Project 2

ARIMA Model for the VXX with RSI Regressor

## Abstract

In our previous Project 1, we identified that the best ARIMA model for our 2/24/17 to 2/13/18 time period was an ARIMA(2,1,2) with drift. We had also noticed that our closing price data was right-tailed and log transformed it successfully to be more normally distributed. For this project we will continue with our original data set and append on days from 2/13/18 to 4/2/18. We will then include a RSI indicator values to be used as a regressor with our model.

## Introduction

The regressor that we have chosen, the Relative Strength Index, is a measure that is calculated using a simple moving average of the price up-days divided by a simple moving average of the price down-days. Up-days is defined as the close now minus the close previous and the Down-days is defined as the close previous minus the close now [3]. Overall the goal of the Relative Strength Index is to measure the price momentum, the rate of change of price. We will investigate whether including RSI as a regressor to our ARIMA model will or will not improve its performance.

## Methods

Just as our data originated from the collection of VXX trades on the NinjaTrader platform, the RSI data was also created by the platform [4]. We decided on using the different RSI periods of 20, 40, 80, and 150 to see if there was any difference between them as regressors. Once the data was imported we created a violin plot, Figure 1, for visual inspection.

From Figure 1 we can see that as the period of the RSI get longer, from 20 to 150, the distributions of the variables becomes smaller. Also note that each one is right-tailed which makes sense for our data since during this time period the price continually dropped until a low point was reached and then it spent more time in the 30 to 50 price range.

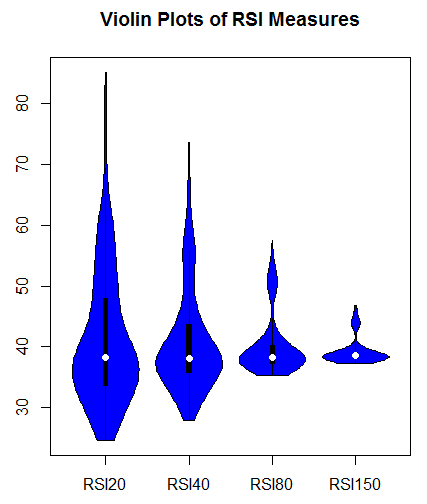


Figure 1. Violin Plot of RSI Variables.

In order to include the RSI variables as regressors in our ARIMA models we use the following code for each RSI variable. We then extract the actual model, model coefficients, forecasting accuracies, and plots of the forecast for each RSI variable.

arima20 <- auto.arima(data.train.vxxClose, trace=TRUE, seasonal = TRUE, stepwise = FALSE, approximation = FALSE, xreg=data.train.vxxRSI20)

arima20.forecast <- forecast(arima20, h=77, xreg = data.test.vxxRSI20)

plotarimapred(data.test.vxxClose, arima20, xlim=c(0,300), range.percent = 0.05, xreg = data.test.vxxRSI20)

arima20

accuracy(arima20.forecast, data.test.vxxClose)

BestModel1 <- auto.arima(data.train.vxxClose, trace=TRUE, test="kpss", ic="aic", xreg = data.test.vxxRSI20)

BestModel2 <- auto.arima(data.train.vxxClose, trace=TRUE, test="kpss", ic="bic", xreg = data.test.vxxRSI20)

BestModel3 <- auto.arima(data.train.vxxClose, trace=TRUE, test="kpss", ic="aicc", xreg = data.test.vxxRSI20)

BestModel4 <- auto.arima(data.train.vxxClose, trace=TRUE, test="adf", ic="aic", xreg = data.test.vxxRSI20)

BestModel5 <- auto.arima(data.train.vxxClose, trace=TRUE, test="adf", ic="bic", xreg = data.test.vxxRSI20)

BestModel6 <- auto.arima(data.train.vxxClose, trace=TRUE, test="adf", ic="aicc", xreg = data.test.vxxRSI20)

BestModel7 <- auto.arima(data.train.vxxClose, trace=TRUE, test="pp", ic="aic", xreg = data.test.vxxRSI20)

BestModel8 <- auto.arima(data.train.vxxClose, trace=TRUE, test="pp", ic="bic", xreg = data.test.vxxRSI20)

BestModel9 <- auto.arima(data.train.vxxClose, trace=TRUE, test="pp", ic="aicc", xreg = data.test.vxxRSI20)

## Results

The results that we achieved are as follows. For the RSI20 variable as the regressor the best model is an ARIMA(0,1,0) with errors, Figure 2. The forecast of the predictions can be seen in Figure 3.

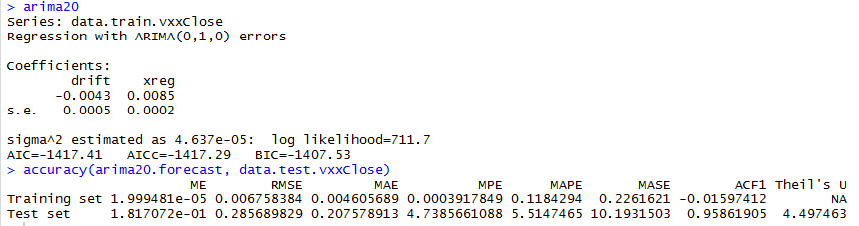


Figure 2. Best Model for RSI20 Variable.

