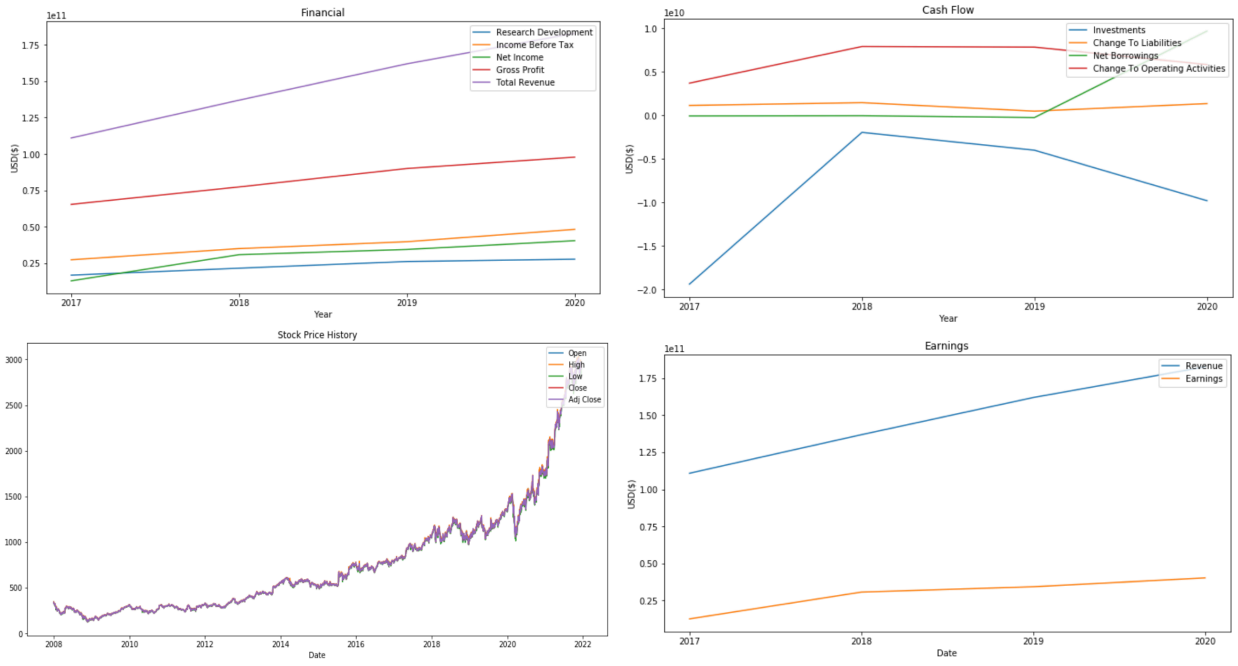
Stock Price Prediction by LSTM And ARIMA
Final Project Report
Group 33

Introduction

For this project, our group decided to use python to predict the stock price of Alphabet Inc. (NASDAQ: GOOGL). Alphabet Inc. is an American leading multinational technology company located in Mountain View, California, which is the parent company of Google. We will extract the dataset of the last six years' stock prices by using pandas and build models with ARIMA and LSTM to predict the stock price, then compare the results.

Data

The raw data is downloaded from Yahoo! Finance, from 2008-01-02 to 2021-12-14. The 10 years data (2008-2018), the historical stock prices, is for the purpose of machine learning models. The raw data downloaded with the cash flow, financial status, and earnings for the past four years (2017-2020). The purpose of these descriptive data are for the purpose of visualizing the income activities and the trends of Alphabet Inc. From the data we obtained, it could be concluded that Alphabet Inc has developed, invested, and increased income steadily in the past four years.



Plots of Data in Each Table

Model

LSTM: The Long Short Term Memory model is a class of Recurrent Neural Networks. The LSTM network consists of 4 gates such as the input, update, output and reset. Each one of

