PROJECT REPORT ON

"Stocks Price Prediction using LSTM and ARIMA Model"

Paper Title:

Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models

Paper link/QR Code:

https://arxiv.org/ftp/arxiv/papers/2009/2009.10819.pdf

Submitted by:

Charanveer Singh

SJSU ID: 011826308

Under the Guidance of

Prof. Shilpa Gupta

ISE 244

16th May 2022

Problem Definition

The purpose of this project is an attempt to accurately anticipate the stock values of NIFTY-50 stock from the National Stock Exchange (NSE) of INDIA. This paper discussed the deep learning model called the Long-short term memory (LSTM) model and Auto Regressive Integrated Moving Average (ARIMA) model which were used to perform the research. The performance of these models was compared using different metrics such as R squared values, Mean absolute error, Root mean square error, etc. LSTM, an artificial recurrent neural network architecture used in deep learning, comprises models that are extensively effective for sequence connection problems because they can preserve useful past information while disregarding irrelevant data. Because the stock market is fluctuating, it is vital to research this topic in order to make accurate predictions.

Project Objectives

LSTM Model:

A recurrent Neural Network (RNN) is a special kind of Neural network that stores the previous output temporarily to work on current input data. However, this model has some disadvantages such as

- Incapable to store information for a longer period of time
- Incapable to keep useful information, eliminate useless information
- Vanishing gradients during the training process

As a result, the concept of LSTM was introduced.

Long Short Term Memory networks are a special type of RNN developed to deal with cases where RNN fails. They both have chain-like structures however, there is an additional hidden layer in the LSTM model.

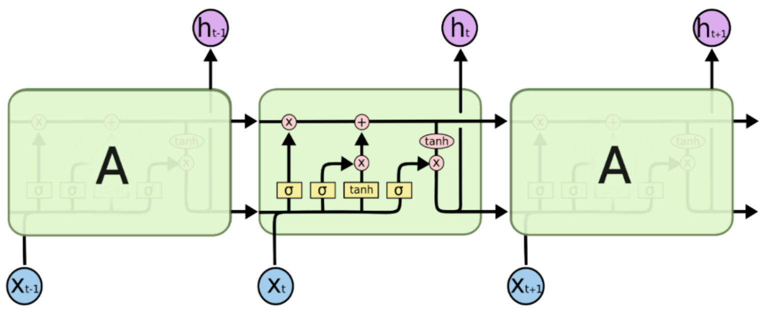


Figure1: LSTM Model structure

Stepwise working of LSTM model:

Step 1: In this step, the first sigmoid layer named, the “Forget gate layer” decides which information is discarded

Step 2: In this step, the second sigmoid layer named, the "Input gate layer" decided the value which needs to be updated
Step 3: In this step, the third tanh layer controls the flow of data through the network
Step 4: Finally, the state is updated using the Step 2 and 3

ARIMA Model:

Auto Regressive Integrated Moving Average model also known as the ARIMA model uses a technique to predict future values based on information from the past. In ARIMA:

- **AR** stands for Auto regressive. This part of the model is responsible for generating continuous output values by multiplying input with some weights. It's termed Auto Regressive since the inputs are prior time-series values called lags.

- **I** stand for Integrated. This part of the model is intended to reduce trends (increasing/decreasing) within the dataset.

- **MA** stands for Moving Average. This part of the model works just like AR; however, instead of considering previous values we now consider previous errors called error lags.

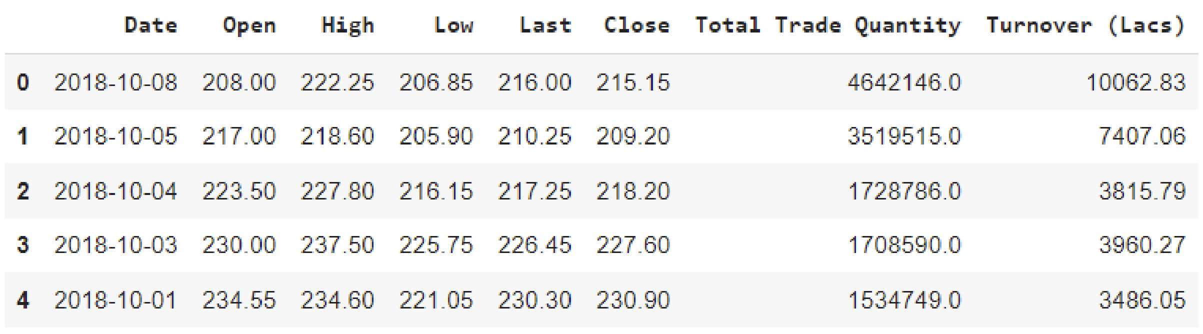
There are three terms that are used to define the ARIMA model and these are:

- *P*: which means the number of lags or autoregressive terms
- *D:* which means the number of differences required to make model stationery
- *Q:* which means the number of error lags or moving average terms

Analysis

The project is split into three main segments, the use of LSTM and ARIMA models on the NIFTY-50 dataset. Furthermore, the best-performing model was tested on the Tesla dataset. Let's begin by understanding the two datasets and their attributes.