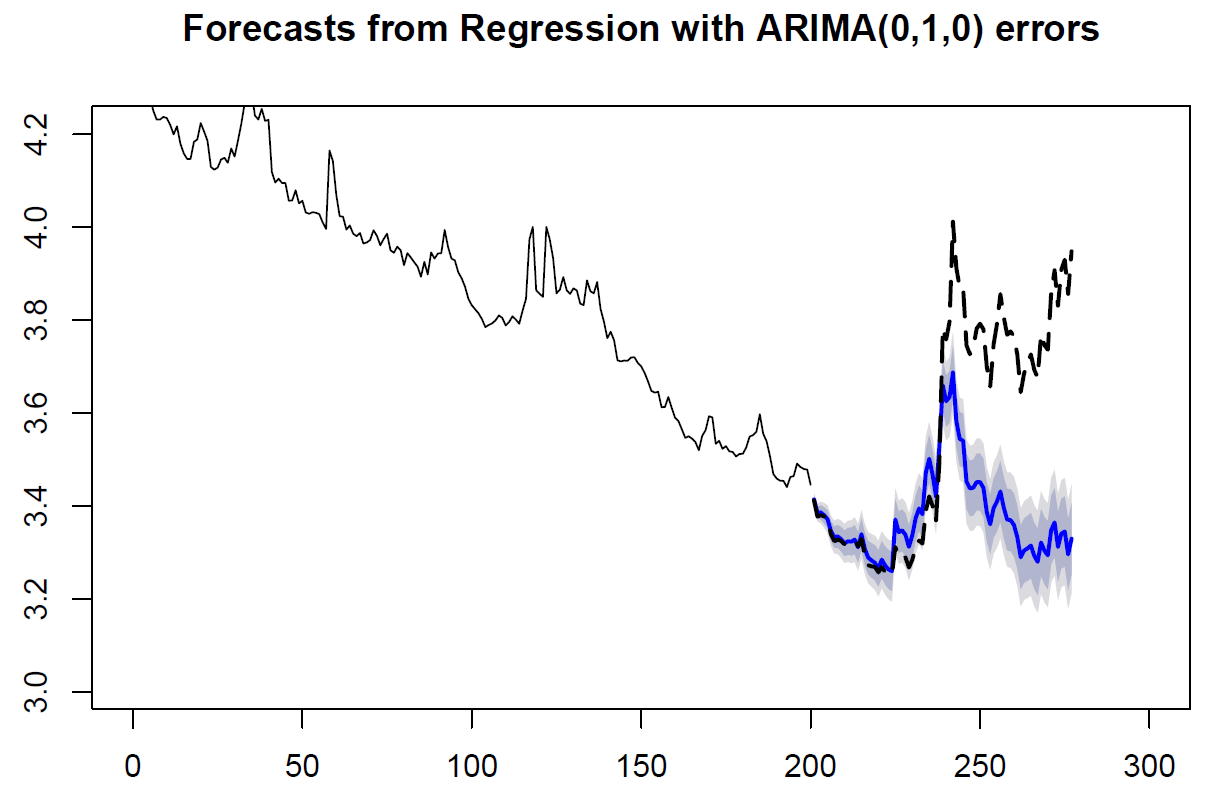


Figure 3. RSI20 Forecast.

For the RSI40 variable as the regressor the best model is an ARIMA(0,1,0) with errors, Figure 4. The forecast of the predictions can be seen in Figure 5.

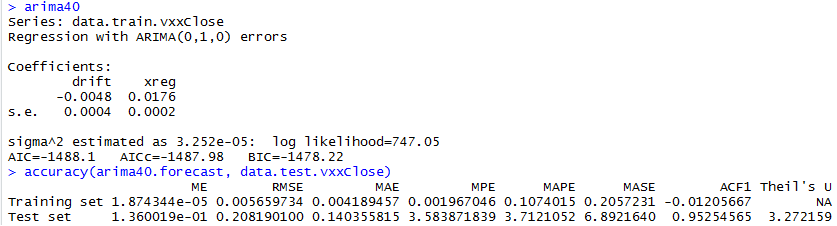


Figure 4. Best Model for RSI40 Variable.

