Stat 153 Project: Stock Price Models

Holly Cheek, Justin Han, Isaac Schmidt, Leonard Yang

May 06, 2020

Link to Blog: https://www.ocf.berkeley.edu/~ischmidt/2020/05/04/stat-153-project/

1 Executive Summary

Due to changing consumer preferences, Mediocre Social Network Apps Incorporated's (MSNAI) stock prices have been struggling for a few years. In this report, we will examine the Stocks data set from MSNAI, which includes closing stock prices for every trading day from January 2nd, 2015 to September 30th, 2019. In order to model this data set, we designed a model composed of a logarithmic Variance Stabilizing Transform, a quadratic trend, and an autoregressive noise term. Then, we used this model to forecast the first ten trading days in October 2019. As MSNAI had hoped, our forecast shows that its stock price will increase for at least the first few days of the last quarter in 2019.

2 Exploratory Data Analysis

We begin by plotting our stocks data in Figure 1 to examine some characteristics of the time series. Right away, we notice a decreasing trend and consequently, a non constant mean. From the ever so slightly curved trajectory of the data, we conclude that the underlying trend is likely captured by a nonlinear function of time. We might also recognize a few significant peaks in the data and attribute that to some sort of seasonal effect. However, the periodogram only shows a large spike at index 1, meaning the long-term trend is predominant over any seasonality. Moreover, the height of each peak seems to decrease as time increases, allowing us to assume that our data is marginally heteroscedastic.

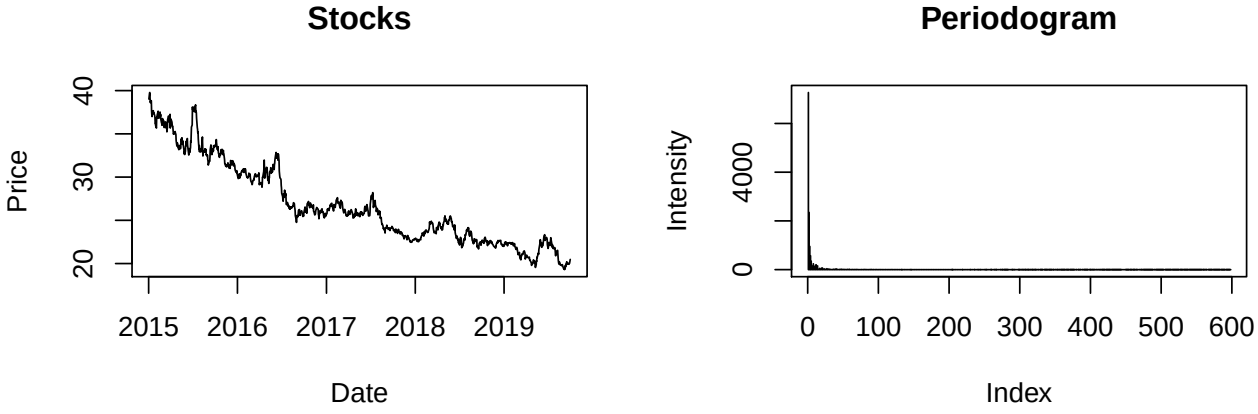

Figure 1: Stocks data from the beginning of 2015 to the end of 2019, and the resulting periodogram.

3 Models

Here are the details of our five models, which vary from simple to complex.

3.1 Model 1 - First Difference

The first goal when building any model is to reach stationarity. As shown earlier, this time series is clearly not stationary—the mean decreases with time. To get rid of this trend, the first thing we do is difference the data. Now, each point D_t in our transformed series equals $Y_t - Y_{t-1}$. Figure 2 shows the differenced data.

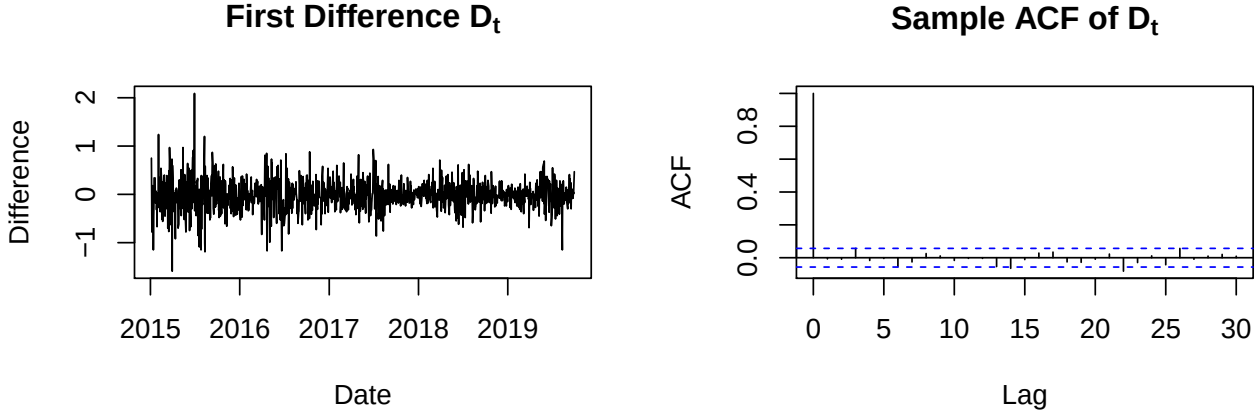

Figure 2: First difference of the stocks price data, and its correlogram.

