Forecasting Market Movement in Dhaka Stock Exchange: LSTM vs. ARIMA

By

Pranjal Chakraborty

18175001

A thesis submitted to the Department of Economics and Social Sciences in partial fulfillment of the requirements for the degree of M.Sc. in Applied Economics

Department of Economics and Social Sciences,

Brac University

April 2019

Declaration

It is hereby declared that

1. The thesis submitted is my own original work while completing degree at Brac University.
2. The thesis does not contain material published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

_____

Pranjal Chakraborty

18175001

Approval

The thesis/project titled “Forecasting Market Movement in Dhaka Stock Exchange: LSTM vs. ARIMA” submitted by

1. Pranjal Chakraborty (18175001)

of Spring, 2019 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of M.Sc. in Applied Economics on 28 March, 2019.

Examining Committee:

Supervisor:
(Member)

_____
Dr. Wasiqur Rahman Khan
Professor,
Department of Economics and Social Sciences
Brac University

Program Coordinator:
(Member)

_____
Dr. ATM Nurul Amin
Professor and Chairperson,
Department of Economics and Social Sciences
Brac University

Departmental Head:
(Chair)

_____
Dr. ATM Nurul Amin
Professor and Chairperson,
Department of Economics and Social Sciences
Brac University

Abstract

This paper creates and examines the performance of LSTM models against ARIMA models, using Dhaka Stock Exchange data from beginning of 2000 to the beginning of 2019. An algorithm has been designed to simultaneously train and test the models using datasets of 340 companies of the market. The empirical result shows the absolute dominance of ARIMA over LSTM, which contradicts some previous works. At the end, we try to discuss the possible reasons behind the unsatisfactory performance of LSTM and explore some of the possible future expansions and extensions of the current work.

Keywords: Long Short-Term Memory, Autoregressive Integrated Moving Average, Econometrics, Data Mining, Deep Learning, Neural Network, Machine Learning, Artificial Intelligence, Capital Market, RMSE, Dhaka Stock Exchange

Acknowledgement

I express my utmost gratitude to my honorable supervisor, Dr. Wasiqur Rahman Khan. His econometric insights ultimately shaped my thesis from being a mere programming project to a post-graduate research work. It has been a privilege to work with him.

I am also grateful and thankful to my team lead of my office, Manuel Rony Gomes, Senior Software Engineer of Therap Services LLC. From the beginning of my post-graduate study, he helped me from his side to effortlessly continue my course works.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ARIMA	Auto-Regressive Integrated Moving Average
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
ANN	Artificial Neural Network
AI	Artificial Intelligence
RMSE	Root Mean Square Error
DSE	Dhaka Stock Exchange
EMH	Efficient Market Hypothesis
HTTP	HyperText Transfer Protocol

Chapter 1

Introduction

In this era of Artificial Intelligence, wherever we see, we find applications of Machine Learning and Deep Learning. Our smartphones are getting more powerful with limited resources, self driving cars are already on the streets, and not surprisingly, multinational investment banks like JPMorgan Chase and Morgan Stanley now have their own Machine Learning department to assist them to make investment decisions using AI.

Capital markets are very popular investment choices worldwide and forecasting capital markets has been an active research area for a long time. According to Efficient Market Hypothesis (EMH), capital markets follow a random walk pattern, which makes it harder to forecast. Though this hypothesis is widely accepted by the research community as a central paradigm governing the markets in general, researchers have attempted to extract patterns in the way stock markets behave and respond to external stimuli.

In Bangladesh, there are two stock exchanges in operation, one of which is Dhaka Stock Exchange (DSE). DSE began trading in 1976, with a paid-up capital of BDT 0.138 billion. At that time, it had market capitalization of BDT 0.147 billion. Correctly forecasting movement for the companies of this stock exchange can be very beneficial for the investors, as it has market capitalization of BDT 4,228.95 billion (closing price) in 2017. DSE has 581 listed companies, including banks, financial institutions, treasury bonds, textiles, pharmaceuticals, insurances and mutual funds. If treasury bonds (221 in number) are excluded, there are 360 companies.

In this paper, we will forecast the stock price movement for these 360 companies by implementing models of LSTM (Long Short-Term Memory), an Artificial Recurrent Neural Network architecture, and analyze its performance against respective ARIMA (Auto-Regressive Integrated Moving Average) models, which is the most widely used (Hyndman and Athanasopoulos, 2018) econometric tool for capital market movement analysis and forecasting. LSTM has performed very good in forecasting several capital markets, but has not yet been put to test for Bangladesh. ARIMA (the Box-Jenkin's approach) is widely used across financial institutions. We will develop an algorithm, which will be equipped to simultaneously create, split, train, forecast and analyze performance of LSTM models and ARIMA models for these 360 companies.

Chapter 2

Literature Review

Research shows that, the more developed the country is, the more their stock markets show random walk attributes (Solnik, 1973). Although, there is no definitive findings on the market efficiency of the developing countries. Some thinks developing markets have weak form of efficiency (Ojah & Karemara, 1999; Branes, 1986), others think, developing markets do not show random walk features (Khababa, 1998; Harvey, 1994). It indicates that, market is not efficient in its weak state. It is possible to predict better in these types of markets.