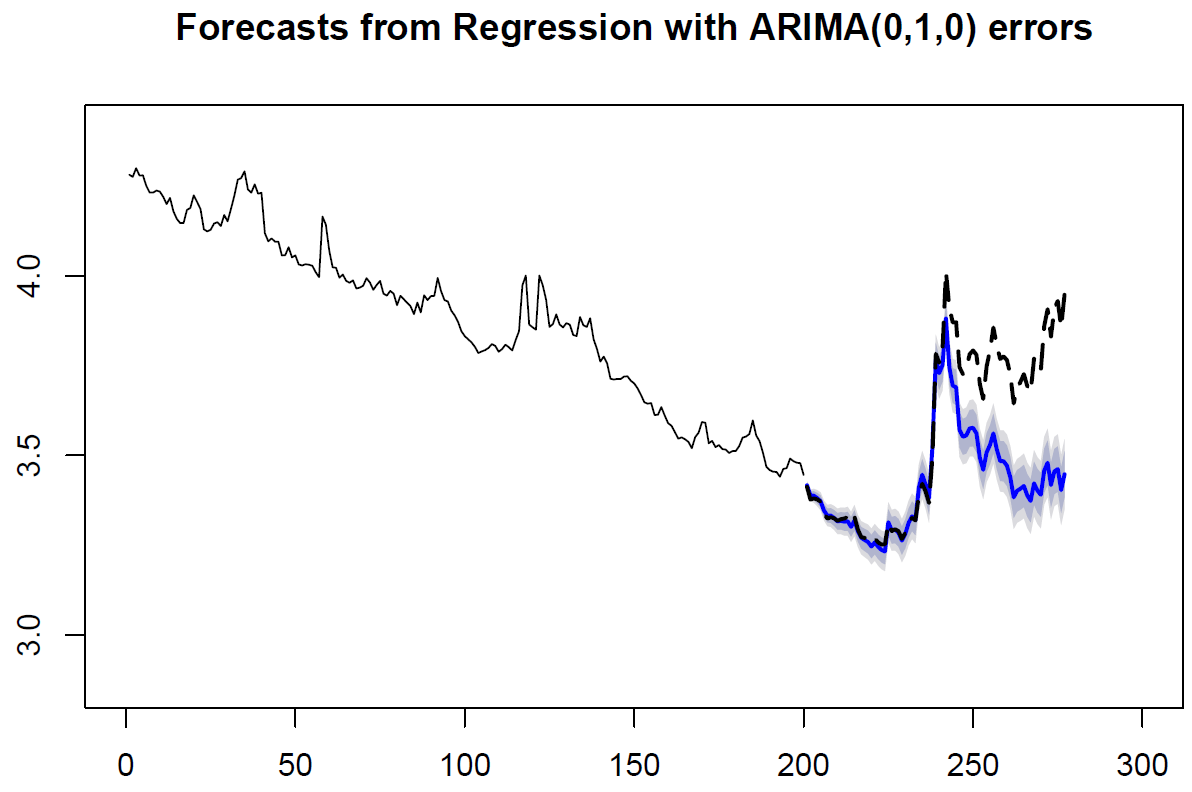


Figure 5. RSI40 Forecast.

For the RSI80 variable as the regressor the best model is an ARIMA(2,1,3) with errors, Figure 6. This is a very similar ARIMA model compared to the best model for Project 1. The forecast of the predictions can be seen in Figure 7.

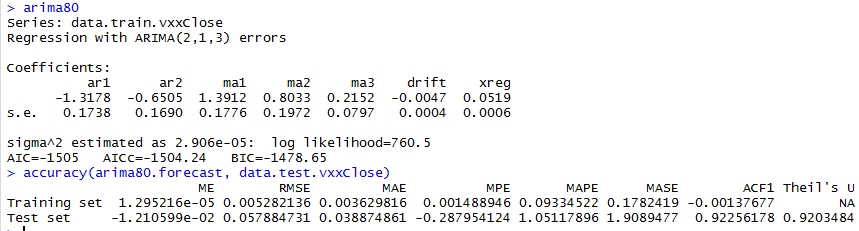


Figure 6. Best Model for RSI80 Variable.