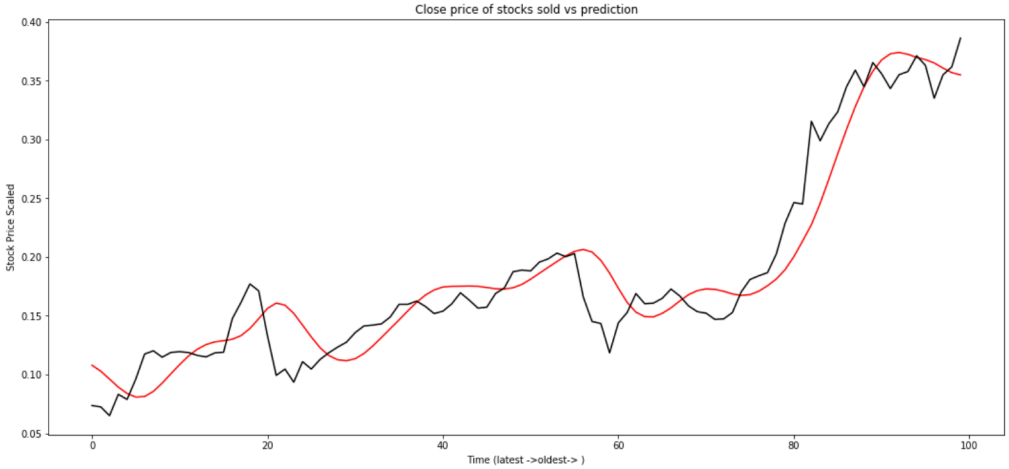
these gates consists of an activation function such as the tanh and sigmoid. The main role of the activation function is to make sure the values are within a range of [0,1] for tanh and [1,1] for sigmoid.

ARIMA: Arimathea Autoregressive Integrated Moving Average Model is a statistical model widely used for time series forecasting. This model is proficient for understanding and predicting time-series data. The model of ARIMA involves three parameters, lag observations (p), the order of the moving average (q), and the number of different observations (d). Unlike the regression model, which is directly aimed at modeling stationary data, the ARIMA model is modeling on non-stationary data with its own process to transform non-stationary data.

Analysis

LSTM:

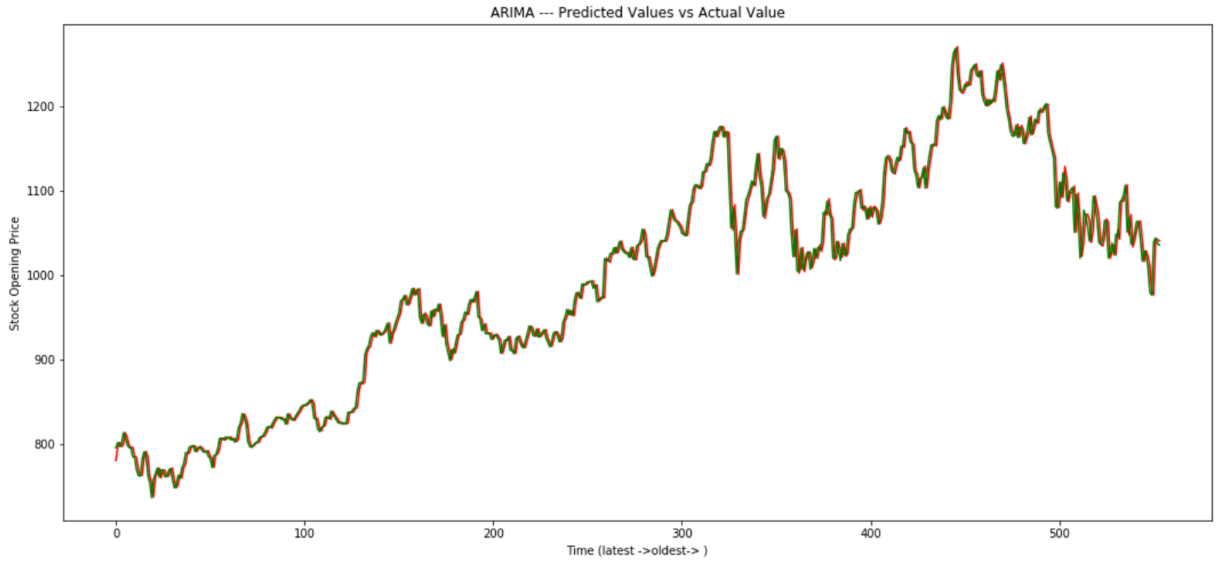
- After splitting, the data was scaled using MinMaxscaler and the training set was divided into X, Y
- The Dataset was divided into training and testing set. Various libraries such as Pandas, Keras, Dense, LSTM were used. The Dataset was divided on an 80/20 train test split with 'Open', 'Close', 'High' and 'Low' as the features and following visualizations were plotted.
- After splitting the data was scaled using a MinMaxscaler.Sequential, Dense and LSTM was imported and the model was optimized using the ADAM optimizer with MSE as the loss function.
- The model was fitted over the training set and tested upon the validation set.
- A Batch Size of 10 was used and the model was run over 100 epochs.
- The Predicted VS Actual value was plotted over the entire training set.
- The above steps were repeated over the test data.
- Overall, the LSTM gave an accuracy of 72%. This was over the closing price and depicts how close the predicted price is towards the actuals closing price. Since the model considers new features apart from the older constraints which it was trained on this accuracy was expected as in the recent years many factors have changed due to the pandemic.