Irfan et al (2010) analyzed Karachi Stock Exchange of Pakistan, stating it as an emerging market, to investigate weak form efficiency. They used 10 years of data and used root test, autocorrelation tests and ARIMA model. They found that, the market does not follow the random walk model and it is not efficient in weak form.

There are works with ARIMA on Dhaka Stock Exchange, although not company specific (Rahman & Hossain, 2006). They found ARIMA(3,0,1) and ARIMA(1,0,1) models were best fit. They used these models for a validation period and compared it with actual values. Similar works were done by Mollick and Bepari (2008). They found ARIMA(1,0,2) as best fit.

Jia (2016) investigated the effectiveness of LSTM for stock market prediction. He found the returns data was resistant to overfitting and generally works better with more layers and more cells. He found LSTM to be effective to find pattern in the stock market. There have been some extensive researches on forecasting capital market movement, incorporating deep learning techniques, but based on twitter activities. Mittal and Goel (2013) classified tweets into four classes- ‘Happy’,

‘Calm’, ‘Alert’ and ‘Kind’. These sentimental values are correlated with Dow Jones Industrial Average Data using LSTM. Bao et al (2017) proposed a model consisting LSTM, WT (wavelet transforms) and SAEs (stacked autoencoders). They emphasized mostly on SAEs, which minimized error between input and output data between layers, which resulted in learning invariant and abstract features. Persio and Honchar (2016) used CNN and RNN on S&P500 time series data. Roondiwala et al (2015) used 5 years of historical data of NIFTY 50 for market prediction using LSTM.

Namin and Namin (2018) worked with several major capital market data (e.g. N225, DJIA, GSPC), from 1985 to 2018, to compare between ARIMA and LSTM. In his work, LSTM performed better in every case. LSTM improved the prediction by 85% on average. Adebiyi et al (2014) worked with stock data of Dell Incorporated and analyzed forecasting performance of ARIMA and Artificial Neural Network (ANN). They found both performed similar in their work.

Chapter 3

Methodology

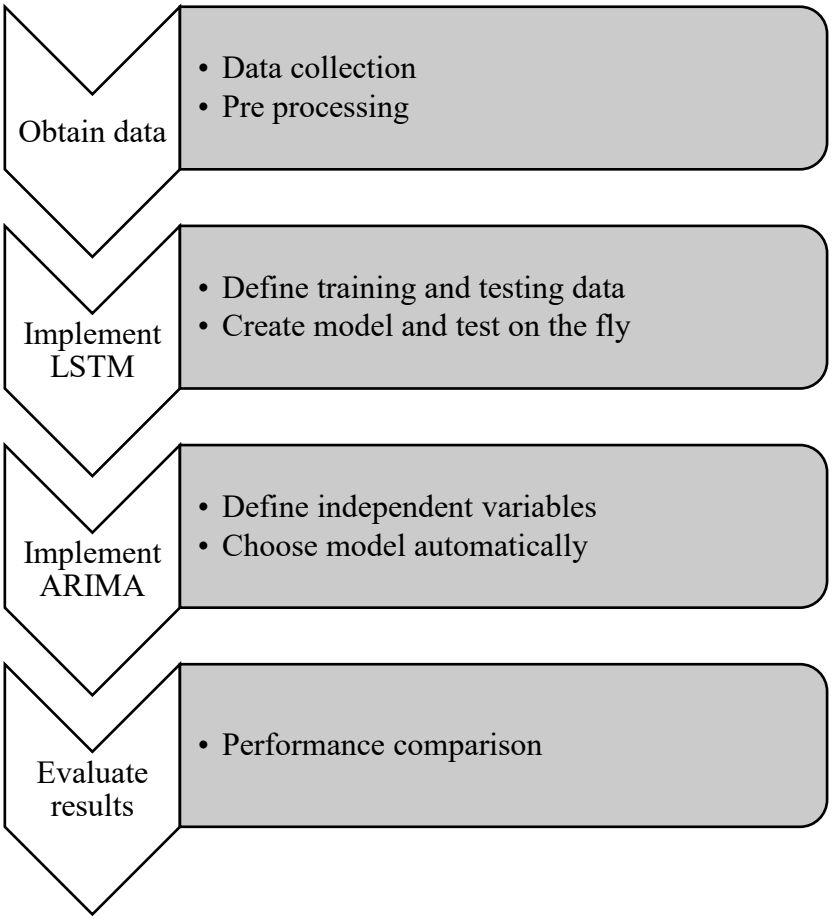

Figure 1. Work flow of the analysis

The above procedure will be applied to all companies separately and parallelly. Now we will discuss about the tools that we are using here, ARIMA and LSTM for forecasting and RMSE (Root Mean Square Error) for error measurement.

3.1 ARIMA

The ARIMA (Autoregressive Integrated Moving Average) model is a generalized version of the ARMA (Autoregressive Moving Average) model. Both of them serves the purpose of better understanding the data or forecasting by being fitted to time series data. The model contains of three parts-

AR = The evolving independent variable is regressed against its own lagged values.

I = The data is "integrated", meaning, the data values have been replaced (once or more) with the difference between their values and the previous values.

MA = The lagged observations are applied with the dependency between the observation and the residual error from a moving average model.