The mean remains constant over time, and while the variance may be slightly decreasing over time, it still looks relatively stable. Barring a few outliers early on, we pass the shoebox test. Additionally, the autocorrelogram and partial autocorrelogram both have no truly significant large lags beyond 0. Not only is the differenced data stable, but it is likely to be white noise as well! This means that we do not need to further model the differenced data as ARMA or some other process. Therefore, for the original stocks dataset, our model is as follows:

$$Y_t = Y_{t-1} + \mu_D + W_t \tag{1}$$

where μ_D is the mean of the differenced data, and W_t is white noise. While the mean of the differenced data looks to be 0, it is actually around -.015. We would expect it to be negative, as there is a clear downward trend in the original data.

3.2 Model 2 - ARIMA

This next model was generated by using the `auto.arima` function. This function fits a subset of all possible ARIMA models, by trying different combinations of p, d, and q up to a reasonable limit. It then chooses the model with the lowest AIC, given that it is invertible and causal.

For the stocks data, `auto.arima` returned the ARIMA(3,1,1) model. Shown in Figure 3 are diagnostic plots for this model.

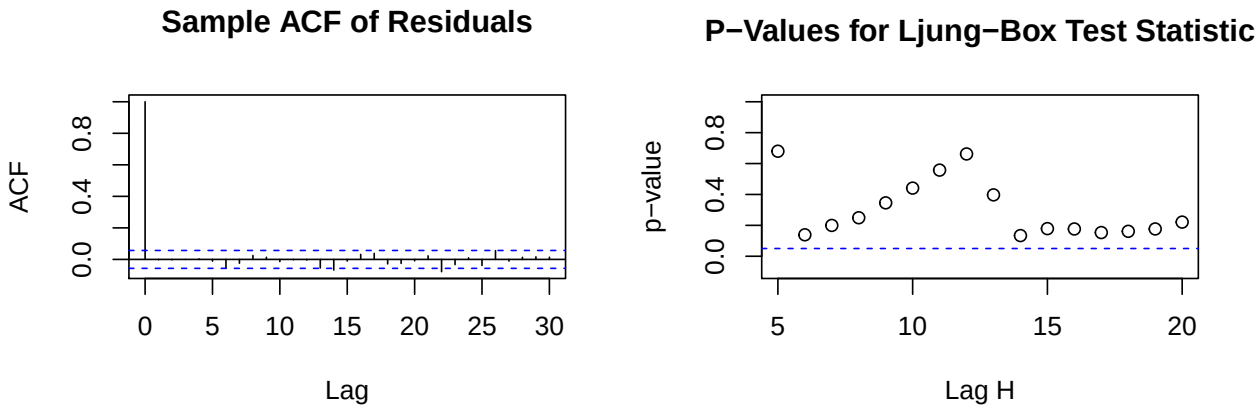

Figure 3: Autocorrelogram and p-values of the Ljung-Box test for the residuals of the ARIMA(3,1,1) model.

The residuals of the ARIMA(3,1,1) model are stable, and look like white noise. The ACF plot also shows that most values are under the 95% confidence interval bands. The Ljung-Box test shows no significant p-values across all lags, which means it fails to reject the null hypothesis the data follows this model. This makes us believe that the fit of this model is acceptable.

3.3 Model 3 - Month and Weekday Indicators

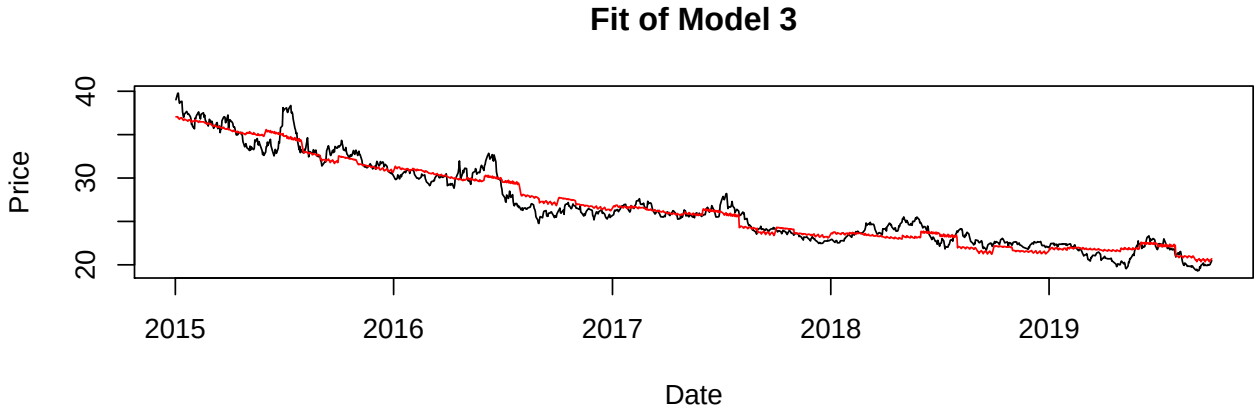

Figure 4: Indicator model fit on stock price data.

