else{
    pred_AMZN[i]=pred_AMZN[i-1]+pred_mean[i]
    pred_upper[i]=pred_upper[i-1]+pred_mean[i]+pred_sd[i]
    pred_lower[i]=pred_lower[i-1]+pred_mean[i]-pred_sd[i]
  }
}
pred_AMZN=exp(pred_AMZN)
pred_upper=exp(pred_upper)
pred_lower=exp(pred_lower)

pred_upper=as.xts(pred_upper,time_pred)
pred_lower=as.xts(pred_lower,time_pred)
pred_AMZN=as.xts(pred_AMZN,time_pred)

par(mfrow=c(1,1))
plot(AMZN_close[c(2700:3019)], type = "l", main = "amzon-close with ARIMA and GARCH")
lines(pred_AMZN,col=3,lwd=3)
#lines(pred_upper,col=3,lwd=3)
#lines(pred_lower,col=3,lwd=3)
error3 = sqrt(sum((pred_AMZN-true_value)^2)/127)

######################################### end of GARCH model detaction

############## Walmart Stock ############
rm(list=ls())
WMT=read.table("WMT_2006-01-01_to_2018-01-01.csv",sep=',',head=TRUE)
par(mfrow=c(1,1))
WMT_close=as.xts(WMT$Close,as.Date(WMT$Date))
plot(WMT_close, main = "Original")
plot(diff(log(WMT_close)), col=4, main = "After Transformation")

#We notice non-stationary pattern, so need to take diff
plot(log(WMT_close))
plot(diff(log(WMT_close)))
library(astsa)
```

```
acf2(diff(diff(log(WMT_close)),5))
# seems like there is someseasonality

plot(diff(WMT_close,5))
acf2(diff(WMT_close,5))


plot(diff(diff(WMT_close),5))
acf2(diff(diff(WMT_close),5))

#ma model
sarima(log(WMT_close), p=0, d=1, q=1, P=0, D=1, Q=1, S = 0)
sarima(WMT_close[c(2707:3000),], p=0, d=1, q=0)


######### Walmart Stock (ARIMA and Garch model)
WMT=read.table("/Users/bingyuesu/Time series/FinalProject/archive/
WMT_2006-01-01_to_2018-01-1.csv",
sep=',',head=TRUE)
WMT_close=as.xts(WMT$Close,as.Date(WMT$Date))
time_pred=as.Date(WMT$Date[c(2895:3020)]) #half year:2895, two month: 2980
logWMT=log(WMT_close[c(1:2894)])
plot(logWMT)
plot(WMT_close)
dlogWMT=diff(logWMT)[-1]
time_dlogWMT <- index(dlogWMT)

plot(dlogWMT,col='blue')
adf.test(dlogWMT)
acf2(dlogWMT)
dlogWMT_fit=sarima(dlogWMT,p=5, d=0, q=0, no.constant = TRUE)
sarima(logWMT,p=5, d=1, q=0, no.constant = TRUE)

WMT_sarima_fit=sarima.for(logWMT,n.ahead=126,p=5, d=1, q=0)
#half year: 126, two month: 41
WMT_sariam_s_fit=sarima.for(logWMT,n.ahead=126,p=5, d=1, q=0,0,1,0,260)
pred_WMT_sarima_s_mean=WMT_sariam_s_fit$pred
pred_WMT_sarima_mean=WMT_sarima_fit$pred
pred_WMT_sarima_upper=WMT_sarima_fit$pred+WMT_sarima_fit$se
pred_WMT_sarima_lower=WMT_sarima_fit$pred-WMT_sarima_fit$se
plot(WMT_close[c(2700:3020)])
lines(as.xts(as.vector(exp(pred_WMT_sarima_mean)),time_pred),col=2,lwd=3)
#lines(as.xts(as.vector(exp(pred_WMT_sarima_upper)),time_pred),col=2,lwd=3)
#lines(as.xts(as.vector(exp(pred_WMT_sarima_lower)),time_pred),col=2,lwd=3)

doubleWMT=HoltWinters(WMT_close[c(1:2894)],gamma=FALSE)
```

```
pred_WMT_double=predict(doubleWMT,n.ahead=126)
lines(as.xts(as.vector(pred_WMT_double),time_pred),col=3,lwd=3)
lines(as.xts(as.vector(exp(pred_WMT_sarima_s_mean)),time_pred),col=4,lwd=3)


sqrt(sum((WMT_close[c(2895:3020)]-as.vector(exp(pred_WMT_sarima_mean)))^2)/126)
sqrt(sum((WMT_close[c(2895:3020)]-as.vector(pred_WMT_double))^2)/126)
sqrt(sum((WMT_close[c(2895:3020)]-as.vector(exp(pred_WMT_sarima_s_mean)))^2)/126)