The non-seasonal ARIMA model, that we will be using in our work, is generally denoted by ARIMA (p,d,q) [$p,d,q \geq 0$], where,

p = number of autoregressive terms, which is associated with AR

d = number of nonseasonal differences needed for stationarity, which is associated with I

q = number of lagged forecast errors in the forecasting equation, which is associated with MA

In case of two of the three parameters being zero, they can be referred to based on the non-zero parameter. In that way, ARIMA (1,0,0) is AR (1), ARIMA (0,1,0) is I (1) and ARIMA (0,0,1) is MA (1).

Let,

$X_t =$ The time series data (real numbers). t here, is the time index.

The $ARMA(p', q)$ model would be,

$$X_t - \alpha X_{t-1} - \cdots - \alpha_{p'} X_{t-p'} = \varepsilon_t + \theta_t \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$$

Which can be re-written into,

$$\left(1 - \sum_{i=1}^{p'} \alpha_i L^i\right) X_t = \left(1 - \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$$

Here,

L = lag operator, α_i = parameters of the autoregressive part, θ_i = parameters of the moving average part and ε_t = error terms.

Assuming, the polynomial $(1 - \sum_{i=1}^{p'} \alpha_i L^i)$ has a unit root (a factor $(1 - L)$) of multiplicity d, then,

$$\left(1 - \sum_{i=1}^{p'} \alpha_i L^i\right) = \left(1 - \sum_{i=1}^{p'-d} \phi_i L^i\right) (1 - L)^d$$

This polynomial factorization property is expressed by an ARIMA (p,d,q) with $p = p' - d$, and so,

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) (1 - L)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$$

The above equation can be generalized as,

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) (1 - L)^d \, X_t = \, \delta + \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$$

It defines an ARIMA (p,d,q), along with drift $\frac{\delta}{1 - \sum \phi_i}$.

Order determination: Akaike Information Criterion (AIC) is a useful criterion to determine the order of nonseasonal ARIMA model. AIC is written as,

$$AIC = \, -2 \log(L) + 2(p + q + k)$$

where, L = likelihood of the data, p = order of the autoregressive part, q = order of the moving average part and k = intercept of the model.

Corrected AIC for ARIMA model is written as,

$$AIC_c = AIC + \frac{2\,(p + q + k)(p + q + k + 1)}{(T - p - q - k - 1)}$$

The Bayesian Information Criterion would be,

$$BIC = AIC + (\log(T) - 2)(p + q + k)$$

We would like to minimize the AIC, $\mathrm{AIC_c}$ or BIC values to find a better model. As we will be creating ARIMA models programmatically, we will be using these criterions to choose the best ARIMA model on a particular dataset to perform a particular forecasting.

To forecast, an ARIMA model can be observed as a cascade of two models:

Non-stationary: $Y_t = (1 - L)^d \, X_t$

Wide-sense stationary: $\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) Y_t = \, \left(1 - \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$

Now, for the process Y_t , we can generalize the method of autoregressive forecasting and then forecast.

For our task, we will be using pyramid-arima, an auto-ARIMA implementation, packaged as a python library.

3.2 LSTM

LSTM (Long Short-Term Memory) networks are a variant of Recurrent Neural Networks (RNN). LSTM is capable of learning long-term dependencies. It is so powerful that, it can process single data points to sequences of data, using its feedback connection structure, which makes it a "general purpose computer". Its capabilities range from computer vision, speech recognition to time series analysis.

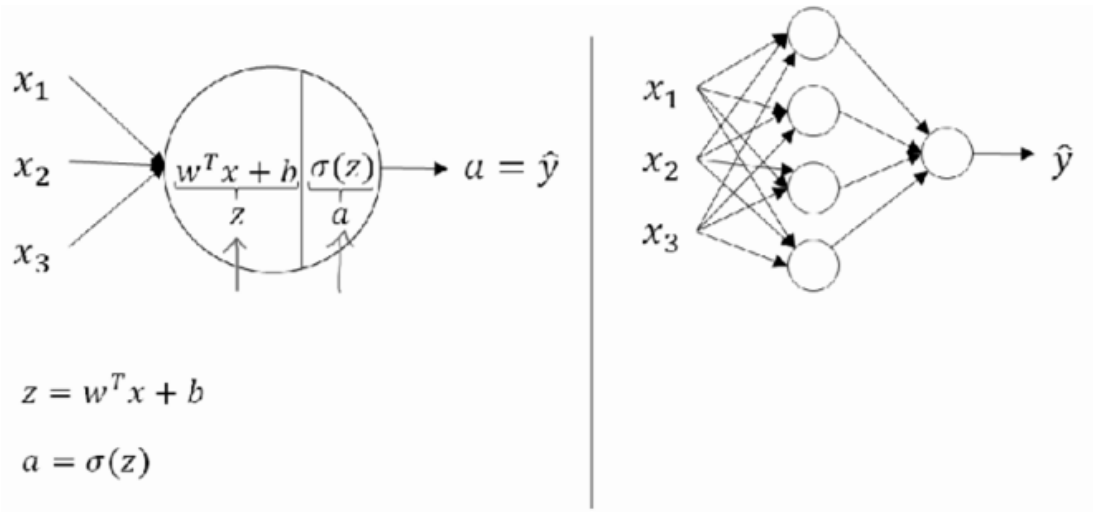

Figure 2. Fundamental structure of Neural Network. by A. Ng, 2017,

https://www.coursera.org/specializations/deep-learning

Fundamentally, in each basic neural network cell, there is 2-steps of computation.

$$z = W^T x + b$$

$$propagated\ output, a = activation\_function(z)$$

Here, W is the coefficient matrix, which is shared across layers, and b is the intercept matrix. T denotes the layer. X is the input, or, the previous layer output. The activation function can be sigmoid, tanh, ReLU, or leaky ReLU.

