Univariate Time Series Stock Prediction

Shihao Niu

November 25, 2020

Abstract

Recently, stock market price prediction has become an emerging field of academic interests, due to its finical implications. A successful algorithm that can predict the stock price can surely result in a significant return of investments. While statistical models are being used to forecast a time series, recently many machine learning models are being developed to create forecasting models. In this Capstone project, my aim is to implement several statistical models as well as machine learning models to explore the engineering problems of creating such algorithms.

1 Motivation and Introduction

Although a successful prediction of stock prices can result in a huge profits for the investors, making such algorithm highly sought after, my goals for this project are much smaller in scope. The project will focus on implementing different statistical models for time series forecasting, such as Holt's forecasting model, Holt-Winter's forecasting model, ARIMA models, and machine learning models, such as Single layer LSTM model and vertically stacked LSTM model.

2 Data Analysis

This project will use data from Yahoo! Finance. I have surveyed other exchange markets and online services to source the historical data, but Yahoo! Finance is free, easy to use and has relatively large free usage of API calls. The exact codes of downloading the data can be found in the project's GitHub repo https://github.com/asherniu/Capstone.

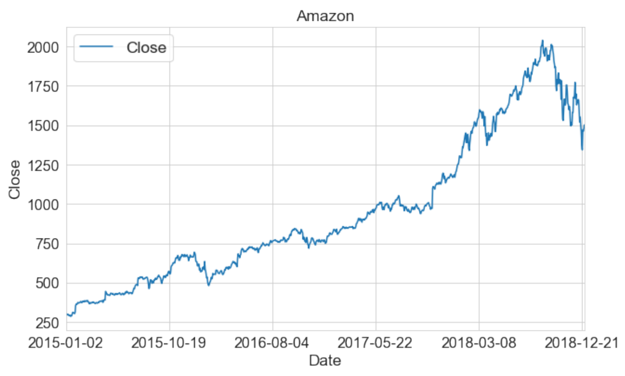

Figure 1: Amazon stock price, 2015-2018

For this particular project, I am only considering the closing price recorded at the end (4:00PM EST) of trading day. The frequency of the data is one trading day increment. And while volume of trading is an excellent feature to increase model complexity, it is not considered in this project. The stock being analyzed is Amazon.com Inc. (AMZN) during the trading period from 01/01/2015 to 12/31/2018. In other words, this is a univariate time series forecasting problem, with the closing price of the stocks being the variable I collected over time.

Another important thing I need to mention is that stock markets are not open every day. Typically, the average number of trading days for U.S. markets is around 252 days, but it's not warranted for every year. For this reason, I have to include the historical stock price from the whole year.

Lastly, since there are 4 years of trading data, I allocated the first 3 years, 2015, 2016, 2017 for training purposes, and used the last year 2018 for testing purposes. There are 1006 trading days in total, where the training set has 755 days and testing set has 251 days.

3 Models and Experiments

3.1 Holt's Model

Common senses indicate that recent events should have higher weights on the stock prices compared to older, more distant events. In this model, I used exponential smoothing to smooth the time series data by giving more weights to recent events, rather than using a moving average where all events are equally weighted. Mathematically speaking, the model can be explained as follows:

$$S_t = \alpha y_{t-1} + (1-\alpha)S_{t-1} \tag{1}$$

, where S is the smoothed value, α is the smoothing factor, y is the data point of time series, and t is the time tamp. To forecast, compute

$$S_{t+1} = \alpha y_n + (1-\alpha)S_t \tag{2}$$

This simple exponential smoothing model, also dubbed as Holt linear model, does not capture the hidden trends in the data. Clearly from figure 2, the prediction is just a horizontal line. The choice of smoothing factor will only change the y-intercept of the prediction.

3.2 Holt-Winter's Model

Improvements can be made to previous model. In order to capture the trends single exponential smoothing couldn't, double exponential smoothing introduces a second equation to model the trend. Mathematically, it can be expressed as follows:

$$S_t = \alpha y_{t-1} + (1-\alpha)(S_{t-1} + b_{t-1}) \tag{3}$$

$$b_t = \beta(S_t - S_{t-1}) + (1-\beta)b_{t-1} \tag{4}$$

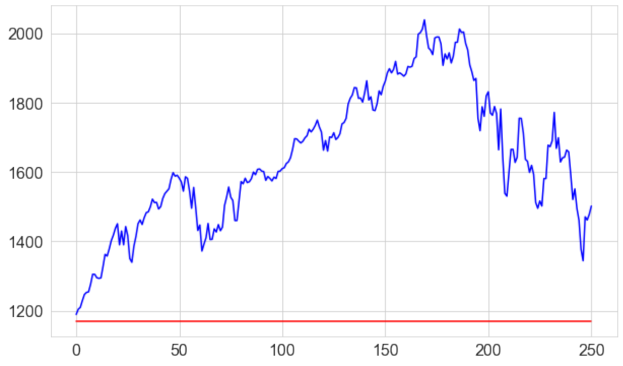

Figure 2: Simple Exponential Smoothing

, where the newly added b is the estimation of the trend and β is the smoothing factor for the trend. The rest of the variables correspond to that of Holt's Model. To forecast, compute