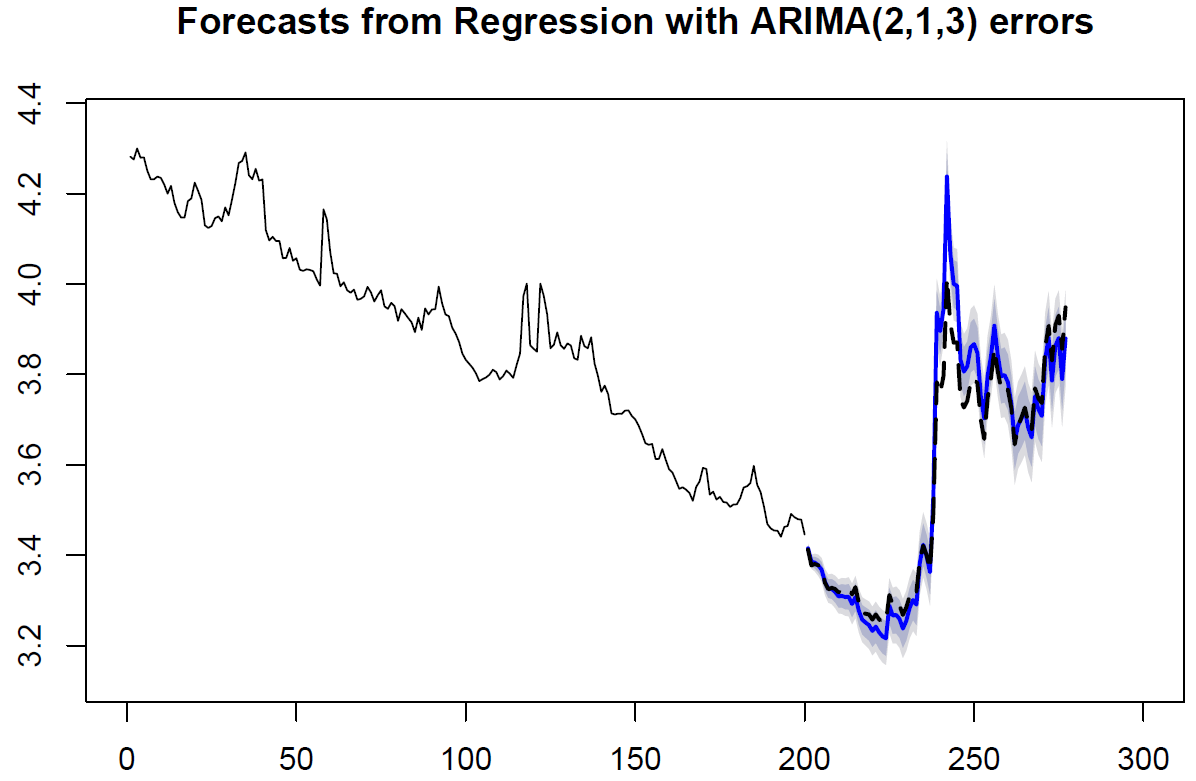


Figure 7. RSI80 Forecast.

For the RSI150 variable as the regressor the best model is an ARIMA(3,1,1) with errors, Figure 8. The forecast of the predictions can be seen in Figure 9.

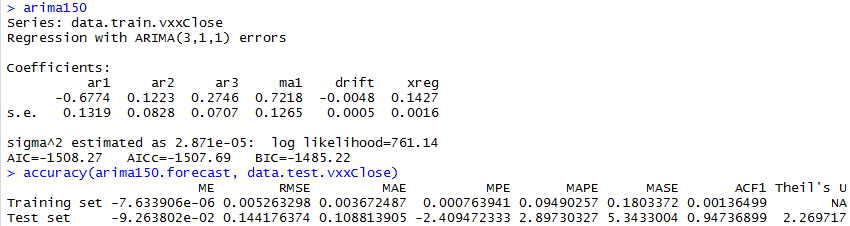


Figure 8. Best Model for RSI150 Variable.

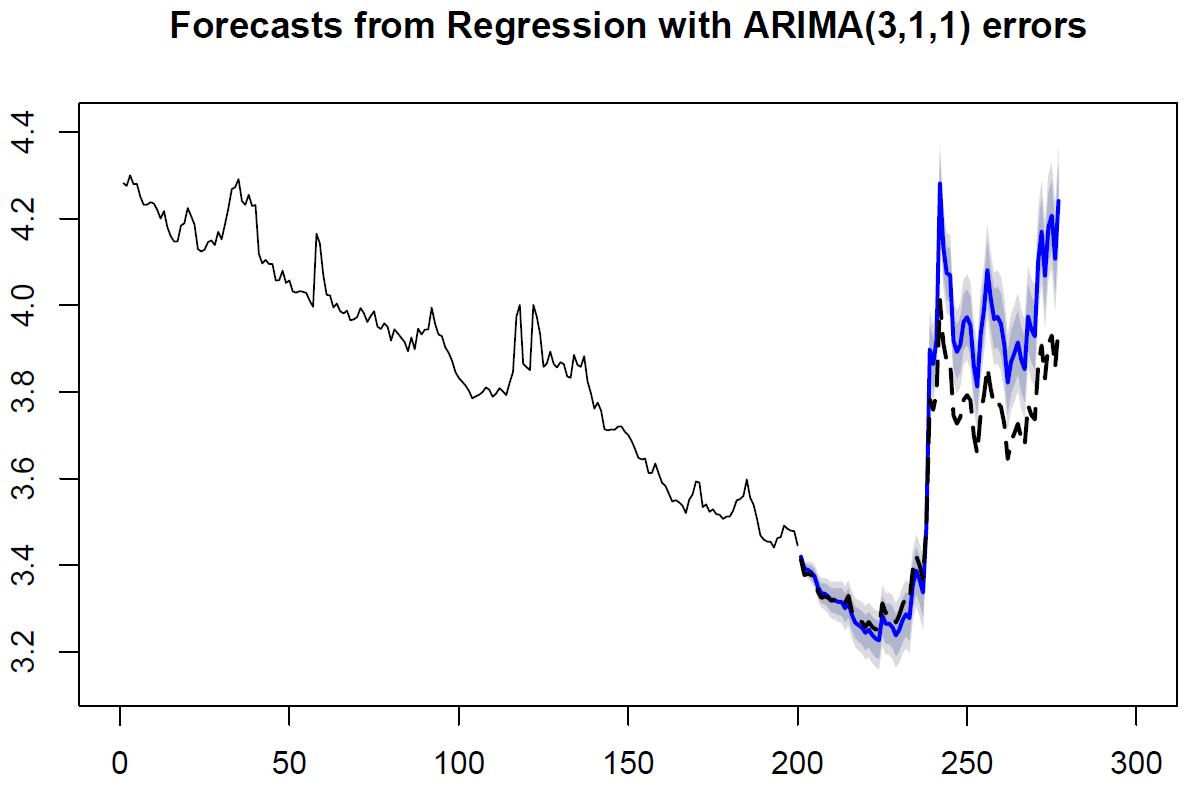


Figure 9. RSI150 Forecast.