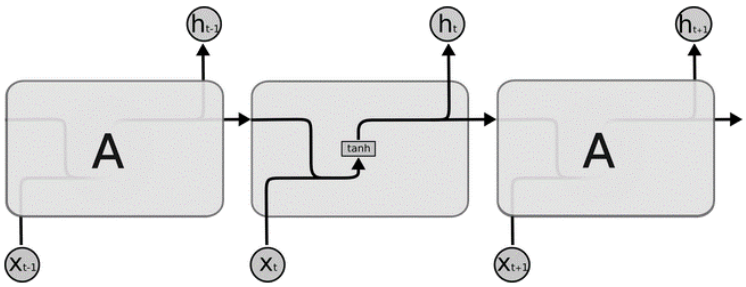

Figure 3. LSTM layer structure. by C. Olah, 2015, http://colah.github.io/posts/2015-08-Understanding-LSTMs

Here, X_t is a single cell value of an input vector (matrix for a multi dimensional input, such as images) and h_t is the first layer output or the second layer input. There is a single link from and to cells in a layer.

In case of LSTM, there is this same chain like structure, with a unique four-layer neural network.

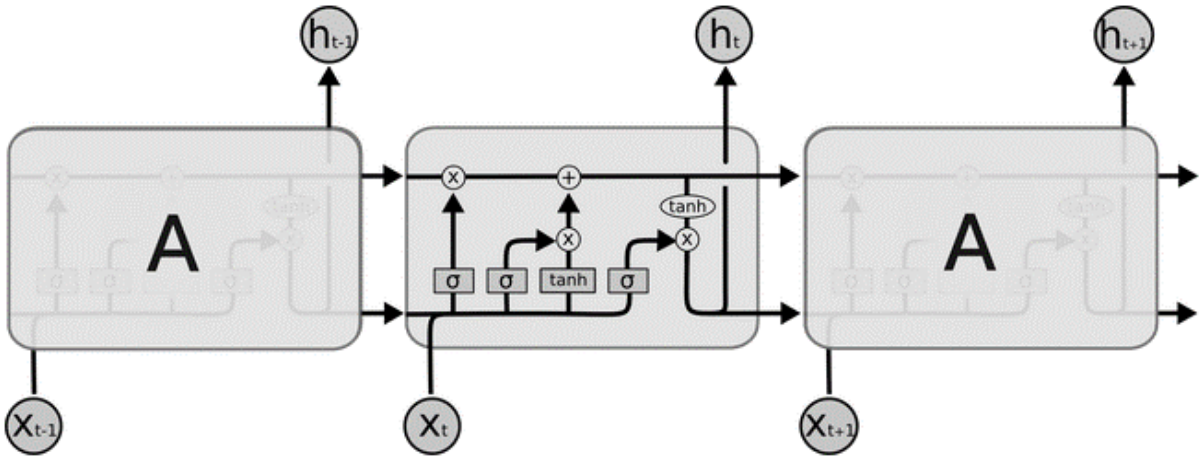

Figure 4. LSTM cell detailed structure. by C. Olah, 2015, http://colah.github.io/posts/2015-08-Understanding-LSTMs

A forget gate layer, to decide to discard information, implements a sigmoid function layer.

$$f_t = \sigma(W_f.[h_{t-1}, X_t] + b_f)$$

Then, an input gate layer decides which values are needed to be updated. This is also a sigmoid function.

$$i_t = \sigma(W_i.[h_{t-1}, X_t] + b_i)$$

After that, new candidate values, $\tilde{C}_t$ are created as a vector by a tanh layer. These values are added to the state.

$$\tilde{C}_t = \tanh(W_c.[h_{t-1}, X_t] + b_c)$$

Then the final C_t is calculated discarding the f_t and propagated to a sigmoid function and a tanh function to get our final output.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_0.[h_{t-1}, X_t] + b_0)$$

$$h_t = o_t * \tanh(C_t)$$

There are several other variants of LSTM, peephole connection, Gated Recurrent Unit, Depth Gated RNN, Clockwork RNN etc. But we will be using the traditional LSTM in our work.

For our task, we will be using keras, a deep learning framework, packaged as a python library. It is programmable to use Graphics Processing Unit alongside Central Processing Unit for faster matrix operations.

As any other supervised learning algorithm, we will be needing to split our whole dataset (for a particular company) into training and testing datasets to implement the LSTM model.

3.3 RMSE

Root Mean Square Error is a method to measure errors thorough the standard deviation of the residuals. This quadratic scoring method penalizes the larger errors the most. This means RMSE is more useful when large errors are particularly undesirable.