In this model, we combine models for each of trend, seasonality, and noise to build a full model. To model trend, we first use the quadratic function of time. To model seasonailty, this model utilizes a parametric fit through two groups of indicators. One set of indicator variables corresponds to weekdays–only Monday through Friday since trading days are only weekdays. The second group of indicator variables corresponds to the month that the stock price was recorded. We used these indicators because we felt that they would have the highest effect on volatility without bringing in issues of collinearity. We also included an interaction between the weekday and month to illustrate a possible dependency between the two. Without this interaction we would have 17 weights, but now we have 60 weights for each unique combination of month and weekday. As a result, our model is defined as:

$$Y_t = \alpha_i(\text{I(month)}) + \lambda_i(\text{I(weekday)}) + \tau_{ij}(\text{I(month)} \cdot \text{I(weekday)}) + \beta_1 t + \beta_2 t^2 + Z_t \tag{2}$$

$$\alpha_i\text{: } i \in \{1, 2, \ldots, 12\};\ \lambda_i\text{: } i \in \{1, 2, \ldots, 5\};\ \tau_{ij}\text{: } i \in \{1, 2, \ldots, 12\},\ j \in \{1, 2, \ldots, 5\} \tag{3}$$

where Z_t is an AR(1) process, and α, λ, τ, and β are calculated using linear regression. The resulting fit is shown in Figure 4. The model seems to fit the stocks data set well, but with so many parameters, the model could possibly be overfitting.

From Figure 5, we see a cutoff at lag 1 on the PACF of the residuals–indicating they follow an AR(1) process.

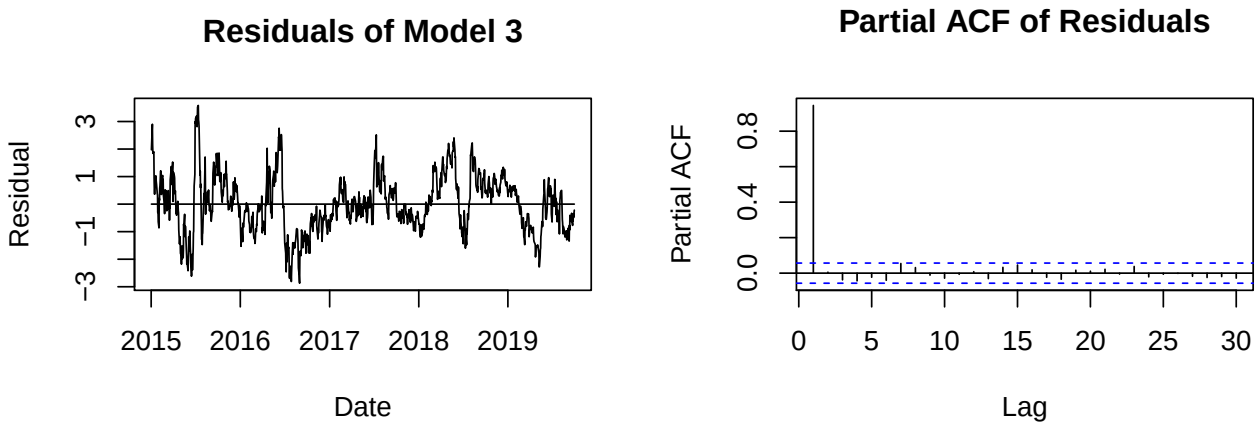

Figure 5: Residuals of the model fit and their partial autocorrelogram.

3.4 Model 4 - Sinusoid Interactions

When pursuing stationarity, we aim to model the signal as best as we can in hopes of having residuals that resemble white noise. In this model, we decide to capture trend with a quadratic function of time to as we believe it will account for the slight curvature in the data. As mentioned during the EDA section, we see noticeable peaks occuring somewhat regularly throughout our time series, making it reasonable to examine any potential seasonal components that may be present. The periodogram in Figure 1 has a significant spike at index $j = 1$ which means that our data has a frequency of $f = \frac{j}{n} \approx 0.0008$ where $n = 1194$, the total number of observations in our stocks data.