NIFTY-50 dataset has a shape of (1235, 8) and a sample of the dataset is shown below:



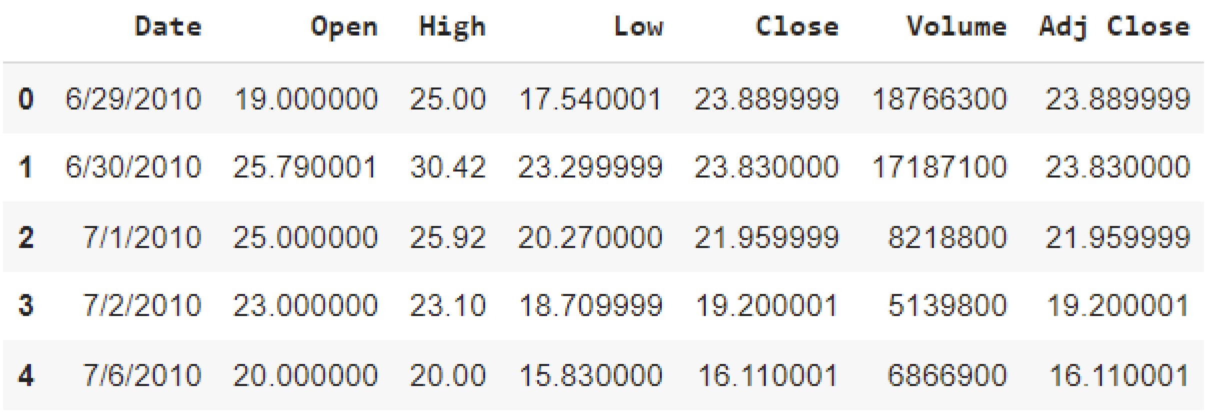
	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-10-08	208.00	222.25	206.85	216.00	215.15	4642146.0	10062.83
1	2018-10-05	217.00	218.60	205.90	210.25	209.20	3519515.0	7407.06
2	2018-10-04	223.50	227.80	216.15	217.25	218.20	1728786.0	3815.79
3	2018-10-03	230.00	237.50	225.75	226.45	227.60	1708590.0	3960.27
4	2018-10-01	234.55	234.60	221.05	230.30	230.90	1534749.0	3486.05

Figure 2: NIFTY-50 dataset sample

Attributes definition:

Date: Date when the stock price was observed
Open: Starting stock price for the day
High: Highest stock price for the day
Low: Lowest stock price for the day
Last: Last Stock price noted for a day
Close: Stock closing price for the day
Total Trade Quantity: The number of times stock has been traded
Turnover: The number of shares exchanged times the price of each share

Tesla dataset has a shape of (1692, 7) and a sample of the dataset is shown below:



	Date	Open	High	Low	Close	Volume	Adj Close
0	6/29/2010	19.000000	25.00	17.540001	23.889999	18766300	23.889999
1	6/30/2010	25.790001	30.42	23.299999	23.830000	17187100	23.830000
2	7/1/2010	25.000000	25.92	20.270000	21.959999	8218800	21.959999
3	7/2/2010	23.000000	23.10	18.709999	19.200001	5139800	19.200001
4	7/6/2010	20.000000	20.00	15.830000	16.110001	6866900	16.110001

Figure 3: Tesla dataset sample

Attributes definition is the same except now we have “Volume” and “Adj Close”:

Volume: The number of times stock has been traded
Adj Close: Adjusted closing price for the day

The variable observed in both datasets was "Close," and the attribute trajectory for the datasets is as follows:

a) NIFTY-50 dataset:

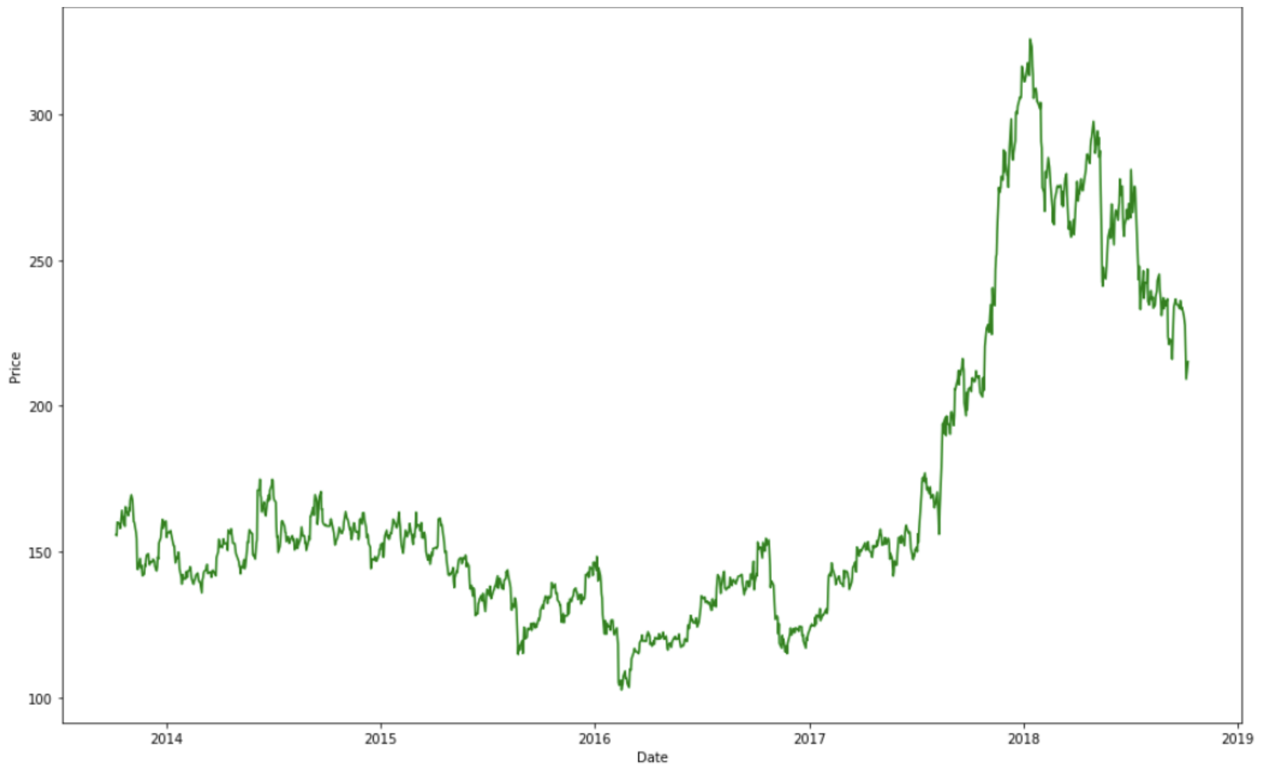


Figure 4: Close variable trend in the NIFTY-50 dataset

b) Tesla dataset:

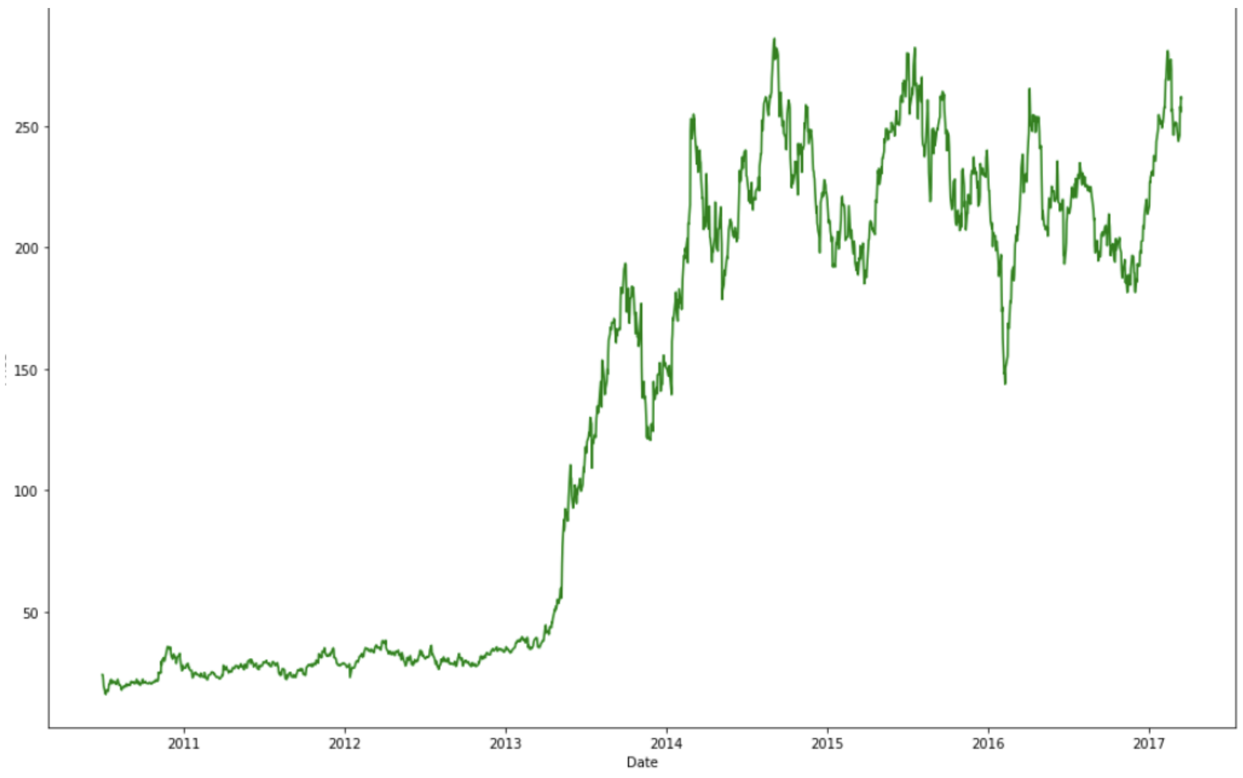
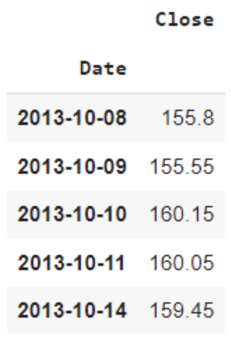


Figure 5: Close variable trend in Tesla dataset

To be fair, both models and datasets went through the same data preprocessing procedure, which consists of three primary stages:

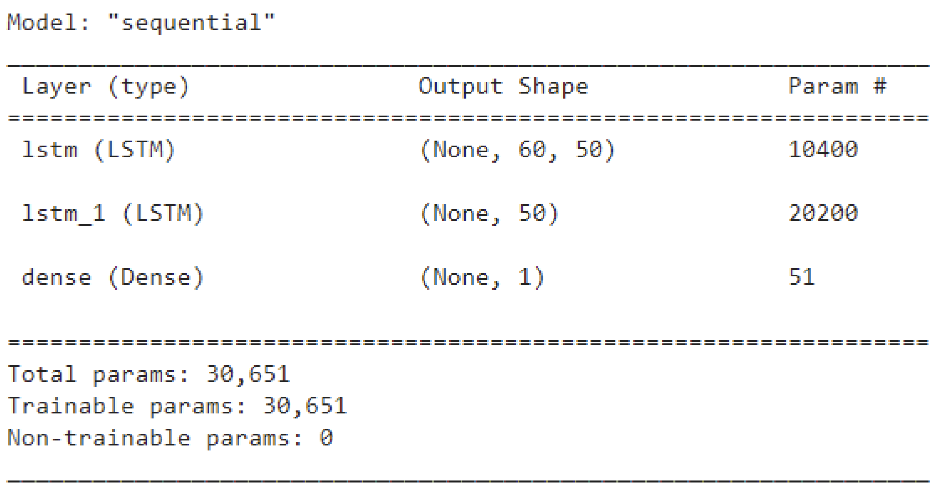
a) Set “Date” as an index variable (*shown in figure 6.*)
b) Splitting the dataset into standard 80-20 ratio
c) Scaling the data using normalization



Date	Close
2013-10-08	155.8
2013-10-09	155.55
2013-10-10	160.15
2013-10-11	160.05
2013-10-14	159.45