The formula of RMSE stands-

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Here, n is the number of observations, y_j is the true value, whereas, $\hat{y}_j$ is the predicted value.

Chapter 4

Data Collection and Preprocessing

As we have mentioned earlier, there are currently 360 companies (without the treasury bonds) in Dhaka Stock Exchange. This list of companies is available in their official website. To scrape the list, we have written a python script.

When the company list is ready, we will be collecting the historical data for each of those companies. Dhaka Stock Exchange distributes the data physically and through websites like AmarStock, who do not publish the data, rather use it for plotting and visualization purpose. The dynamic graph view of different companies is publicly available in AmarStock, and the graphs are generated by using JSON formatted historical data, which can be intercepted and collected.

For each company, we triggered an HTTP request containing start date (start EPOCH), end date (end EPOCH) and the company name. Here, the EPOCHs are the UNIX formatted time. This request returns data for all available dates in between the dates. For example, we are trying to retrieve data of BATBC for a particular date range. The result looks like this-


```
"{\n \"t\": [1543708800, 1543795200, 1543881600, 1543968000, 1544054400,
1544313600, 1544400000, 1544486400, 1544572800, 1544659200, 1545004800,
1545091200, 1545177600, 1545264000, 1545523200, 1545609600, 1545782400,
1545868800],\n \"c\": [3291.1, 3308.5, 3290.3, 3299.8, 3293.1, 3313.1, 3344.6,
3483.8, 3471.1, 3398.5, 3397.1, 3374.5, 3427.9, 3456.7, 3503.2, 3524.9, 3536.5,
3541.7],\n \"o\": [3330, 3281.2, 3320, 3300, 3280, 3293.1, 3314, 3351.2, 3461,
3400, 3374, 3394.4, 3398.4, 3420, 3460.1, 3461, 3528, 3539],\n \"h\": [3335,
3354.6, 3320, 3300, 3300, 3335, 3370, 3511.8, 3497.3, 3448.1, 3400, 3397.7, 3430,
3499.9, 3530, 3530, 3550, 3550],\n \"l\": [3280, 3280, 3280, 3294, 3275.1, 3280,
3300, 3340, 3400, 3385, 3370, 3350.1, 3398.4, 3391, 3460.1, 3461, 3470, 3500],\n
\"v\": [104, 386, 403, 323, 435, 89, 14805, 17536, 2298, 623, 314, 124, 15881,
30028, 5621, 1349, 5747, 2050],\n \"s\": \"ok\"\n}"
```

This data contains timestamps (t), closing prices (c), opening prices (o), highest price (h), lowest price (l) and volumes of trade (v). After collecting data for all companies, we have formatted the data in a human readable manner. The timestamps are not converted to regular date formats for simplicity.

Company	Time	Opening	Closing	High	Low	Volume
BATBC	1543708800	3330	3291.1	3335	3280	104
BATBC	1543795200	3281.2	3308.5	3354.6	3280	386
BATBC	1543881600	3320	3290.3	3320	3280	403
BATBC	1543968000	3300	3299.8	3300	3294	323
BATBC	1544054400	3280	3293.1	3300	3275.1	435
BATBC	1544313600	3293.1	3313.1	3335	3280	89
BATBC	1544400000	3314	3344.6	3370	3300	14805
BATBC	1544486400	3351.2	3483.8	3511.8	3340	17536
BATBC	1544572800	3461	3471.1	3497.3	3400	2298
BATBC	1544659200	3400	3398.5	3448.1	3385	623
BATBC	1545004800	3374	3397.1	3400	3370	314
BATBC	1545091200	3394.4	3374.5	3397.7	3350.1	124
BATBC	1545177600	3398.4	3427.9	3430	3398.4	15881
BATBC	1545264000	3420	3456.7	3499.9	3391	30028
BATBC	1545523200	3460.1	3503.2	3530	3460.1	5621
BATBC	1545609600	3461	3524.9	3530	3461	1349
BATBC	1545782400	3528	3536.5	3550	3470	5747
BATBC	1545868800	3539	3541.7	3550	3500	2050

Table 1. First 18 rows of the dataset of BATBC

Our timeframe for all companies was from 1 January 2000 to 2 January 2019, which means, for older companies (e.g. BATBC), we have ignored the data before 1 January 2000. For the rest, we retrieved their data as soon as they came into the market.

In this way, we have collected 845808 rows of data, including all the companies. For each company, separate csv files are created. We want to do our analysis based on the closing prices, for which we dropped rest of the columns, keeping only “company”, “time” and “closing” column.