We can see that the best model for our VXX data with an RSI regressor is using an RSI with an 80 period and an ARIMA(2,1,3) with errors. Theil’s U, 0.92, is less than 1.00 therefore we interpret our results to have been better than random guessing. Another way to confirm this is from Figure 6 which shows our model coefficients do not add up to 1 which means we are doing better than random guessing and being lucky. The following Figure 10 is a plot of the diagnostics for our best model.

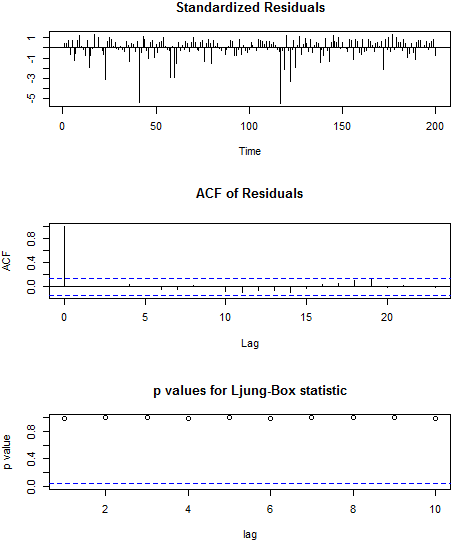


Figure 10. Standardized Residuals, ACF of Residuals, and Ljung-Box statistic.

## From Figure 10, we can see that the Standardized show that we have several outliers, roughly 4 of them. These outliers are not an issue and come from the major spikes in price movement. The residuals show no sign of non-linearity or heteroscedasticity. The ACF of residuals shows us a value of 1 for the lag 0 which means that our model needs an AR for the first term, which it does. The p-values for the Ljung-Box statistic are all > 0.05 which means that for the first 10 lags we reject the alternative hypothesis that the residuals are dependent.

## At this point we see that we have a good model that is quite accurate in forecasting and predicting future prices. However, there is one issue that throws a major wrench into using this model for anything other than historical look back. When forecasting using the “forecast” package in R and using a regressor, the algorithm requires future values of the regressor. However, in practice you do not always have these values and in our case there is no way to know the exact value of tomorrow’s RSI80 indicator.

In order to try to mitigate this issue we replace the forecasting periods RSI80 values with the mean of the values for our time period, excluding our forecast period. This will show us if the model is actually better in real-life. From Figure 11 we see the forecast and from Figure 12 the accuracy measures.

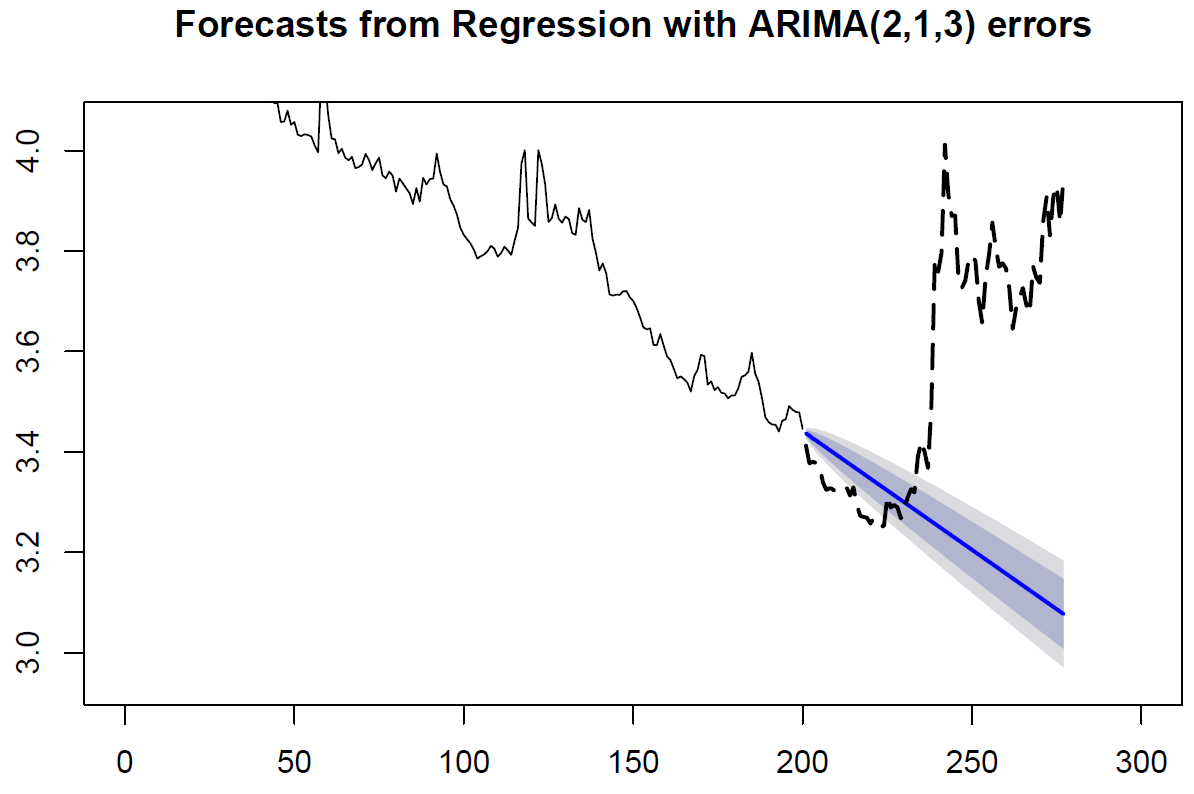


Figure 11. Corrected RSI Prediction Values.

