Forecasting S&P 500 Index:
ARIMA + GARCH, Neural Networks, and Support Vector Machine Models

Natalia Gust-Bardon

September, 2019

Introduction

The aim of this project is to present models that can be used for forecasting on time series problems. We train our models on S&P 500 Index (Yahoo Finance) from 1995-01-01 to 2018-08-01. The test set comprises 12 months (2018-09-01 to 2019-08-01). We present three methods: ARiMA + GARCH, Neural Networks, and Support Vector Machine.

Figure 1

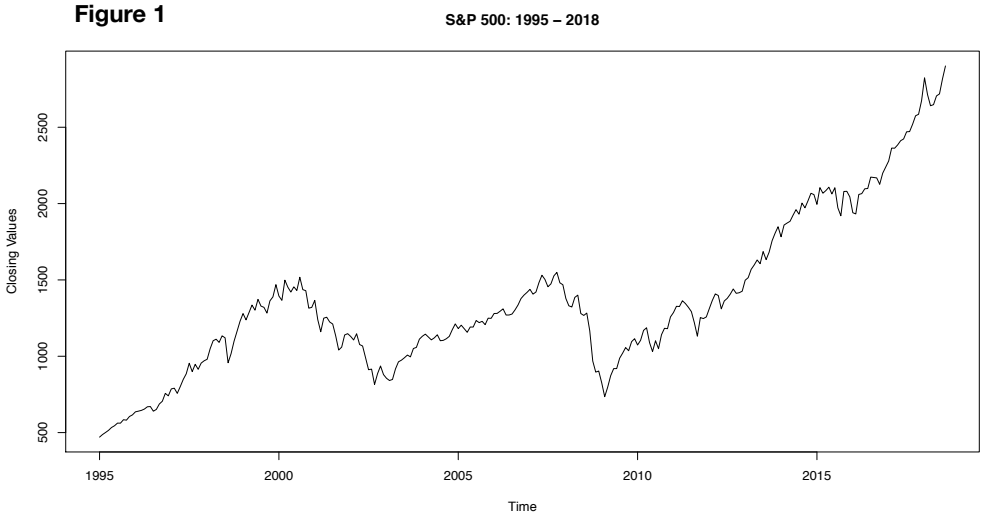
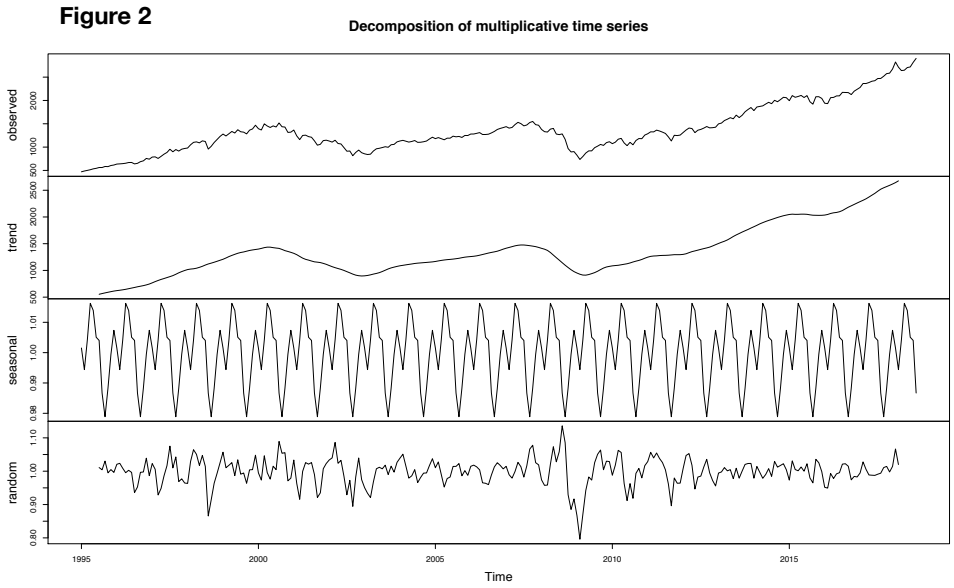


Figure 1 presents our training set.