$$F_{t+m} = S_t + mb_t \tag{5}$$

In this model, the trends are adjusted by adding the trend of the previous time period to the last smoothed value. Results can be found in the figure 3. Still, the predictions are pretty awful as they still form a linear line, instead of showing zig-zag patterns that can be found in real stock price graph.

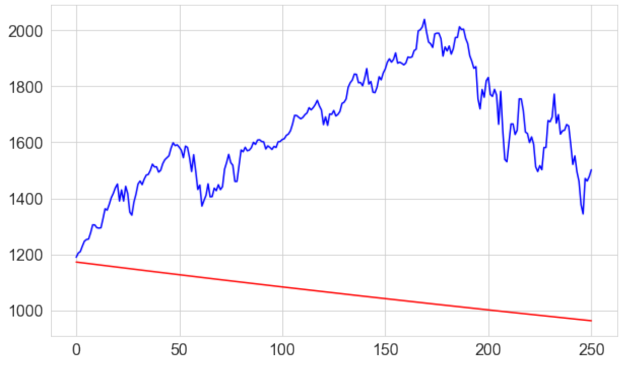

Figure 3: Double Exponential Smoothing

Finally, Holt-Winters method, also known as the triple exponential smoothing, introduces a third equation to capture the seasonality of the graph as well. They can be expressed as follows:

$$S_t = \alpha(y_{t-1} - I_{t-L}) + (1-\alpha)(S_{t-1} + b_{t-1}) \quad (6)$$

$$b_t = \beta(S_t - S_{t-1}) + (1-\beta)b_{t-1} \quad (7)$$

$$I_t = \gamma(y_t - S_t) + (1-\gamma)I_{t-L} \quad (8)$$

, where the newly added variables L denotes the seasonality period, I the estimation of the seasonal component, and $gamma$ the smoothing factor for seasonal component. In this model, I set the cycle of graph to 252, which is the typical number of trading days in U.S. markets. I have also experimented with other numbers such as 30 days for monthly cycle, 90 days for quarterly or 180 days for semiannual cycle, but test results indicated that those seasonal lengths greatly degrade the performances. The final performance can be found here in 4.

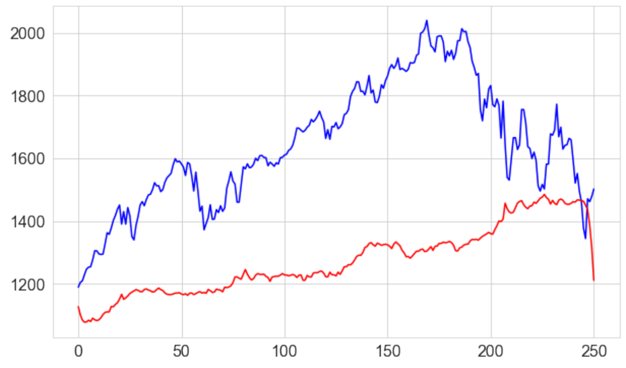

Figure 4: HW Triple Exponential Smoothing

3.3 ARIMA

The third model I implemented is Auto Regressive Integrated Moving Average (ARIMA), which is a combination of Auto Regressive (AR) model and Moving Average (MA) model. This model makes use of 3 main parameters. p, the number of lag observations, which helps adjust the time series. d, the order of differencing between timestamps to make time series into stationary, and q, the size of the moving average window.

Auto regressive means that the forecast of the variable is a linear combination of past values of said variable, and moving average uses past forecast errors in a regression-like combination. In this context, integration is the reverse of differencing, because in order to use AR or MA, the time series needs to be stationary.

Mathematically speaking, a stationary time series data y_t, giving an ARIMA model with parameters p, q and d, can be expressed as follows:

$$y'_t = c + \phi_1 y'_{t-1} + ... + \phi_p y'_{t-p} \quad (9)$$

$$+\theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} + \epsilon_t \quad (10)$$

, where c is a constant, y'_t is the dth order differenced series, ϵ_t the white noise, ϵ_{t-q} the lagged errors, and ϕ, θ the corresponding constant factors.