#dlogWMT_fit=sarima(dlogWMT,p=2, d=0, q=0, no.constant = TRUE)
res=dlogWMT_fit$fit$residuals
acf2(res)
acf2(res^2)
Garch_WMT=garchFit(~arma(5,0)+garch(1,1),cond.dist = "std",data=dlogWMT)
summary(Garch_WMT)
plot(Garch_WMT)

plot(time_dlogWMT, dlogWMT, type = "l", main = "WMT-close",
     ylab='diff of log WMT close', xlab='time')
fitted_sd <- Garch_WMT@sigma.t
lines(time_dlogWMT, fitted_sd, col="red")
lines(time_dlogWMT, -fitted_sd, col="red")
lines(time_dlogWMT, 2*fitted_sd, col="blue")
lines(time_dlogWMT, -2*fitted_sd, col="blue")

predict(Garch_WMT, n.ahead = 126)
forecast(Garch_WMT, n.ahead=126)

###another garch model
spec <- ugarchspec(variance.model=list(model = "sGARCH",
garchOrder = c(1, 1)),
mean.model = list(armaOrder = c(5,0)),
distribution.model = "std")
# without fixed parameters here
fit <- ugarchfit(spec, data = dlogWMT)
pred_garch=ugarchforecast(fit,dlogWMT,n.ahead=126)
pred_mean=pred_garch@forecast$seriesFor
pred_sd=pred_garch@forecast$sigmaFor

diff_upper=pred_mean+pred_sd
diff_lower=pred_mean-pred_sd

pred_WMT=rep(0,126)
pred_upper=rep(0,126)
pred_lower=rep(0,126)
```

```
for(i in 1:126){
  if(i==1){
    pred_WMT[i]=logWMT[2894]+pred_mean[i]
    pred_upper[i]=logWMT[2894]+diff_upper[i]
    pred_lower[i]=logWMT[2894]+diff_lower[i]
  }else{
    pred_WMT[i]=pred_WMT[i-1]+pred_mean[i]
    pred_upper[i]=pred_upper[i-1]+diff_upper[i]
    pred_lower[i]=pred_lower[i-1]+diff_lower[i]
  }
}
pred_WMT=exp(pred_WMT)
pred_upper=exp(pred_upper)
pred_lower=exp(pred_lower)

pred_upper=as.xts(pred_upper,time_pred)
pred_lower=as.xts(pred_lower,time_pred)
pred_WMT=as.xts(pred_WMT,time_pred)
plot(WMT_close[c(2700:3020)])
lines(pred_WMT,col=6,lwd=3)
addLegend(legend.loc = "topleft",
          legend.names = c("ARIMA",
          "Double Exponential Smoothing", "SARIMA","GARCH"),
          col = NULL,
          col = c(1,2,3,6))
legend("topleft", legend=c("ARIMA", "Double Exponential Smoothing", "SARIMA","GARCH"),
       col=c(1,2,3,6),lty=c(1,2,3,6), cex=0.8)

lines(pred_upper,col=3,lwd=3)
lines(pred_lower,col=3,lwd=3)

sqrt(sum((WMT_close[c(2895:3020)]-as.vector(pred_WMT))^2)/126)
exp(pred_WMT_sarima_mean)

##############
GAM model via Prophet package in Python
##############
import numpy as np
import pandas as pd
ama_data = pd.read_csv('../input/final-data/
AMZN_2006-01-01_to_2018-01-01.csv')
wmt_data = pd.read_csv('../input/stock-time-series-20050101-to-20171231/
WMT_2006-01-01_to_2018-01-01.csv')
test_data_size = 126
#ama_data = pd.read_csv('../input/final-data/AMZN_2006-01-01_to_2018-01-01.csv')
#all_data = ama_data[['Date','Close']].copy()
```