1. ARIMA + GARCH Model

Figure 2

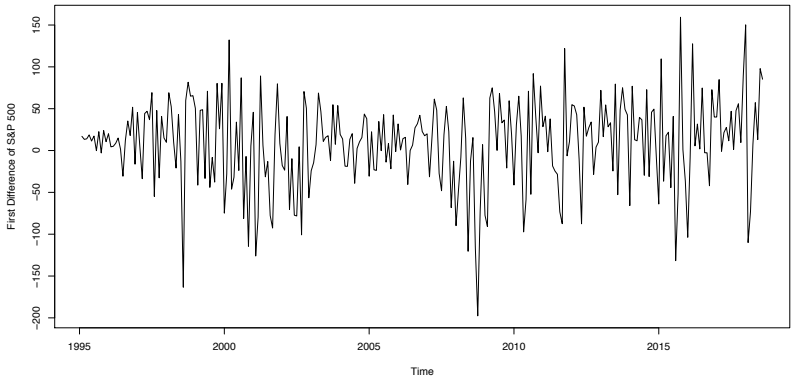


Based on *Figure 2*, we conclude the following:

- The trend is not linear. It has three visible picks (2000, 2008, and 2015)
- There is no significant seasonal effect
- The random component seems to have a constant mean
- We can see high volatility

Conclusion: because of high volatility, an ARIMA model would have failed to capture this phenomenon, as it assumes constant variance of the errors. To get rid of the trend, we take the first differences of the S&P500 Index.

Figure 3 Time Series Plot of the First Differences of S&P 500: 1995 – 2018



The general upward trend has now disappeared, no seasonality is observed. The mean seems to be stable around 0, the changes in the variance are worrisome (*Figure 3*).

Figure 4 ACF plot of First Difference of S&P 500

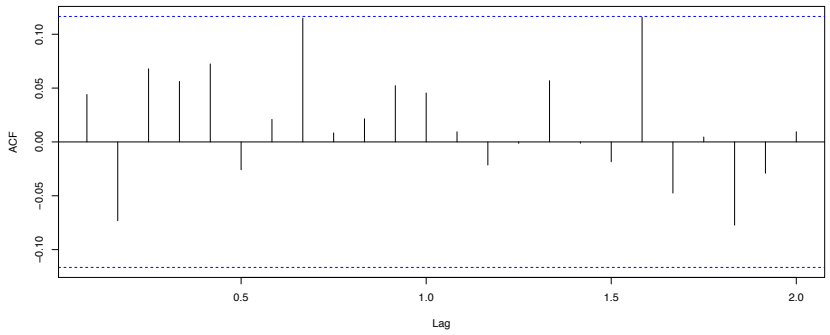
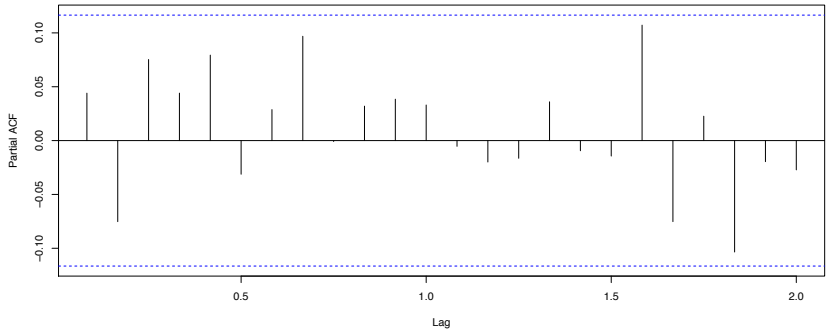


Figure 5 PACF plot of First Difference of S&P 500



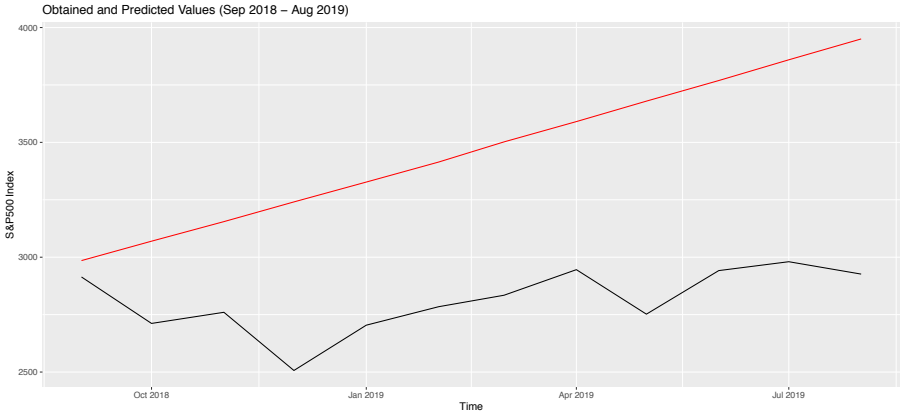
Based on ACF and PACF plots (*Figure 4*, *Figure 5*) we can conclude that the S&P500 Index follows a random walk process (ARIMA(0,1,0)). The *auto.arima()* function in R proves the same. Nonetheless, we violate the constant variance assumption.