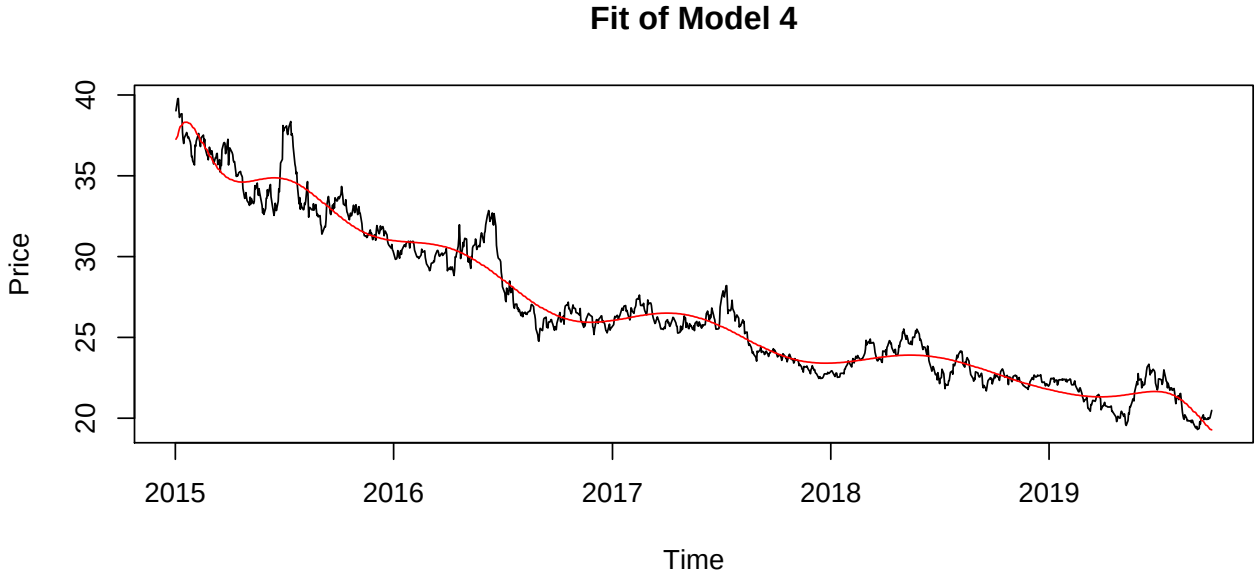

Figure 6: Quadratic and sinusoid interaction model fit.

We then use this frequency in our sine and cosine functions to approximate the seasonal component. By letting $s = sin(2\pi tf)$ and $c = cos(2\pi tf)$ where t is time and f is the frequency found from the periodogram, we ctake a four way interaction between t, t^2, s, and c to arrive at a parametric model for our signal:

$$f_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \beta_3 s + \beta_4 c + \beta_5(t \cdot t^2) + \beta_6(t \cdot s) + \beta_7(t^2 \cdot s) + \beta_8(t \cdot c) + \beta_9(t^2 \cdot c) + \beta_{10}(s \cdot c) + \beta_{11}(t \cdot t^2 \cdot s) + \beta_{12}(t \cdot t^2 \cdot c) + \beta_{13}(t \cdot s \cdot c) + \beta_{14}(t^2 \cdot s \cdot c) + \beta_{15}(t \cdot t^2 \cdot s \cdot c) \quad (4)$$

We take the cross between all four of these terms in order to capture the non constant peak sizes along with the decreasing pattern in the overall trend as shown in Figure 6. The residual plots for this model look very similar to the residuals in the previous model as shown in Figure 6. An AR(1) model is appropriate since there is a tailing effect in the spikes of the ACF along with a cutoff after lag 1 in the PACF. In turn, this model is defined as:

$$Y_t = f_t + Z_t \quad (5)$$

where Z_t is an AR(1) process.

3.5 Model 5 - Log Transform, Quadratic Trend, and AR Residuals

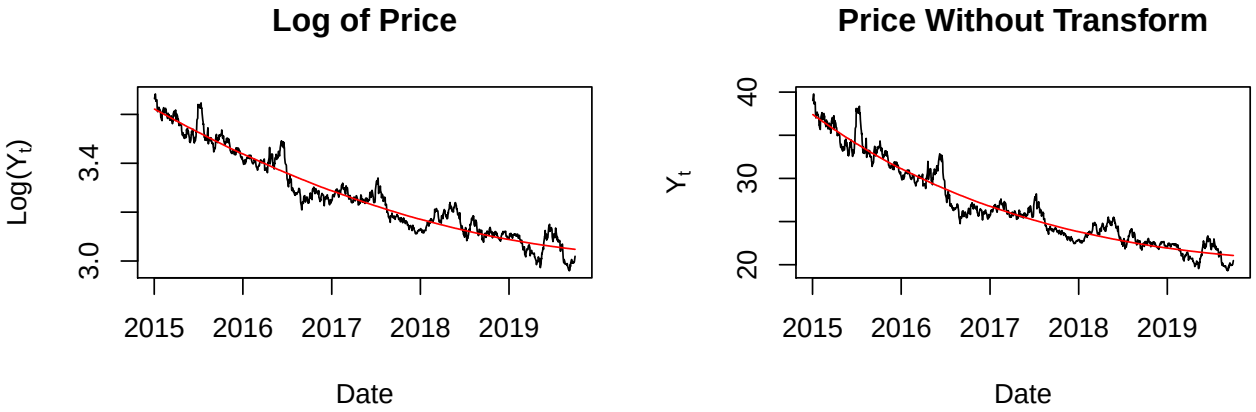

Figure 7: Log of the stock price over time, with a quadratic fit, and the back-transformed fit.