Figure 6: Using “Date” as an index variable

<u>LSTM model structure:</u>

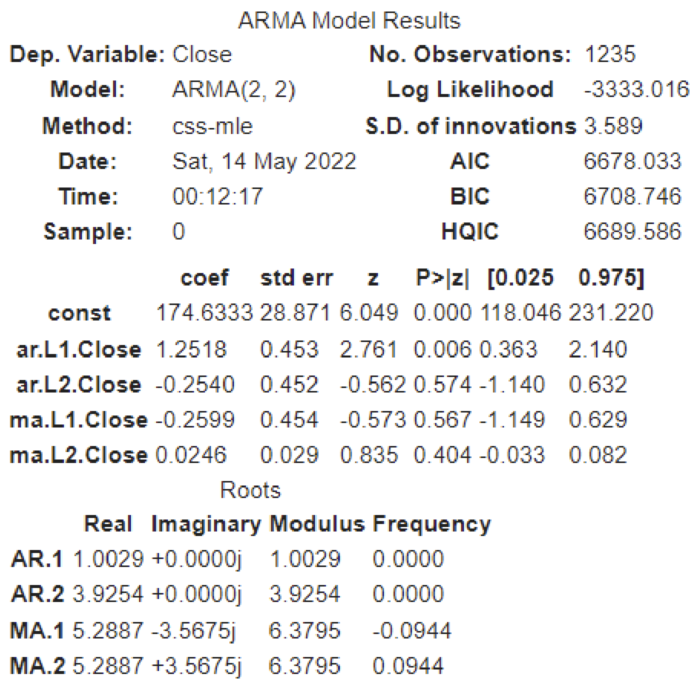


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 50)            10400

 lstm_1 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 30,651
Trainable params: 30,651
Non-trainable params: 0
_________________________________________________________________
```

Figure 7: LSTM structure

To develop our model, we only used two LSTM layers with 50 neurons and one output dense layer.

<u>ARIMA Model structure:</u>



ARMA Model Results

Dep. Variable:	Close	**No. Observations:**	1235
Model:	ARMA(2, 2)	**Log Likelihood**	-3333.016
Method:	css-mle	**S.D. of innovations**	3.589
Date:	Sat, 14 May 2022	**AIC**	6678.033
Time:	00:12:17	**BIC**	6708.746
Sample:	0	**HQIC**	6689.586

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	174.6333	28.871	6.049	0.000	118.046	231.220
ar.L1.Close	1.2518	0.453	2.761	0.006	0.363	2.140
ar.L2.Close	-0.2540	0.452	-0.562	0.574	-1.140	0.632
ma.L1.Close	-0.2599	0.454	-0.573	0.567	-1.149	0.629
ma.L2.Close	0.0246	0.029	0.835	0.404	-0.033	0.082

Roots

	Real	**Imaginary**	**Modulus**	**Frequency**
AR.1	1.0029	+0.0000j	1.0029	0.0000
AR.2	3.9254	+0.0000j	3.9254	0.0000
MA.1	5.2887	-3.5675j	6.3795	-0.0944
MA.2	5.2887	+3.5675j	6.3795	0.0944

Figure 8: ARIMA structure

This model was designed using p, d, and q values as (2, 0, 2). Examining the coefficients is one way to determine the appropriate number of lags and error lags required in the model. We can use these values because our coefficient value is very close to 0. Furthermore, looking at our P value, the coefficient is statistically significant.

Results

As we know the structure of both the models we'll be using and the datasets let's now look at the outcomes.

<u>LSTM Model Predictions for NIFTY-50 dataset:</u>

You might be asking why there are only two LSTM layers and one output layer with so few neurons. Because of the model’s prediction as shown below.

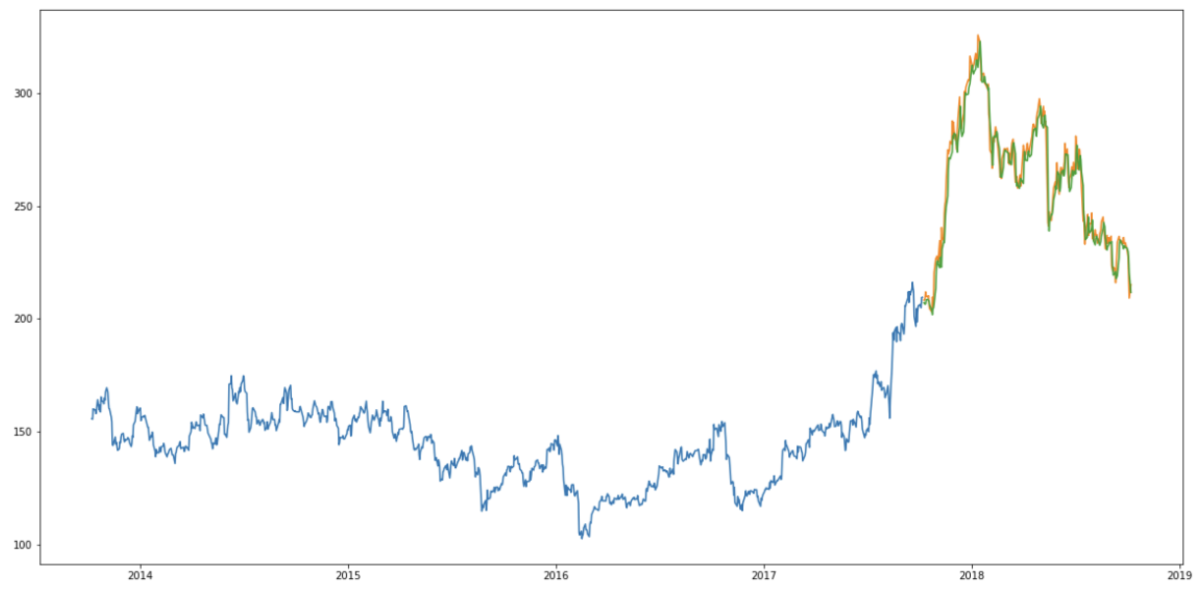


Figure 9: LSTM model predictions on the NIFTY-50 dataset

Furthermore, shown below are the RMSE and MAE values of the model:

```
RMSE Score for LSTM Model: 6.46755
MAE Score for LSTM Model: 4.86768
```

The outcomes were as expected, and they met the paper's predictions. This is why I decided against adding further layers or increasing the number of neurons in the model. There felt no need for such a small number of error scores. (However, it did reduce these error scores even further upon trying)

<u>ARIMA Model Predictions for NIFTY-50 dataset:</u>

These results surprised me and were not at all what I had expected.

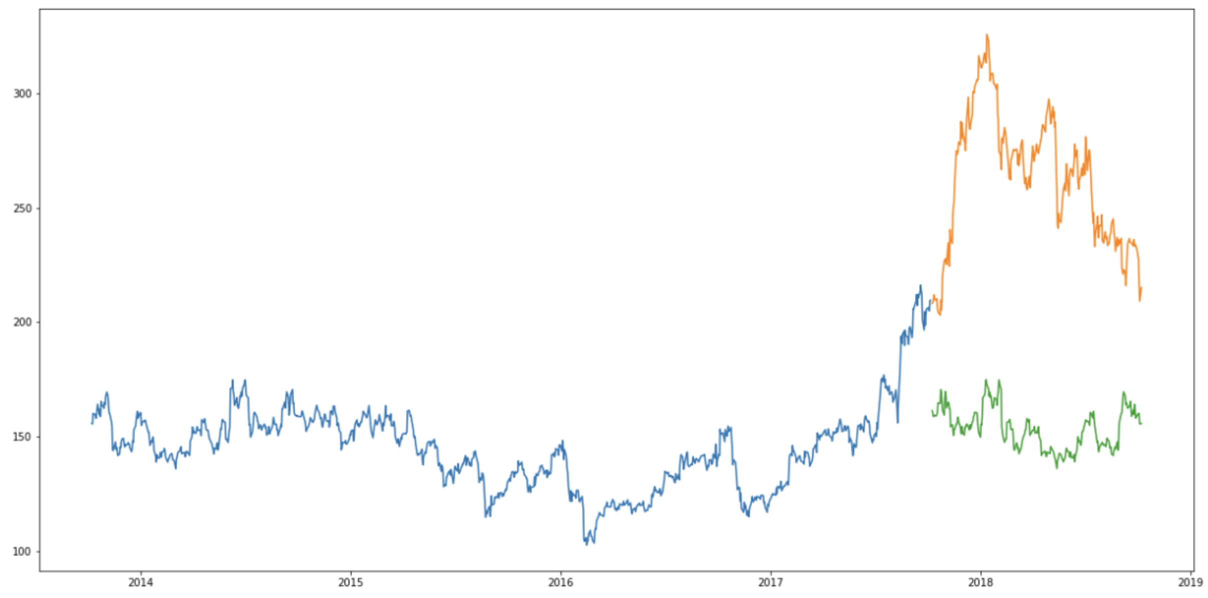


Figure 10: ARIMA model predictions on the NIFTY-50 dataset

The RMSE and MAE values of the model were:

```
RMSE Score for ARIMA Model: 113.13833
MAE Score for ARIMA Model: 109.04663
```

We chose to use the LSTM on another dataset (Tesla) because it completely surpassed the ARIMA model in making predictions. Below are the results:

LSTM Model Predictions for Tesla dataset:

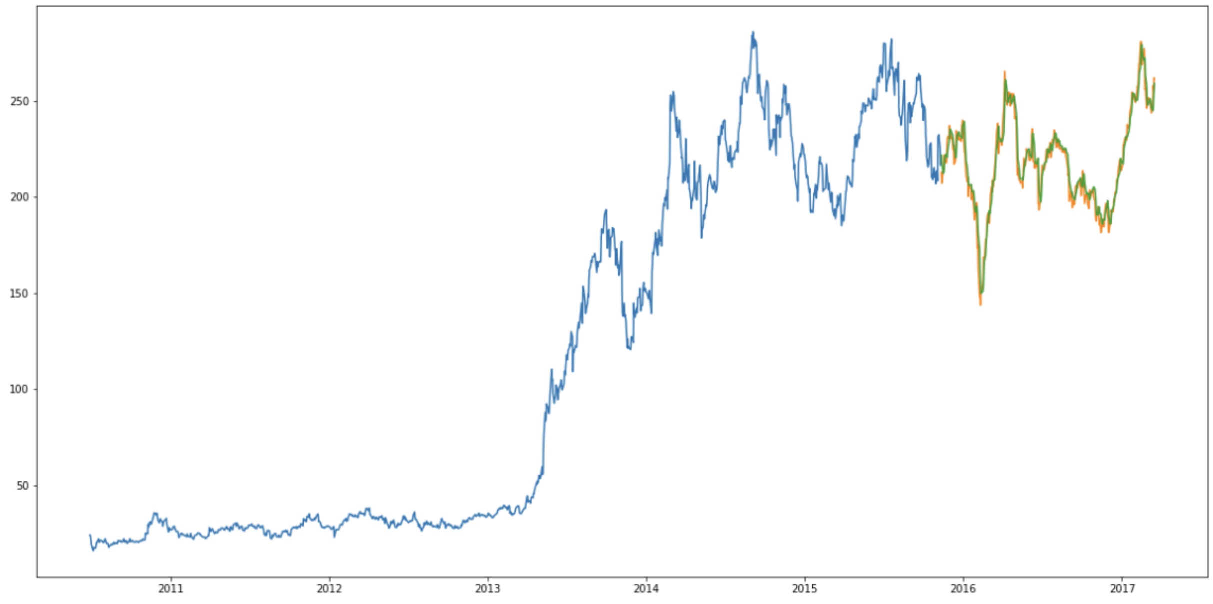
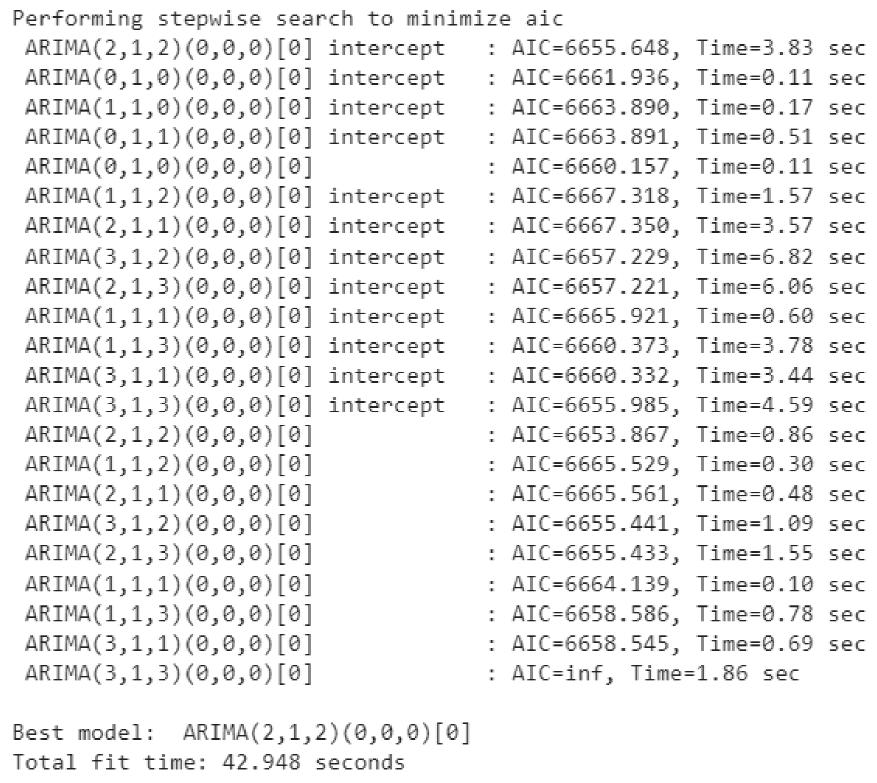


Figure 11: LSTM model predictions on the Tesla dataset