We fit the ARIMA + GARCH model using g*archFit()* function. *Table 1* shows the summary of the model.

Table 1

```
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~arma(0, 0) + garch(1, 1), data = d1, trace = FALSE)

Mean and Variance Equation:
 data ~ arma(0, 0) + garch(1, 1)
<environment: 0x7fa5038078f8>
 [data = d1]

Conditional Distribution:
 norm

Coefficient(s):
       mu      omega     alpha1      beta1
 10.16697  111.71365    0.25128    0.74667

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu      10.16697     2.49794    4.070  4.7e-05 ***
omega  111.71365    77.09033    1.449 0.147302
alpha1   0.25128     0.07274    3.454 0.000551 ***
beta1    0.74667     0.06448   11.580  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 6



The predicted values skyrocket away from the observed values (*Figure 6*).

ARIMA + GARCH model does not look like a good fit for our data.

2. Neural Network

In this section, we use feed-forward neural network with a single hidden layer and lagged inputs for forecasting S&P 500 Index using *nnetar()* function in R.

Figure 7

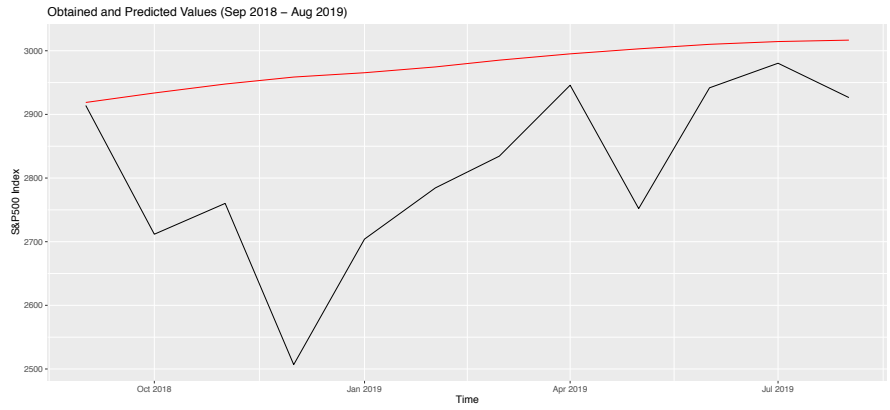


Figure 8

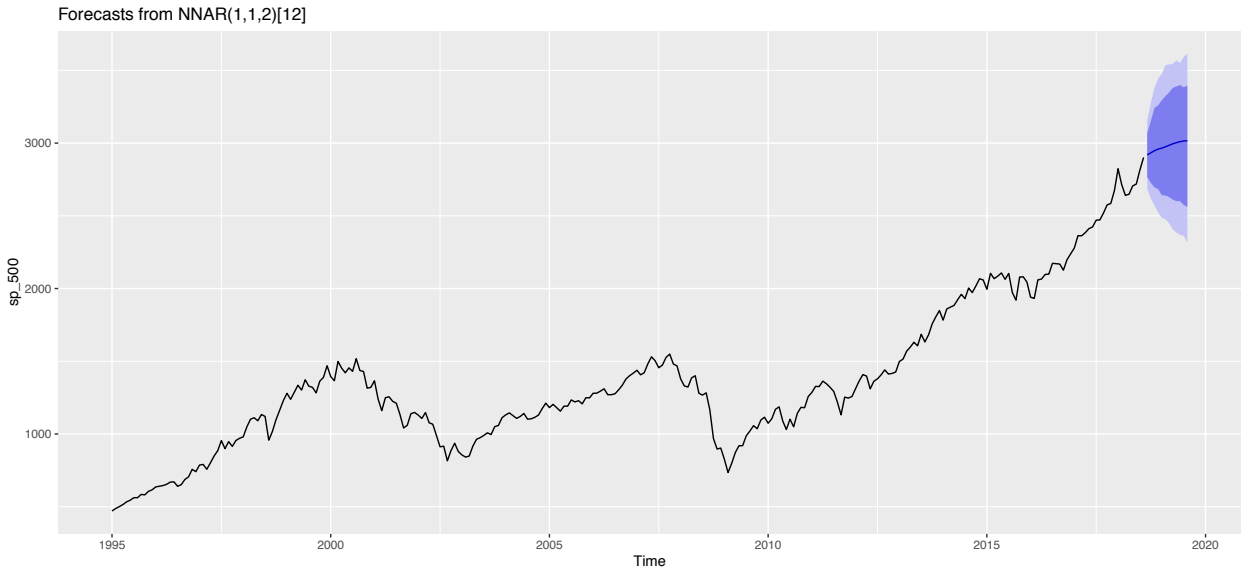


Figure 7 and *Figure 8* present the forecast.

3. Support Vector Machines (SVM)

We use *tune.svm()* function to find the best gamma and cost arguments. Subsequently, we use *svm()* function to fit the model.

Figure 9

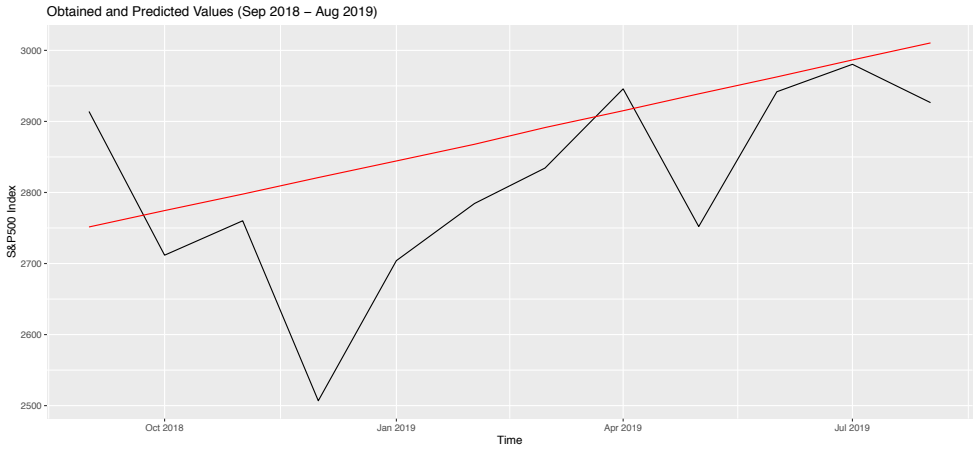


Figure 9 shows the predicted values.

Summary

Figure 10

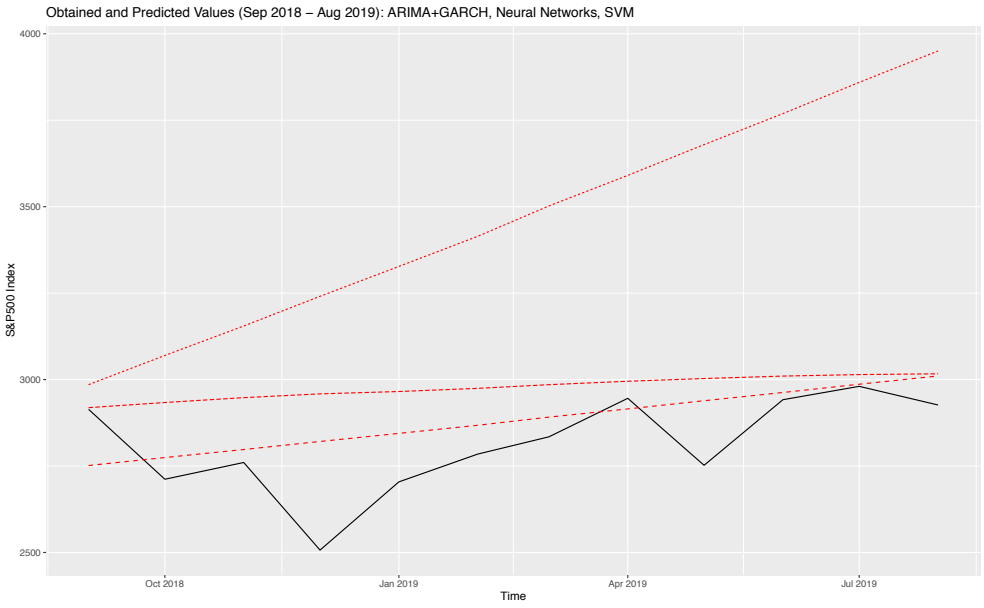


Figure 10 shows that the closest prediction to the actual S&P 500 Index was provided by the Neural Networks Model and the Support Vector Machines Model.

Attachment