The next model we tried is another parametric fit. However, for this model, we used a variance stabilizing transform to try to account for the larger variance at the beginning. The square root transformation did not have much of an effect, so we decided to take the natural logarithm of the original stocks dataset. There was no real rationale for this, but Figure 7 shows the variance to be a little more constant over time. First, we fit a linear model to the logged data, but there was curvature in the residuals so we tried a quadratic fit. The residuals of the quadratic model passed the shoebox test, but they still look similar to the other residual plots we've seen. The ACF is exponentially decreasing, and the PACF has one tall spike at 1, so it makes sense to model the residuals as an AR(1) process. Thus, this model is defined as the following:

$$\log(Y_t) = \beta_0 + \beta_1 t + \beta_2 t^2 + Z_t \quad (6)$$

where Z_t is an AR(1) process, and β is calculated using linear regression. As Figure 7 shows, the logged data is modeled fairly well by this process, after we exponentiate our fitted values to back-transform our data.

4 Cross Validation and Model Selection

While all of our models seem to fit reasonably well in-sample, we performed cross-validation to determine each model's fit out of sample. We split the stocks data into training and test sets, with a rolling window of 10 points for our test set. To ensure our smallest training set had a sufficient amount of data, the first fold's training set contained the first 604 data points, and the test set contained the next 10. Then our second fold contained the first 614 data points, and so on. For each fold, we trained the model on the training set, and compared that model's forecast with our test set. We calculated two scores—sum of squared errors and root mean squared error. In Table 1 below, SSE represents the sum of all squared errors across all predictions for each fold, and RMSE represents the average root mean squared error score for each fold.

Model	Description	SSE	RMSE
1	First Difference only	211.41	0.507
2	ARIMA(3,1,1) from auto.arima	218.91	0.510
3	Quadratic Trend, Monthly and Weekday Indicators, AR(1)	1629.92	1.371
4	Quadratic and Sinusoid Interactions, AR(1)	3224.25	1.923
5	Log VST, Quadratic Trend, AR(1)	200.99	0.498

Table 1: The cross-validation scores for all five models.

The model that scored best in both metrics was our last model, the logarithm variance stabilizing transform, with a quadratic fit and AR(1) modeled on the residuals. Therefore, we choose this model for prediction.

5 Results

The model we select for the stocks data is as follows:

$$\log(Y_t) = \beta_0 + \beta_1 t + \beta_2 t^2 + Z_t \tag{7}$$

$$Z_t - \phi_1 Z_{t-1} = W_t \tag{8}$$

Each β is calculated with linear regression, and ϕ_1 is calculated using the conditional sum of squares and maximum likelihood estimator from the `arima` function. W_t is a white noise process. The parameters are listed in Table 2 below.

5.1 Parameter Estimates

Parameter	Estimate	Std. Error
β_0	3.622	$3.642 * 10^{-3}$
β_1	$-7.978 * 10^{-4}$	$1.408 * 10^{-5}$
β_2	$2.653 * 10^{-7}$	$1.141 * 10^{-8}$
ϕ_1	0.964	$7.587 * 10^{-3}$

Table 2: The estimated parameters for Model 5.

Clearly, each parameter is significant as the estimates are much larger than their standard errors.