```
wmt_data = pd.read_csv('../input/stock-time-series-20050101-to-20171231/
WMT_2006-01-01_to_2018-01-01.csv')
all_data = wmt_data[['Date','Close']].copy()
all_data.columns = ['ds','y']
all_data['ds']= pd.to_datetime(all_data['ds'])
train_data = all_data[:-test_data_size]
test_data = all_data[-test_data_size:]
from fbprophet import Prophet
m = Prophet(weekly_seasonality=False)
m.fit(train_data)
future = m.make_future_dataframe(periods=test_data_size)
forecast = m.predict(future)
# Visualize the data
plt.figure(figsize=(16,6))
plt.title('GAM')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(train_data['y'])
plt.plot(test_data['y'])
plt.plot(forecast['yhat'])
plt.legend(['Train', 'Test', 'Predictions'], loc='lower right')
plt.show()

############
LSTM
############
#wmt_data = pd.read_csv('../input/wmtcsv/WMT_2006-01-01_to_2018-01-01.csv')
#all_data = wmt_data['Close'].values.astype(float).reshape(-1,1)
ama_data = pd.read_csv('../input/final-data/AMZN_2006-01-01_to_2018-01-01.csv')
all_data = ama_data['Close'].values.astype(float).reshape(-1,1)
# Scale the data
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(all_data)
# Create the training data set
# Create the scaled training data set
test_data_size = 126
training_data_len = len(scaled_data)-test_data_size
train_data = scaled_data[0:int(training_data_len), :]
# Split the data into x_train and y_train data sets
x_train = []
y_train = []

for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
```

```
    y_train.append(train_data[i, 0])
    if i<= 61:
        print(x_train)
        print(y_train)
        print()

# Convert the x_train and y_train to numpy arrays
x_train, y_train = np.array(x_train), np.array(y_train)

# Reshape the data
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
# x_train.shape
from keras.models import Sequential
from keras.layers import Dense, LSTM

# Build the LSTM model
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(x_train, y_train, batch_size=1, epochs=2)
#one-step
test_data = scaled_data[training_data_len - 60: , :]
# Create the data sets x_test and y_test
x_test = []
y_test = all_data[training_data_len:]
for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])

# Convert the data to a numpy array
x_test = np.array(x_test)

# Reshape the data
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1 ))

# Get the models predicted price values
predictions = model.predict(x_test)
predictions_one_step = scaler.inverse_transform(predictions)

# Get the root mean squared error (RMSE)
```

```
rmse = np.sqrt(np.mean(((predictions_one_step - y_test) ** 2)))
#multi-steps
x_test = scaled_data[training_data_len - 60:training_data_len, :]
predictions = []
for i in range(test_data_size):
    pred = model.predict(x_test[-60:].reshape(-1,60,1))
    x_test = np.append(x_test,pred)
    pred_unscaled = scaler.inverse_transform(pred)
    predictions.append(pred_unscaled)
predictions = np.array(predictions)
rmse = np.sqrt(np.mean(((predictions - y_test) ** 2)))
# Plot the data
train =  all_data[:training_data_len]
# Visualize the data
plt.figure(figsize=(16,6))
plt.title('LSTM')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(range(len(train)),train)
plt.plot(range(len(train),len(train)+len(y_test)),y_test)
plt.plot(range(len(train),len(train)+len(y_test)),predictions_one_step)
plt.plot(range(len(train),len(train)+len(y_test)),predictions.reshape(len(predictions),))
plt.legend(['Train', 'Test', 'One-Step','Multi-Steps'], loc='lower right')
plt.show()
```

References

[Marjanovic(2018)] B. Marjanovic. Huge stock market dataset. *Kaggle*, 2018.

[Robert and Stoffer(2017)] S. Robert and Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer., 2017.

[Sak et al.(2014)Sak, Senior, and Beaufays] H. Sak, A. Senior, and F. Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[Taylor and Letham(2018)] S. J. Taylor and B. Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.