Now, the datasets are ready for the next phase. After all the processing and formatting, they look like this-

Company	Time	Closing
BATBC	1543708800	3291.1
BATBC	1543795200	3308.5
BATBC	1543881600	3290.3
BATBC	1543968000	3299.8
BATBC	1544054400	3293.1
BATBC	1544313600	3313.1
BATBC	1544400000	3344.6
BATBC	1544486400	3483.8
BATBC	1544572800	3471.1
BATBC	1544659200	3398.5
BATBC	1545004800	3397.1
BATBC	1545091200	3374.5
BATBC	1545177600	3427.9
BATBC	1545264000	3456.7
BATBC	1545523200	3503.2
BATBC	1545609600	3524.9

BATBC	1545782400	3536.5
BATBC	1545868800	3541.7

Table 2. First 18 rows of BATBC dataset after formatting

Chapter 5

Experiment and Result

For our experiment, we will be trying to predict the closing price of a company of day `t` based on its prices of previous 60 days, meaning, from day `t-61` to day `t-1`. To simultaneously execute ARIMA and LSTM, we have written a python program, which will take a company, fetch its dataset, analyze it using both algorithms, store the results and move on to the next company. The pseudo code will be-

```
for a_company from list_of_companies():
       dataset = retrieve_dataset(a_company)
       dataset.make_index("Date")
       dataset.keep_columns("Closing")

       arima_store = perform_arima(dataset)
       lstm_store = perform_lstm(dataset)

       ##Performance analysis
       arima_performance = rmse(arima_store)
       lstm_performance = rmse(lstm_store)
       performance_store(arima_performance, lstm_performance)

generate_result()
```

The `perform_arima` and `perform_lstm` functions are defined in their respective sections.

To make the dataset a vector for a particular company, we have dropped the default index (0, 1, 2, 3, ….) and used the date of the data as index and sort the data. Now, the dataset is a vector with only closing prices, indexed by the dates. After that, we can format the date from this numerical UNIX format to human readable date-time (YYYY-MM-dd) format. This data will

only be stored in the memory. After being analyzed, this temporary data will be discarded. Now, the data will look like this-

	Closing
Date	
2000-01-01	103.3
2000-01-02	103.0
2000-01-03	99.9
2000-01-04	99.9
2000-01-06	111.0
2000-01-08	105.0
2000-01-10	105.4
2000-01-11	104.0
2000-01-12	103.5
2000-01-13	103.0

Table 3. Data structure of BATBC dataset before analysis with ARIMA model

Due to very low amount of data, we had to exclude ACFL, BDSERVICE, DEBARACEM, DEBBDLUGG, DEBBDWELD, DEBBDZIPP, DEBBXDENIM, DEBBXFISH, DEBBXKNI, DEBBXTEX, GENEXIL, IBP, KAY&QUE, KTL, MLDYEING, SEMLFBSLGF, SILVAPHL, SKTRIMS, SSSTEEL, VFSTDL from our analysis.

Now we will proceed to our next task.

5.1 ARIMA

As we mentioned in the methodology, we will use pyramid-arima, a python library, to programmatically create ARIMA models and choose the best model for prediction. So far, we have used BATBC data for demonstration purpose. We will do the same here.

Here, we will divide our dataset into two equal half and use the later one. This is because, in our LSTM model creation step, we will be needing first half of the data just to train the model, on which we will not be performing any prediction. That data will be unavailable to compare with if we use the whole dataset in our ARIMA model creation step. For the sake of consistency, we are sacrificing first half of the dataset in this step.

The following pseudo-code shows how we will proceed with this step-

```
def perform_arima(dataset):
	data_to_be_analyzed = last_half_of_the_dataset(dataset)
	for data from data_to_be_analyzed:
		date = data.date()
		train_data = data_to_be_analyzed(from=date-61, to=date-1)
		model = auto_arima(train_data).fit()
		predicted = model.predict(for=1)		//for 1 day
		arima_store(data.value, predicted.value)

	return arima_store
```

What we are doing here is, for a company, creating a separate model for each prediction. A single model for a company is possible, but auto ARIMA will then show us a trend, not any particular value for future dates.

Here is the first iteration of prediction for BATBC-

```
<---1/2312--->

Fit ARIMA: order=(2, 1, 2) seasonal_order=(0, 0, 0, 1); AIC=nan, BIC=nan, Fit
time=nan seconds

Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 0, 0, 1); AIC=349.070, BIC=353.225,
Fit time=0.010 seconds

Fit ARIMA: order=(1, 1, 0) seasonal_order=(0, 0, 0, 1); AIC=350.970, BIC=357.202,
Fit time=0.035 seconds

Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 0, 0, 1); AIC=350.944, BIC=357.177,
Fit time=0.087 seconds

Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 0, 0, 1); AIC=nan, BIC=nan, Fit
time=nan seconds

Total fit time: 0.140 seconds
```

We can see that the ARIMA(0,1,0), or I(1), has the least AIC and BIC score. So, our algorithm will choose this model over others and will predict using this. We can find the details of our model in the model summary.

```
Statespace Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   60
Model:               SARIMAX(0, 1, 0)   Log Likelihood                -172.535
Date:                Sat, 23 Mar 2019   AIC                            349.070
Time:                        13:48:33   BIC                            353.225
Sample:                             0   HQIC                           350.692
                                  - 60
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0034      0.747      0.005      0.996      -1.460       1.467
sigma2        20.3030      3.559      5.705      0.000      13.328      27.278
===================================================================================
Ljung-Box (Q):                       30.63   Jarque-Bera (JB):                19.56
Prob(Q):                              0.86   Prob(JB):                         0.00
Heteroskedasticity (H):               0.56   Skew:                             1.17
Prob(H) (two-sided):                  0.20   Kurtosis:                         4.58
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Predicted value:213.80338983050848
```

Here, for this one prediction, Ljung-Box test's p-value is well above 0.05, which indicates non-significance, which is desirable. On the other hand, a 0.0 p-value means our data is normally distributed. We can see our predicted value at the bottom.