5.2 Predictions

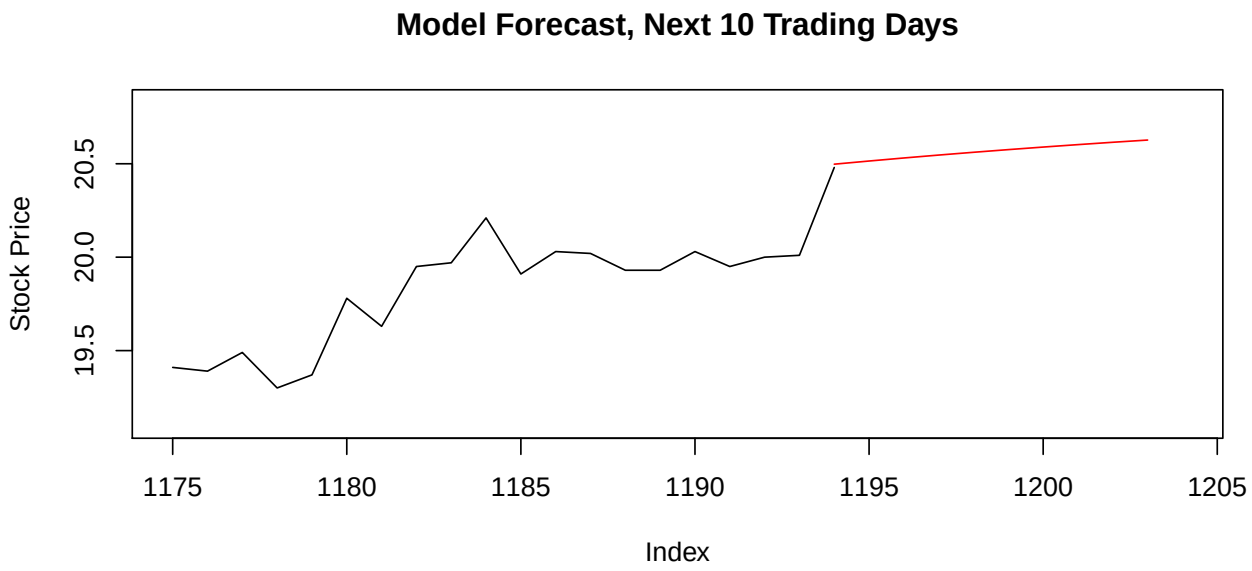

Figure 8: Our forecast for the next 10 data points, from Model 5.

Figure 8 shows our forecast of stock price for the first 10 trading days in October, following the previous month of September. We predict a continuous slight increase in stock price, following the trend of the past month. Surprisingly, given our model, our predictions look close to linear, although there is in fact some slight curvature.

Of course, our model's predictions come with uncertainty, which increases as we predict farther into the future. One danger of using a parametric model such as this one is that the predictions can be very off for times far out of sample. However, we are only predicting 10 additional points based off close to 1200, and this is the model that performed best in cross-validation.