```r
# loading data
sp_500 <- ts(GSPC$Adj.Close, start=c(1995,1), end=c(2018,8), freq=12)

# Decomposision
plot(decompose(sp_500, type = "multi"))

acf(sp_500)

pacf(sp_500)

d1 <- diff(sp_500, differences = 1)

plot(d1, ylab = 'First Difference of S&P 500', main = 'Time Series Plot of the First Differences of S&P 500: 1995 - 2018')
acf(d1, main = 'ACF plot of First Difference of S&P 500')
pacf(d1, main = 'PACF plot of First Difference of S&P 500')


# Test for stationarity
Box.test(d1, lag=20, type = 'Ljung-Box')

auto.arima(sp_500)

model_1 <- arima(sp_500, order = c(0,1,0))

sp_500_full <- ts(GSPC$Adj.Close, start=c(1995,1), end=c(2019,8), freq=12)
pred <- predict(model_1, n.ahead=12)

model_1_pred <- pred$pred

plot(model_1_pred)

# ARIMA + GARCH Model and Predictions

garch.fit <- garchFit(formula = ~arma(0, 0) + garch(1, 1), data = d1, trace = FALSE)
garch.pred <- predict(garch.fit, n.ahead = 12, trace = FALSE, mse=c('cond'))

garch.forecast <-ts(garch.pred[,1]+garch.pred[,2])

sp_500_last_value <- sp_500[length(sp_500)]

garch.fc.og <-c()
for(i in 1:12){
  garch.fc.og[i] <- sp_500_last_value +
    sum(garch.forecast[1:i])
}

garch.fc.og


# ARIMA + GARCH Model Prediction Plot

actual <- GSPC$Adj.Close[285:296]

x <- seq(as.Date("2018-09-01"), by="1 month", length.out=12)
var1 <- c(actual)
var2 <- c(garch.fc.og)
df <- data.frame(x, var1, var2)

p <- ggplot(df, aes(x)) +
  geom_line(aes(y=var1), colour="black") +
  geom_line(aes(y=var2), colour="red") +
  labs(y="S&P500 Index", x = "Time") +
  ggtitle("Obtained and Predicted Values (Sep 2018 - Aug 2019)")

p
```

```r
# Neural Network Model

fit <- nnetar(sp_500, lambda=0)
autoplot(forecast(fit,h=12))
autoplot(forecast(fit,h=12, PI=TRUE))

forecast(fit,h=12)

# Neural Network Plot

for_nn <- as.numeric(forecast(fit,h=12)$mean)
x <- seq(as.Date("2018-09-01"), by="1 month", length.out=12)
var1 <- c(actual)
df <- data.frame(x, var1, for_nn)

n <- ggplot(df, aes(x)) +
  geom_line(aes(y=var1), colour="black") +
  geom_line(aes(y=for_nn), colour="red") +
  labs(y="S&P500 Index", x = "Time") +
  ggtitle("Obtained and Predicted Values (Sep 2018 - Aug 2019)")

n

# SVM Model

monthly_data <- GSPC$Adj.Close[1:284]
months <- 1:284
DF <- data.frame(months,monthly_data)
colnames(DF)<-c("x","y")

tuned_parameters <- tune.svm(y~x, data = DF, gamma = 10^(-5:-1), cost = 10^(-3:1))
summary(tuned_parameters )

svmodel <- svm(y ~ x,data=DF, type="eps-regression",kernel="radial",cost=10, gamma=0.1)
```

```r
# SVM ploting predicted values

nd <- 285:296
pred_svm <- predict(svmodel, newdata=data.frame(x=nd))
x <- seq(as.Date("2018-09-01"), by="1 month", length.out=12)
var1 <- c(actual)
df <- data.frame(x, var1, pred_svm)

s <- ggplot(df, aes(x)) +
  geom_line(aes(y=var1), colour="black") +
  geom_line(aes(y=pred_svm), colour="red") +
  labs(y="S&P500 Index", x = "Time") +
  ggtitle("Obtained and Predicted Values (Sep 2018 - Aug 2019)")

s
```

```r
# Summary

summary_data <- data.frame(
  date = seq(as.Date("2018-09-01"), by="1 month", length.out=12),
  var1 = c(actual),
  var2 = c(garch.fc.og),
  for_nn = as.numeric(forecast(fit,h=12)$mean),
  pred_svm = predict(svmodel, newdata=data.frame(x=nd)),
  names = c("Real values", "ARIMA+GARCH", "Neural Networks", "SVM")
)

summary_data_long <- melt(summary_data, id='date')

ggplot(summary_data_long, aes(x=date, y=value)) +
  geom_line(aes(color=variable, linetype = variable)) +
  labs(y="S&P500 Index", x = "Time") +
  ggtitle("Obtained and Predicted Values (Sep 2018 - Aug 2019): ARIMA+GARCH, Neural Networks, SVM ") +
  scale_color_manual(values = c("black", "red", "red", "red")) +
  theme(legend.position='none')
```