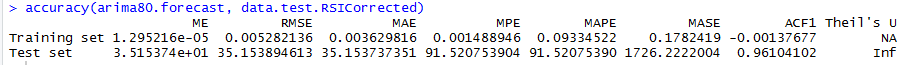


Figure 12. Corrected Accuracy Measures.

The forecast is now only reflecting the downward trend, this is due to us taking the average of RSI80 for our training interval, which itself is a downward trend. Theil’s U is now Infinity which means our model is infinitely worse than random guessing. Also, the MAPE for the training is 0.09 while it is 91.52 for the testing.

We can see from these results how the best model we created was so accurate, it was including future prices. Through the nature of how the RSI indicator is built future values, or good enough regurgitations of them are included in the future prediction. Basically, we were predicting the values we already knew to be true. This is an excellent example of how you can build a model, or a trading system, that inherently incorporates future values, but in practice will utterly fail since you do not have those future values after the back testing of the model.

## 

## Conclusion

In conclusion, we find that the best model for our daily data set that we generated from 36 million rows of tick data was an ARIMA(2, 1, 3) model. We saw that for when building a model for forecasting we need to ensure that the model makes sense from a practical real-world perspective. Hindsight is always 20/20 and we need to guard against using future prices, which we will not know at the moment we run the model, in the creation of our models as that leads to an inherently inaccurate real-world model. If we wanted to use the model we generated as a historical lesson on how an RSI regressor helps fit a model to past values only then that would be a proper application of the model.

## Appendix

### 

### Reference

1. Daróczi Gergely (2013). Introduction to R for Quantitative Finance. Packt Publishing.
2. Professor Chad Maybin (2018). Southern Methodist University.
3. Relative Strength Index: <https://en.wikipedia.org/wiki/Relative_strength_index>
4. Relative Strength Index (rsi): Common Overbought and Oversold Thresholds: https://ninjatrader.com/blog/relative-strength-index-rsi-common-overbought-oversold-thresholds/

### 

### Complete Code Base

##########################################################

## Clean Up, only use on first time code run

##########################################################

rm(list = ls())

gc()

par(mfrow=c(1,1))

##########################################################

## Load Packages & Set Working Directory

##########################################################

setwd("C:/Users/jbeer/Documents/R/VXX")

library(data.table)

require(data.table)

library(zoo)

library(dplyr)

library(tidyr)

library(xts)

library(forecast)

library(tseries)

library(TSPred)

library(changepoint)

library(vioplot)

library(forecastHybrid)

##########################################################

## Read in Data from 2/24/17 to 4/2/18

##########################################################

# Read in our tick data

vxx <- fread("VXX RSI.csv", sep=",")

# Check the data

head(vxx)

##########################################################

## Munge Data

##########################################################

# Format Date Column

vxx$Date <- as.POSIXct(vxx$Date, tz = "UTC","%m/%d/%Y")

# Did we do it right?

sum(is.na(vxx$Date))

# Ensure there are no Null values

colSums(is.na(vxx))

# Final check of DF

str(vxx)

summary(vxx)

# Let's plot our data, we will see an issue with price near the midpoint of our data set. It jumps from around 14 to 50.

plot(vxx$Close)

plot.ts(vxx$Close)

# Check the Violin Plots again to confirm a more normal distribution of our data.

x1 <- vxx$Close

vioplot(x1, names=c("Close"), col="blue")

title("Violin Plot of Close")

# Visual check of the Violin Plots for each variable

x2 <- vxx$RSI20

x3 <- vxx$RSI40

x4 <- vxx$RSI80

x5 <- vxx$RSI150

vioplot(x2, x3, x4, x5, names=c("RSI20", "RSI40", "RSI80", "RSI150"), col="blue")

title("Violin Plots of RSI Measures")

# Log Transform our close also since the QQ plots of the ARIMA model residuals are non-normal (heavy tailed)

vxx$Close <- log(vxx$Close)

# Check our data

head(vxx)

# Check the Violin Plots again to confirm a more normal distribution of our data.

x1 <- vxx$Close

vioplot(x1, names=c("Close"), col="blue")

title("Violin Plot of Close")

# Clean up

gc()

##########################################################

# Splitting our Training and Testing data

##########################################################

# Isolate just the closing price

vxxClose <- vxx[,2]

# Isolate the different RSI measures, this is for later use as an xreg in auto.arima

vxxRSI20 <- vxx[,3]

vxxRSI40 <- vxx[,4]

vxxRSI80 <- vxx[,5]

vxxRSI150 <- vxx[,6]

# How many days of data do we have? Should be 277

count(vxxClose)