Now, in order to apply ARIMA model to our data, the data must satisfy the requirements. The stock prices, after some order of differencing, need to be stationary. We can quickly verify this by plotting the first discrete difference of the series. Referring to 5, it is clear that the mean and variance don't change over time, showing a constant location.

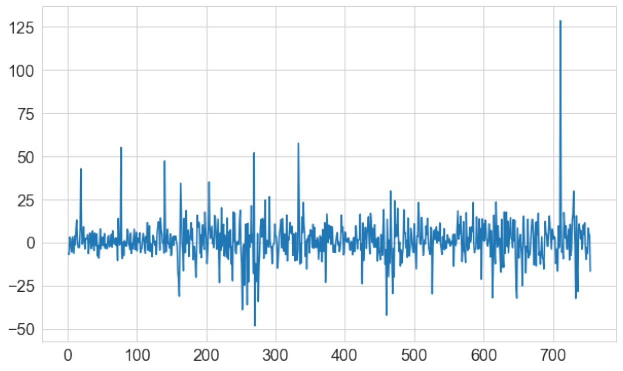

Figure 5: First discrete difference of the series

Furthermore, the lag plots of first 4 lag orders indicate that there are indeed auto correlations between the feature, with respect to the lag order from 1-4. Finally, plotting the Auto Correlation Function (ACF) plot and Partial Auto Correlation Function (PACF) plot can help determine the value of p and q. Referring to figure 7 and 8, it is clear that the ACF is sinusoidal, and PACF has a significant spike at lag 1, indicating that p should be 1, and q should be 0.

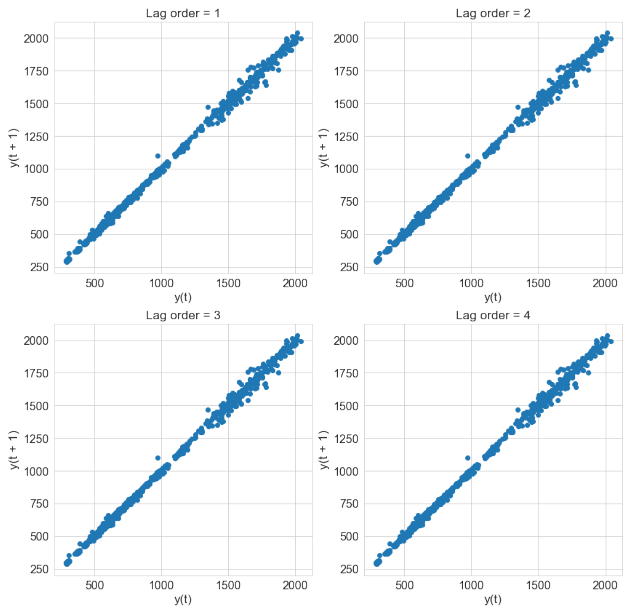

Figure 6: Lag plot for time series

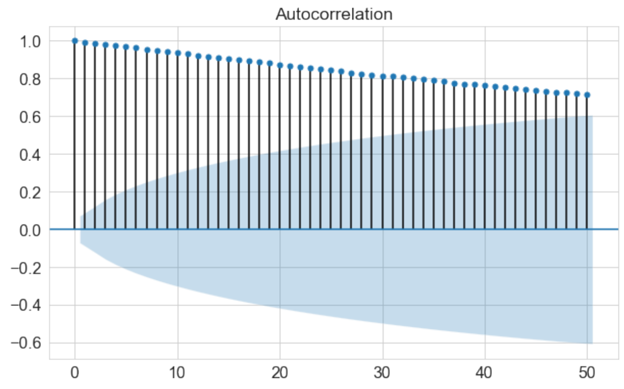

Figure 7: ACF Plot

Finally, with p=1, d=1 and q=0, I created an ARIMA model and fit the training data. The final prediction can be found in figure 9. Compared to Double Exponential Smoothing model, ARIMA somewhat correctly predicts that the price is going

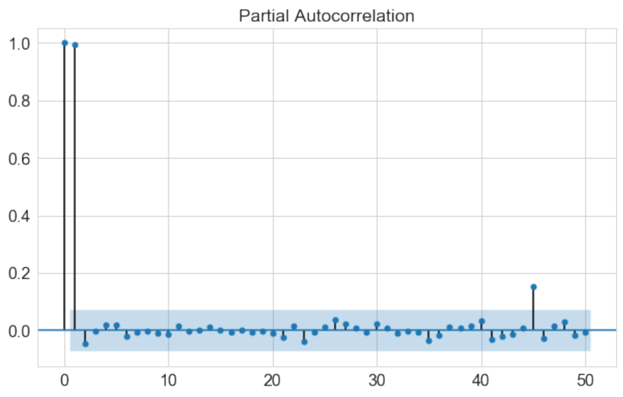

Figure 8: PACF Plot

up. Nevertheless, this model cannot be trusted to help make investment choices.