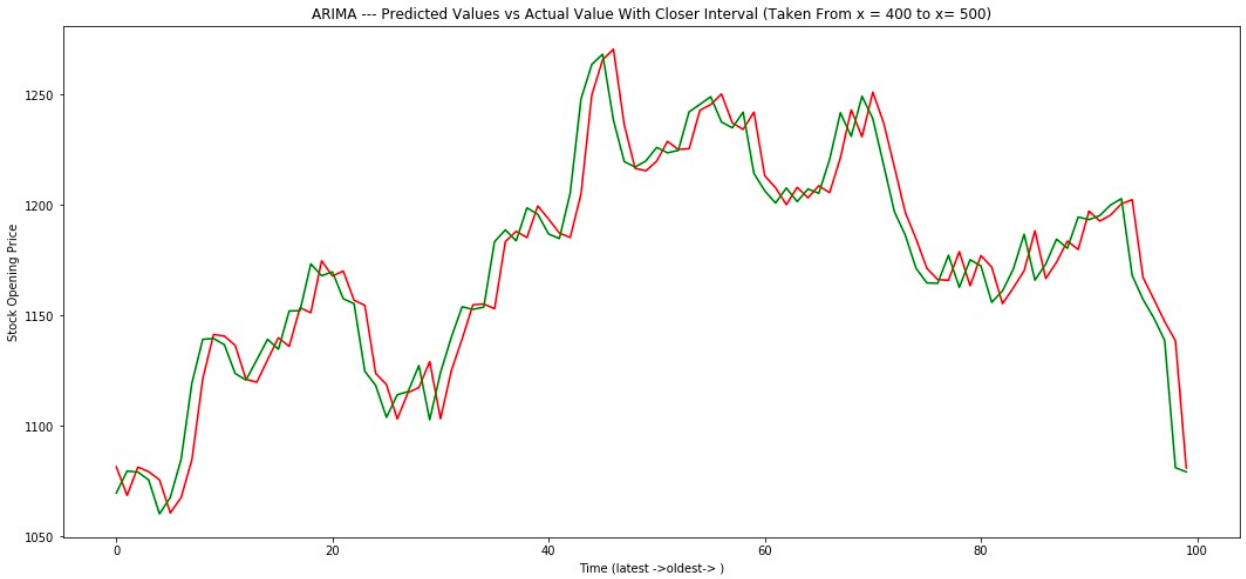
LSTM — Actual Vs Predicted values (closing price)

ARIMA**:**

- The Dataset was divided into training and testing set. Various libraries such as Pandas, Keras, Dense, ARIMA were used. The Dataset was divided on an 70/30 train test split with ‘Open’, ‘Close’, ‘High’ and ‘Low’ as the features and following visualizations were plotted
- After Splitting, the training set was divided into X, Y train and a Validation set to test the accuracy.
- Then the average is calculated by dividing the accuracy by the total number of true and predicted values
- Further we create an ARIMA model by passing the p, d and q values as parameters
- It’s based on the lag observations and uses moving averages. For example, it takes the data from a period say 1-60th day and predicts for 61st and follows the same for the entire data set
- We can see that ARIMA uses past constraints and applies it to the test also and since it removes any scope for future variations it gives a good accuracy.
- The mean absolute percentage error of 15.89% was observed. The model accuracy of 84.11% was concluded.



ARIMA — Predicted Values vs Actual Value



ARIMA — Predicted Values vs Actual Value with Closer Interval

Conclusion

To sum up, we extracted the data from Yahoo Finance by implementing the Pandas, and generated 5 tables in our database. Then analyzed the database and predicted the stock price by using LSTM model and Arima model. For LSTM model we obtained the accuracy of 72%, and 84% accuracy for ARIMA model. In conclusion, therefore, the ARIMA model is better than the LSTM model in our case, since the ARIMA has a higher accuracy.

Future Research

For future research, we will try other methods for predicting the stock prices, and hopefully there will be a better model with higher accuracy. We could implement this model for predicting stock price in real time and for more company. Therefore, stock owners could have a preview of what is the trends of their stocks, and plan their actions base on the prediction.

References:

- https://finance.yahoo.com/quote/GOOG/
- https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/
- https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
- https://stackabuse.com/time-series-analysis-with-lstm-using-pythons-keras-library/
- https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
- https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/
- https://www.analyticsvidhya.com/blog/2020/10/how-to-create-an-arima-model-for-time-series-forecasting-in-python/
- https://towardsdatascience.com/machine-learning-part-19-time-series-and-autoregressive-integrated-moving-average-model-arima-c1005347b0d7
- https://en.wikipedia.org/wiki/Mean_absolute_percentage_error
- https://github.com/nsimakov/eas503