```
RMSE Score for LSTM Model: 6.24399
MAE Score for LSTM Model: 4.68073
```

Discussion

The LSTM model was undoubtedly the best performing, completely outperforming the ARIMA model in terms of predicting the stock price. At first, I concluded it had to do with the *P*, *D*, and *Q* variables we used to create the model, but this was not the case. That was validated by the "auto_arima" function, which identifies the ARIMA model's optimal order automatically by trying different orders. The following were the "auto_arima" results:



```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=6655.648, Time=3.83 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=6661.936, Time=0.11 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=6663.890, Time=0.17 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=6663.891, Time=0.51 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=6660.157, Time=0.11 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=6667.318, Time=1.57 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=6667.350, Time=3.57 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=6657.229, Time=6.82 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=6657.221, Time=6.06 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=6665.921, Time=0.60 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=6660.373, Time=3.78 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=6660.332, Time=3.44 sec
 ARIMA(3,1,3)(0,0,0)[0] intercept   : AIC=6655.985, Time=4.59 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=6653.867, Time=0.86 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=6665.529, Time=0.30 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=6665.561, Time=0.48 sec
 ARIMA(3,1,2)(0,0,0)[0]             : AIC=6655.441, Time=1.09 sec
 ARIMA(2,1,3)(0,0,0)[0]             : AIC=6655.433, Time=1.55 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=6664.139, Time=0.10 sec
 ARIMA(1,1,3)(0,0,0)[0]             : AIC=6658.586, Time=0.78 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=6658.545, Time=0.69 sec
 ARIMA(3,1,3)(0,0,0)[0]             : AIC=inf, Time=1.86 sec

Best model:  ARIMA(2,1,2)(0,0,0)[0]
Total fit time: 42.948 seconds
```

Figure 12: "auto_arima" Model

The "auto_arima" model's sole objective is to lower the AIC score by determining the best optimal order. In our case, the optimal model was (2,1,2). However, we built our model using the optimal order of (2, 0, 2). As a result, I tested the new best sequence, but it didn't substantially raise our accuracy score. Therefore, In terms of stock price forecasts, the LSTM model came out on top.

Evaluation and Reflection

Following were the project's main takeaways:

- The LSTM model is clearly superior to the ARIMA model.
- The LSTM prediction accuracy was almost 20 times better than ARIMA
- In the LSTM model, there is no effect on prediction accuracy after epoch = 25
- Different ARIMA model optimal ordering didn't seem to make a significant difference in accuracy predictions

Key distinctions between the two models:

LSTM Model	*ARIMA Model*
● There is no prerequisite (standardization or level shifts) ● requires a large amount of data ● Can model non-linear function with neural networks ● Slow depending on epochs	● no parameter tuning, easy to construct ● Can easily handle multivariate data ● Quick to run

References

[1] *Understanding LSTM Networks*. Understanding LSTM Networks -- colah's blog. (n.d.). Retrieved May 04, 2022, from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[2] *Stock price prediction - machine learning project in python*. DataFlair. (2021, August 25). Retrieved May 06, 2022, from https://data-flair.training/blogs/stock-price-prediction-machine-learning-project-in-python/

[3] *Stock price prediction using machine learning and LSTM-based deep learning models.* Papers With Code. (n.d.). Retrieved May 09, 2022, from https://paperswithcode.com/paper/stock-price-prediction-using-machine-learning

[4] By: IBM Cloud Education. (n.d.). *What are recurrent neural networks?* IBM. Retrieved May 09, 2022, from https://www.ibm.com/cloud/learn/recurrent-neural-networks

[5] Miguel, T. (2021, January 28). *How the LSTM improves the RNN. Medium*. Retrieved May 14, 2022, from https://towardsdatascience.com/how-the-lstm-improves-the-rnn-1ef156b75121

[6] Samaha, B. (2020, September 4). *Build an Arima model to predict a stock's price*. Medium. Retrieved May 14, 2022, from https://levelup.gitconnected.com/build-an-arima-model-to-predict-a-stocks-price-c9e1e49367d3