# Create our time series objects

ts.vxxClose <- ts(vxxClose, frequency=1, start=1)

ts.vxxRSI20 <- ts(vxxRSI20, frequency=1, start=1)

ts.vxxRSI40 <- ts(vxxRSI40, frequency=1, start=1)

ts.vxxRSI80 <- ts(vxxRSI80, frequency=1, start=1)

ts.vxxRSI150 <- ts(vxxRSI150, frequency=1, start=1)

#Split data into training and testing data set according to using the first 200 obs as our training and the last 77 for prediction

data.train.vxxClose <- window(ts.vxxClose, start=1, end=200)

data.test.vxxClose <- window(ts.vxxClose, start=201, end=277)

data.train.vxxRSI20 <- window(ts.vxxRSI20, start=1, end=200)

data.test.vxxRSI20 <- window(ts.vxxRSI20, start=201, end=277)

data.train.vxxRSI40 <- window(ts.vxxRSI40, start=1, end=200)

data.test.vxxRSI40 <- window(ts.vxxRSI40, start=201, end=277)

data.train.vxxRSI80 <- window(ts.vxxRSI80, start=1, end=200)

data.test.vxxRSI80 <- window(ts.vxxRSI80, start=201, end=277)

data.train.vxxRSI150 <- window(ts.vxxRSI150, start=1, end=200)

data.test.vxxRSI150 <- window(ts.vxxRSI150, start=201, end=277)

# Make sure we have 277 observations still

length(data.test.vxxClose)+length(data.train.vxxClose)

gc()

##########################################################

# ARIMA Model Creation Using XREG = RSI20

##########################################################

# Per the documentation of the "forecast" package, if we are analyzing only one time series and can afford the

# computations, we should set stepwise = FALSE and approximation = FALSE, which we will do here.

# Best model is an ARIMA(2,1,3) with errors

arima20 <- auto.arima(data.train.vxxClose, trace=TRUE, seasonal = TRUE, stepwise = FALSE, approximation = FALSE, xreg=data.train.vxxRSI20)

# Create our forecast

arima20.forecast <- forecast(arima20, h=77, xreg = data.test.vxxRSI20)

# Plot our predictions

plotarimapred(data.test.vxxClose, arima20, xlim=c(0,300), range.percent = 0.05, xreg = data.test.vxxRSI20)

# View the best model, we see that we get

arima20

accuracy(arima20.forecast, data.test.vxxClose)

##########################################################

# ARIMA Model Creation Using XREG = RSI40

##########################################################

# Per the documentation of the "forecast" package, if we are analyzing only one time series and can afford the

# computations, we should set stepwise = FALSE and approximation = FALSE, which we will do here.

# Best model is an ARIMA(2,1,3) with errors

arima40 <- auto.arima(data.train.vxxClose, trace=TRUE, seasonal = TRUE, stepwise = FALSE, approximation = FALSE, xreg=data.train.vxxRSI40)

# Create our forecast

arima40.forecast <- forecast(arima40, h=77, xreg = data.test.vxxRSI40)

# Plot our predictions

plotarimapred(data.test.vxxClose, arima40, xlim=c(0,300), range.percent = 0.1, xreg = data.test.vxxRSI40)

# View the best model, we see that we get

arima40

accuracy(arima40.forecast, data.test.vxxClose)

## Another way to step through all the models so we can print out the forecasts

#BestModel1 <- auto.arima(data.train.vxxClose, trace=TRUE, test="kpss", ic="aic", xreg=data.train.vxxRSI40)

#BestModel2 <- auto.arima(data.train.vxxClose, trace=TRUE, test="kpss", ic="bic", xreg=data.train.vxxRSI40)

#BestModel3 <- auto.arima(data.train.vxxClose, trace=TRUE, test="kpss", ic="aicc", xreg=data.train.vxxRSI40)

#BestModel4 <- auto.arima(data.train.vxxClose, trace=TRUE, test="adf", ic="aic", xreg=data.train.vxxRSI40)

#BestModel5 <- auto.arima(data.train.vxxClose, trace=TRUE, test="adf", ic="bic", xreg=data.train.vxxRSI40)

#BestModel6 <- auto.arima(data.train.vxxClose, trace=TRUE, test="adf", ic="aicc", xreg=data.train.vxxRSI40)

#BestModel7 <- auto.arima(data.train.vxxClose, trace=TRUE, test="pp", ic="aic", xreg=data.train.vxxRSI40)

#BestModel8 <- auto.arima(data.train.vxxClose, trace=TRUE, test="pp", ic="bic", xreg=data.train.vxxRSI40)

#BestModel9 <- auto.arima(data.train.vxxClose, trace=TRUE, test="pp", ic="aicc", xreg=data.train.vxxRSI40)

#modList <- list(BestModel1, BestModel2, BestModel3, BestModel4, BestModel5, BestModel6, BestModel7, BestModel8, BestModel9)