After all the analyses, our final output for BATBC (for the first 3 months) will look like this-

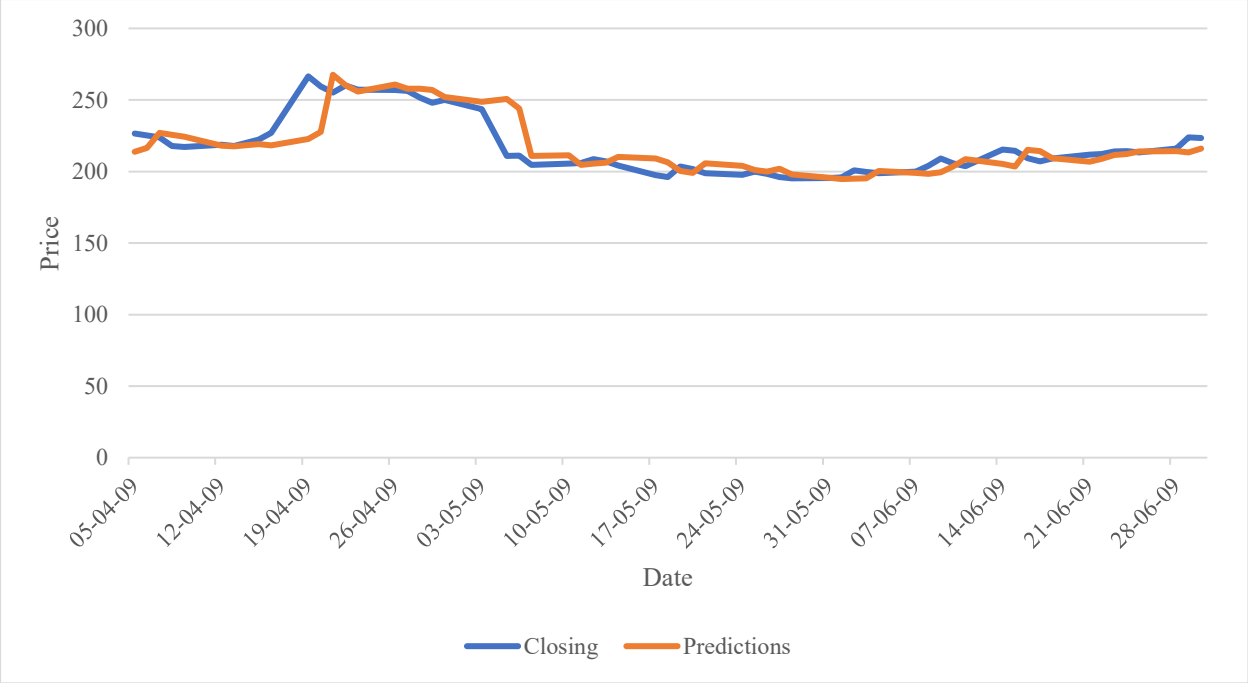

Figure 5. BATBC stock price actual movement vs. ARIMA predictions for first 3 months in test data

The result is generated from the following data (for first 3 months), generated by our program-

Date	Closing	Predictions
05-04-09	226.6	213.8033898
06-04-09	225.3	216.4559322
07-04-09	223.8	226.9288136
08-04-09	217.8	225.6898305
09-04-09	217.2	224.2084746
12-04-09	218.8	218.1084746
13-04-09	217.8	217.4864407
15-04-09	222.2	219.1271186
16-04-09	227.1	218.1864407
19-04-09	266.5	222.740678
20-04-09	259.4	227.5847458
21-04-09	255.1	267.620339
22-04-09	260	260.420339
23-04-09	257.2	255.9237288
26-04-09	257	260.7627119
27-04-09	256.2	257.9457627
28-04-09	251.5	257.8016949

29-04-09	248.1	256.9457627
30-04-09	250	252.0881356
03-05-09	243.5	248.640678
05-05-09	210.8	250.6271186
06-05-09	211.2	244.0186441
07-05-09	204.6	210.8338983
10-05-09	205.5	211.3220339
11-05-09	206	204.6728814
12-05-09	208.6	205.6016949
13-05-09	207	206.1949153
14-05-09	204.1	210.2830771
17-05-09	197.5	209.0321978
18-05-09	196.2	206.3169367
19-05-09	203.5	200.2624972
20-05-09	201.6	198.917421
21-05-09	198.8	205.6448828
24-05-09	197.7	203.8406194
25-05-09	199.9	201.0187471
26-05-09	198.3	199.8618774
27-05-09	196.1	201.8487844
28-05-09	195.2	197.9254237
31-05-09	195.5	195.6033898
01-06-09	195.9	194.7118644
02-06-09	200.8	195.0067797
03-06-09	199.6	195.2949153
04-06-09	198.8	200.2694915
07-06-09	200	199.1254237
08-06-09	204	198.320339
09-06-09	209.2	199.4898305
10-06-09	205.8	203.4966102
11-06-09	203.8	208.7372881
14-06-09	215.4	205.3661017
15-06-09	214.4	203.4050847
16-06-09	209.4	215.2338983
17-06-09	207	214.2474576
18-06-09	209	209.2220339