6 Appendix

```
library(astsa)
library(forecast)

#Load Data
load('stocks.Rdata')

#Model 1 - First Difference
differenced = diff(stocks$price)

acf(differenced)
pacf(differenced)

#Model 1 Cross-Validation
train_splits = seq(604,1194-10,10)
rmses = c()
sses = c()
for (index in train_splits){
  train = stocks$price[1:index]
  test = stocks$price[(index+1):(index+10)]

  differenced_train = diff(train)
  mean_diff = mean(differenced_train)
  predictions = train[index] + (1:10)*mean_diff
  sse = sum((predictions - test)^2)
  sses = c(sses,sse)
  rmse = sqrt(mean((predictions - test)^2))
  rmses = c(rmses,rmse)
}
sum(sses)
mean(rmses)

#Model 2 - auto.arima

auto.arima(stocks$price, trace = TRUE, allowdrift = FALSE)
AutoArimaStock <- sarima(stocks$price, p=3,d=1,q=1)

#Model 2 Cross-Validation
sumSquares <- c(0)
rootMSE <- c(0)
for(entry in seq(604,1184,10)){
  train_set <- stocks$price[1:entry]
  test_set <- stocks$price[c(entry+1, entry+2, entry+3, entry+4, entry+5, entry+6, entry+7, entry+8, en
  #
  forecastModelEquation <- arima(train_set, order=c(3,1,1), include.mean = FALSE)
  forecastModel <- predict(forecastModelEquation, n.ahead=10)$pred

  #
  SSE <- sum((test_set-forecastModel)^2)
  RMSE <- sqrt(mean((test_set-forecastModel)^2))

  #
```

```
  sumSquares <- c(sumSquares, SSE)
  rootMSE <- c(rootMSE, RMSE)
}
sum(sumSquares)
mean(rootMSE)

#Model 3 - Weekday and Monthly Indicators
i_model <- lm(data = stocks, price ~ index + I(index^2) + I(month) * I(weekday))
summary(i_model)

# plot indicator model on the original data
plot.ts(stocks$price, type = 'l', main = "Stocks Time Series", ylab = "Price", xlab = "Time")
lines(i_model$fitted.values, col = 'red', type = 'l')

# residual plot
plot(i_model$residuals, type = 'l', ylab = "Residuals", xlab = "Time", main = "Residual Plot for Indica
abline(h = 0)

# acf & pacf of residuals
acf(i_model$residuals)
pacf(i_model$residuals)

#Model 3 Cross-Validation
rmse <- c()
sse <- c()
start <- 604
end <- length(stocks$price)
time <- stocks$index

for (i in seq(start, end - 10, 10)) {
  train_set <- stocks[1:i,]
  test_set <- stocks[c( i + 1, i + 2, i + 3, i + 4, i + 5, i + 6, i + 7, i + 8, i + 9, i + 10), ]

  # indicator model forecasting
  i_model <- lm(data = train_set, price ~ index + I(index^2) + I(month) * I(weekday))

  #inidicator only models seasonality and quadratic trend is modeled by time variable
  i_model_forecast <- predict(i_model, test_set)

  # residual forecasting with AR(1) model
  residuals <- i_model$fitted.values - train_set$price
  forecast_resid <- sarima.for(residuals, n.ahead = 10, p = 1, d = 0, q = 0)$pred

  # RMSE calculation
  rmse1 <- sqrt(mean(((i_model_forecast + forecast_resid) - test_set$price)^2))
  sse1 <- sum(((i_model_forecast + forecast_resid) - test_set$price)^2)
  rmse <- c(rmse, rmse1)
  sse <- c(sse, sse1)
}
mean(rmse)
sum(sse)

#Model 4 - Quadratic and Sinusoid Interaction
```

```r
#periodogram used to identify index of significant frequency
pgram <- abs(fft(sp)[2:nrow(stocks)])^2/nrow(stocks)
plot(pgram[c(1:floor(nrow(stocks)/2))], type = "h", ylab = "Significance")

#index with largest value
j <- which(pgram==max(pgram[c(1:floor(nrow(stocks)/2))]))

#frequency
f <- j/nrow(stocks)

#period
t <- 1/f

#summary
info <- c(j,f,t)
names(info) <- c("Max_index","Frequency", "Period")
info

#sinusoids with frequency found above
single_sin <- sin(2*pi*time*f)
single_cos <- cos(2*pi*time*f)

#fit with cross of quadratic and seasonal trend components
qs_model <- lm(price ~ time * time2 * single_sin * single_cos, data = stocks)

#note the complexity
summary(qs_model)

#plot qs_model
plot(time, sp, type = "l", ylab="Price", xlab = "Time")
lines(time, qs_model$fitted.values, col = "red")

plot(qs_model$residuals, main = "Residuals for qs_model", type = "l")
abline(h=0, col = "red")

par(mfrow = c(2,1))
acf(qs_model$residuals, lag.max = 200)
pacf(qs_model$residuals, lag.max = 200)

#fit AR(1) to the residuals
ar1 <- sarima(qs_model$residuals,1,0,0)

#Model 4 Cross-Validation

window <- seq(604,nrow(stocks)-10,10)
rmse <- c()
for (i in window) {
  #define training and testing sets
  train <- stocks$price[1:i]
  test <- stocks$price[(i+1):(i+10)]
  #specify time range
  range <- 1:i
  #periodogram to find optimal frequency for given test set
```

```
  #fit with cross of quadratic and seasonal trend components
  qsm <- lm(train ~ range * I(range^2) * I(sin(2*pi*range*f))* I(cos((2*pi*range*f))))
  #fit AR(1) to the residuals
  ar_residuals <- arima(qsm$residuals, order = c(1,0,0), include.mean = FALSE)
  #predict using test set to measure quality of predictions
  predictions <- predict(qsm, data.frame(range = ((i+1):(i+10)))) + predict(ar_residuals, n.ahead = 10)$
  #compute rmse
  rmse <- append(rmse, sqrt(mean((predictions - test)^2)))
}

#average rmse across all windows
rmse
mean(rmse)

#Model 5 - Log VST, Quadratic, AR(1)

#Take Log
stocks$log_price = log(stocks$price)

#Fit Quadratic Model
quad_model = lm(stocks$log_price ~ stocks$index + I(stocks$index^2))
summary(quad_model)

#Check Residuals
acf(quad_model$residuals)
pacf(quad_model$residuals)

#Model AR(1) on Residuals
residuals_ar = arima(quad_model$residuals, order = c(1,0,0), include.mean = FALSE)

#Model 5 Cross-Validation