## Export a forecast for each of these best models.

#for (Amodel in modList){

# i=0

# arima40.forecast <- forecast(Amodel, h=77, xreg = data.test.vxxRSI40) #forecast 24 periods ahead

# arima40.forecast

# plot(arima40.forecast, xlab="Date", ylab="VXX Price")

# library(TSPred)

# plotarimapred(data.test.vxxClose, Amodel, xlim=c(0,300), range.percent = 0.1, xreg = data.test.vxxRSI40)

# accuracy(arima40.forecast, data.test.vxxClose)

# #plot as confirmation

# pdf( paste0("Plot - ", Amodel,format(Sys.time(), "%a %b %d %H %M %S %Y"), " .pdf"),width=7,height=5)

# plotarimapred(data.test.vxxClose, Amodel, xlim=c(0,300), range.percent = 0.1, xreg = data.test.vxxRSI40)

# accuracy(arima40.forecast, data.test.vxxClose)

# dev.off()

#}

##########################################################

# ARIMA Model Creation Using XREG = RSI80

##########################################################

# Per the documentation of the "forecast" package, if we are analyzing only one time series and can afford the

# computations, we should set stepwise = FALSE and approximation = FALSE, which we will do here.

# Best model is an ARIMA(2,1,3) with errors

arima80 <- auto.arima(data.train.vxxClose, trace=TRUE, seasonal = TRUE, stepwise = FALSE, approximation = FALSE, xreg=data.train.vxxRSI80)

# Create our forecast

arima80.forecast <- forecast(arima80, h=77, xreg = data.test.vxxRSI80)

# Plot our predictions

plotarimapred(data.test.vxxClose, arima80, xlim=c(0,300), range.percent = 0.01, xreg = data.test.vxxRSI80)

# View the best model, we see that we get

arima80

accuracy(arima80.forecast, data.test.vxxClose)

# Correcting for knowing future values of prices via the RSI indicator

# Generate the object

Corrected <- c(0, 1.0, 2.0)

# Find the mean of the first 200 observations

RSImean <- mean(vxx$RSI80[1:200])

# Assign out the mean to each value of the Corrected object

for(i in 1:77){

Corrected[i] <- RSImean

return;

}

# Create our time series and data.test window

ts.RSICorrected <- ts(Corrected, frequency=1, start=201)

data.test.RSICorrected <- window(ts.RSICorrected, start=201, end=277)

# Run best model with this data.

arima80.forecast <- forecast(arima80, h=77, xreg = data.test.RSICorrected)

plotarimapred(data.test.vxxClose, arima80, xlim=c(0,300), range.percent = 0.01, xreg = data.test.RSICorrected)

# View the best model, we see that we get

arima80

accuracy(arima80.forecast, data.test.RSICorrected)

##########################################################

# ARIMA Model Creation Using XREG = RSI150

##########################################################

# Per the documentation of the "forecast" package, if we are analyzing only one time series and can afford the

# computations, we should set stepwise = FALSE and approximation = FALSE, which we will do here.

# Best model is an ARIMA(2,1,3) with errors

arima150 <- auto.arima(data.train.vxxClose, trace=TRUE, seasonal = TRUE, stepwise = FALSE, approximation = FALSE, xreg=data.train.vxxRSI150)

# Create our forecast

arima150.forecast <- forecast(arima150, h=77, xreg = data.test.vxxRSI150)

# Plot our predictions

plotarimapred(data.test.vxxClose, arima150, xlim=c(0,300), range.percent = 0.01, xreg = data.test.vxxRSI150)

# View the best model, we see that we get

arima150

accuracy(arima150.forecast, data.test.vxxClose)

##########################################################

# Model Diagnotics: vxxClose

##########################################################

# Our model looks good since the standardized residuals don't show volatility clusters, no significant autocorrelations between the residuals

# according to the ACF plot, and the Ljung-Box test for autocorrelation shows high p-values, so the null hypothesis of independent residuals cannot be rejected.

# To assess how well the model represents the data in the sample, we can plot the raw monthly returns (the thin black solid line) versus the fitted values

# (the thick red dotted line).

# ARIMA1 plot of time series and fitted model values

plot(arima80$x, lty = 1, main = "VXX raw data vs. fitted values", ylab = "VXX Price", xlab = "Date")

lines(fitted(arima80), lty = 2,lwd = 2, col = "red")

# Confidence Interval

confint(arima80)

# Summary

summary(arima80)

# Residuals, ACF, Ljung-Box

tsdiag(arima80)

# if p-value >0.05 then the residuals have a normal distibution (not reject null Hypothesis of normality)

# We see that the p-value < 2.2e-16 therefore we reject the null hypo that we have normally distributed residuals.

jarque.bera.test(arima80$residuals)

# Plot our residuals so we can see the distribution.

qqnorm(arima80$residuals)

qqline(arima80$residuals)

# Plot the distribution of our residuals

plot(density(arima80$residuals))

# Calculate Accuracy Measures

accuracy(arima80)

# Residual Diagnostics

plot.ts(arima80$residuals)

# From the Box-Ljung test we see that the p-value = 0.3046 therefore fail to reject null that the data are independent (i.e. no autocorrelation exists)

Box.test(arima80$residuals, lag=20, type="Ljung-Box")

# We see that the lag 6 barely crosses the boundary but no other lags do therefore no autocorrelation.

acf(arima80$residuals, lag.max = 24, main="ACF of the Model")

# Test for GARCH effect, p-value = 0.1163 therefore no ARCH effect present and GARCH model shouldn't be considered.

Box.test(arima80$residuals^2, lag = 20, type = "Ljung-Box")