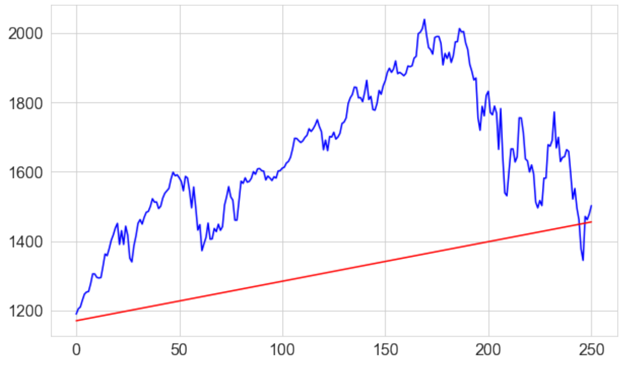

Figure 9: ARIMA result

3.4 LSTM

Long short-term memory (LSTM) is a promising architecture for this task, since there can be lags of unknown duration between important events in a time series. At the beginning I scale the data to a range between 0 and 1, since the stock prices otherwise would have a huge range, negatively impact the model performances. Next, I truncate the time series into a collection of windows of size $N + 1$, where N is the number of features the LSTM model will use, where each feature is just an observation of stock price of a past date. The last element of the window, either an observation from original time series, or a prediction from the model, would be the y-value that the model tries to predict.

In this project, I used 60 as the time steps for the model. In other words, the model will consider past 60 days of stock price and then predict the stock price of current day. Larger window size makes training process longer, while smaller window size might be insufficient for a complex model.

First LSTM model is a single layered LSTM model, where it simply accepts the truncated feature inputs without dropouts and uses $tanh$ for activation. The second LSTM model is a three-layer stacked LSTM model, with 20% dropout rate. Their perfomances can be found in figure 10 and 11, respectively.

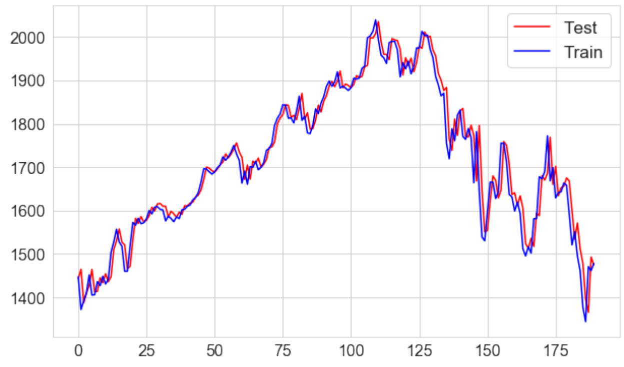

Figure 10: Single Layer LSTM Result

4 Results

Model	Training RMSE	Test RMSE
Simple Exp Smoothing	N/A	511.652
Double Exp Smoothing	N/A	625.248
Triple Exp Smoothing	N/A	407.301
ARIMA	N/A	367.205
Single Layer LSTM	12.509	39.484
Stacked LSTM	23.403	73.561

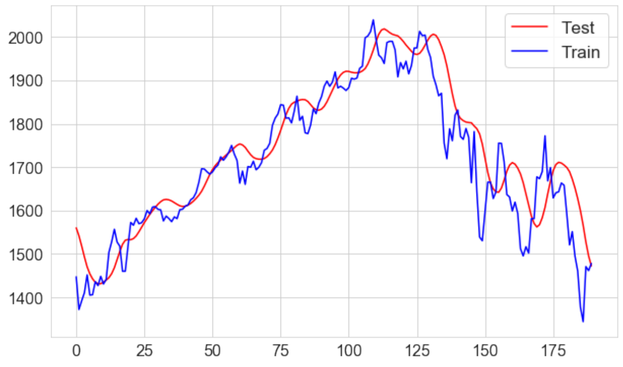

Figure 11: Stacked LSTM Result

5 Conclusion

Stock price prediction is a very hard task. All of the statistical models failed to yield any meaningful results. The single layer LSTM model, while almost perfectly predict the stock price, is highly unusual and I doubt it will perform as well in real market. The stacked LSTM model, on the other hand, is able to predict the general rise and fall of the stock prices, which is a better indication of a model's success.

References

[1] "Non-seasonal arima models." `https://otexts.com/fpp2/non-seasonal-arima.html`. Accessed: 2020-11-20.

[2] "Forecasting stock prices using exponential smoothing." `https://tinyurl.com/y285365r`. Accessed: 2020-11-20.