train_splits = seq(604,1194-10,10)
rmses = c()
sses = c()
for (index in train_splits){
  train = stocks$price[1:index]
  test = stocks$price[(index+1):(index+10)]

  log_train = log(train)
  x = 1:index
  quad_model = lm(log_train ~ I(x) + I(x^2) )

  residuals_ar = arima(quad_model$residuals, order = c(1,0,0), include.mean = FALSE)

  log_predictions = predict(quad_model, data.frame(x = ((index+1):(index+10))))
  + predict(residuals_ar,n.ahead = 10)$pred

  predictions = exp(log_predictions)

  sse = sum((predictions - test)^2)
  sses = c(sses,sse)
  rmse = sqrt(mean((predictions - test)^2))
  rmses = c(rmses,rmse)
```

```
}
sum(sses)
mean(rmses)

#Model 5 Forecast

x = stocks$index
quad_model = lm(stocks$log_price ~ x + I(x^2))
residuals_ar = arima(quad_model$residuals, order = c(1,0,0), include.mean = FALSE)
log_forecast = predict(quad_model, data.frame(x = ((nrow(stocks)+1):(nrow(stocks)+10))))
+ predict(residuals_ar,n.ahead = 10)$pred
forecast = exp(log_forecast)

#Output to CSV File
write.table(forecast, file = 'forecasts.csv', sep = ',', row.names = FALSE, col.names = FALSE)

#Plots

#Plot in Blog Post
stocks$month = as.numeric(stocks$month)
quarter2 = stocks[(stocks$month >= 4) & (stocks$month <= 6) & (stocks$year == 2019),]
quarter3 = stocks[(stocks$month >= 7) & (stocks$month <= 9) & (stocks$year == 2019),]
quarter3 = stocks[(stocks$date >= 18075) & (stocks$date <= 18169),]
plot(x = quarter2$date, y = quarter2$price, col = 'forestgreen', xlim = c(17987,18170),
     ylim = c(18,23.5), type = 'l', xlab = 'Month', ylab = 'Price',
     main = 'MSNAI 2019 2nd and 3rd Quarter Stock Price')
lines(x = quarter3$date, y = quarter3$price, col = 'red')
legend('bottomleft', legend=c('2nd Quarter Stock Price', '3rd Quarter Stock Price'),
       col=c('forestgreen', 'red'),lty=1, cex=0.8)

#Figure 1
par(mfrow = c(1,2))
plot(stocks$date, stocks$price, type = "l", ylab = "Price", xlab = "Date", main = "Stocks")
pgram <- abs(fft(stocks$price)[2:nrow(stocks)])^2/nrow(stocks)
plot(pgram[c(1:floor(nrow(stocks)/2))], type = "h", xlab = 'Index', ylab = 'Intensity',
     main = 'Periodogram')

#Figure 2
differenced = diff(stocks$price)
par(mfrow = c(1,2))
plot(stocks$date[2:nrow(stocks)],differenced,type = 'l', ylab = "Difference", xlab = "Date",
     main = expression(bold("First Difference "*D[t])))
acf(differenced, main = expression(bold("Sample ACF of "*D[t])))

#Figure 3
ppq = 4
rs = arima(stocks$price, order = c(3,1,1))$residuals
nlag = 20
pval = rep(0,nlag)
#stolen from sarima function
for (i in (ppq+1):nlag){
  u = Box.test(rs, i, type = "Ljung-Box")$statistic
  pval[i] = pchisq(u, i-ppq, lower.tail=FALSE)
```

```
}
par(mfrow = c(1,2))
acf(rs, main = 'Sample ACF of Residuals')
plot( (ppq+1):nlag, pval[(ppq+1):nlag], xlab = "Lag H", ylab = "p-value", ylim = c(-.1, 1),
      main = "P-Values for Ljung-Box Test Statistic")
abline(h = 0.05, lty = 2, col = "blue")

#Figure 4
time <- 1:nrow(stocks)
time2 <- time^2
# model creation
i_model <- lm(stocks$price ~ time + I(time^2) + I(stocks$month) * I(stocks$weekday))
# plotting fitted indicator model on price data
plot(x = stocks$date, y = stocks$price, type = 'l', xlab = 'Date', ylab = 'Price',
     main = 'Fit of Model 3')
lines(x =stocks$date, y= i_model$fitted.values, col = 'red', type = 'l')

#Figure 5
par(mfrow = c(1,2))
plot(x = stocks$date, y = i_model$residuals, type = 'l', xlab = 'Date', ylab = 'Residual',
     main = 'Residuals of Model 3')
lines(x = stocks$date, y= rep(0, nrow(stocks)))
pacf(i_model$residuals, main = 'Partial ACF of Residuals')

#Figure 6
j <- which(pgram==max(pgram[c(1:floor(nrow(stocks)/2))]))
#frequency
f <- j/nrow(stocks)
#period
t <- 1/f

#sinusoids with frequency found above
single_sin <- sin(2*pi*time*f)
single_cos <- cos(2*pi*time*f)

#fit with cross of quadratic and seasonal trend components
qs_model <- lm(price ~ time * time2 * single_sin * single_cos, data = stocks)

#plot qs_model
plot(stocks$date, stocks$price, type = "l", ylab="Price", xlab = "Time", main = "Fit of Model 4")
lines(stocks$date, qs_model$fitted.values, col = "red")

#Figure 7
stocks$log_price = log(stocks$price)
quad_model = lm(stocks$log_price ~ stocks$index + I(stocks$index^2))
par(mfrow = c(1,2))
plot(x = stocks$date, y = stocks$log_price,type = 'l', xlab = 'Date',
     ylab = expression("Log("*Y[t]*")"), main = 'Log of Price')
lines(x = stocks$date, y = quad_model$fitted.values, col = 'red')

plot(x = stocks$date, y = stocks$price,type = 'l', xlab = 'Date', ylab = expression(Y[t]),
     main = 'Price Without Transform')
lines(x = stocks$date, y = exp(quad_model$fitted.values), col = 'red')
```

```r
#Figure 8
x = stocks$index
quad_model = lm(stocks$log_price ~ x + I(x^2))
residuals_ar = arima(quad_model$residuals, order = c(1,0,0), include.mean = FALSE)
log_forecast = predict(quad_model, data.frame(x = ((nrow(stocks)+1):(nrow(stocks)+10))))
+ predict(residuals_ar,n.ahead = 10)$pred
forecast = exp(log_forecast)

plot(x = 1175:1194, y = stocks$price[1175:1194], type = 'l', xlim = c(1175, 1204),
     ylim = c(min(stocks$price[(1194-20):1194]) - .2, max(forecast)+.2), xlab = 'Index',
     ylab = 'Stock Price', main = 'Model Forecast, Next 10 Trading Days')
lines(x = 1194:1203, y = forecast, col = 'red')
```