21-06-09	211.7	206.8220339
22-06-09	212.2	208.859322
23-06-09	214	211.6372881
24-06-09	214.2	212.220339
25-06-09	213.3	213.9932203
28-06-09	216	214.2169492
29-06-09	223.9	213.2915254
30-06-09	223.4	215.9949153

Table 4. First few rows of ARIMA model predictions for BATBC dataset

5.2 LSTM

LSTM is a subset of RNN, which is a subset of deep learning. One of the most popular deep learning frameworks is Keras. We will use TensorFlow, a python library that is maintained by Google, for this task, in backend with Keras. Like most other deep learning applications, our task is very computation intensive. The machine that we are using to run our program has a GPU available with CUDA support. Which means, we can configure our program to use our GPU for incredibly faster computation, alongside our CPU. TensorFlow has built in support for this.

We will now split our dataset into two equal halves. First 50% data will be used for training, and the last 50% data will be used for testing. Inside our program, we will format our data in the following way, where Y is the dependent variable, and X1, X2, X3, …, X60 are the independent variables.

X1	X2	X3	…	X58	X59	X60	Y
103.3	103	99.9	…	98.8	100.2	100	99.9
103	99.9	99.9	…	100.2	100	99.9	99.1
99.9	99.9	111	…	100	99.9	99.1	99.1
99.9	111	105	…	99.9	99.1	99.1	99

111	105	105.4	…	99.1	99.1	99	98.9
105	105.4	104	…	99.1	99	98.9	98
105.4	104	103.5	…	99.1	99	98	98.8
104	103.5	103	…	99	98	98.8	99.1
103.5	103	104.4	…	98	98.8	99.1	99.5
103	104.4	106.7	…	98.8	99.1	99.5	99.7

Table 5. Data structure of BATBC dataset before analysis with LSTM model

After training, our model should be capable of looking at consecutive 60 working days' data and predict for the next day. To test the performance, we will format out testing data similarly, without the Y column. It will generate its prediction, and after that, we will be able to measure the model's performance.

Our algorithm for the prediction task is as follows-

```
def perform_lstm(dataset):
	training = first_half_of_the_dataset(dataset)
	testing = last_half_of_the_dataset(dataset)
	date = find_61st_data(training).date()
	training.transform(independent={date-61, date-1}, dependent=date)
	lstm.fit(training)

	for data from testing:
		date = data.date()
		independent_values = testing(from=date-61, to=date-1)
		predicted = lstm.predict(independent_values)
		lstm_store(data.value, predicted.value)

	return lstm_store
```

Here, we will create a single model, which will make predictions for all testing data. We will now train our LSTM model for BATBC. First, or GPU will get configured.

```
Using TensorFlow backend.

Epoch 1/1

2019-03-23 14:14:04.806156: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: FMA

2019-03-23 14:14:04.926956: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1405] Found device 0 with properties:

name: GeForce GTX 750 Ti major: 5 minor: 0 memoryClockRate(GHz): 1.202

pciBusID: 0000:01:00.0

totalMemory: 1.95GiB freeMemory: 1.46GiB
```

Now, the program will begin its task.

```
2019-03-23 14:14:04.927006: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1484] Adding visible
gpu devices: 0

2019-03-23 14:14:05.938439: I tensorflow/core/common_runtime/gpu/gpu_device.cc:965] Device
interconnect StreamExecutor with strength 1 edge matrix:

2019-03-23 14:14:05.938483: I tensorflow/core/common_runtime/gpu/gpu_device.cc:971]      0

2019-03-23 14:14:05.938500: I tensorflow/core/common_runtime/gpu/gpu_device.cc:984] 0:   N

2019-03-23 14:14:05.938732: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1097] Created
TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 1211 MB memory) -> physical GPU
(device: 0, name: GeForce GTX 750 Ti, pci bus id: 0000:01:00.0, compute capability: 5.0)

 - 529s - loss: 1.2374e-05

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

lstm_1 (LSTM)                (None, 60, 50)            10400

_________________________________________________________________

lstm_2 (LSTM)                (None, 50)                20200

_________________________________________________________________

dense_1 (Dense)              (None, 1)                 51

=================================================================

Total params: 30,651

Trainable params: 30,651

Non-trainable params: 0

_________________________________________________________________

None

62.751516029346334

            Closing  Predictions

Date

2009-04-05   226.6   214.660004

2009-04-06   225.3   216.795258

2009-04-07   223.8   219.017227

2009-04-08   217.8   220.612503

2009-04-09   217.2   220.640411

2009-04-12   218.8   220.030380

2009-04-13   217.8   219.595688

2009-04-15   222.2   219.110596

2009-04-16   227.1   219.482590

2009-04-19   266.5   220.953049
```

For demonstration, we generated first 10 predictions in the log. As it shows, our model is fed with 30651 parameters. The output shape of lstm_1 layer has dimension of 60x50. Here, 60 is the number of parameters for each data point. 50 is the number of cells, that our model has generated itself, to calculate our prediction. The dense_1 layer accumulates the result from these cells into a number. The training data has an RMS error of 62.75. It is not on the predicted value.

After the model for BATBC has been trained, the following result is generated (for the first 3 months)-

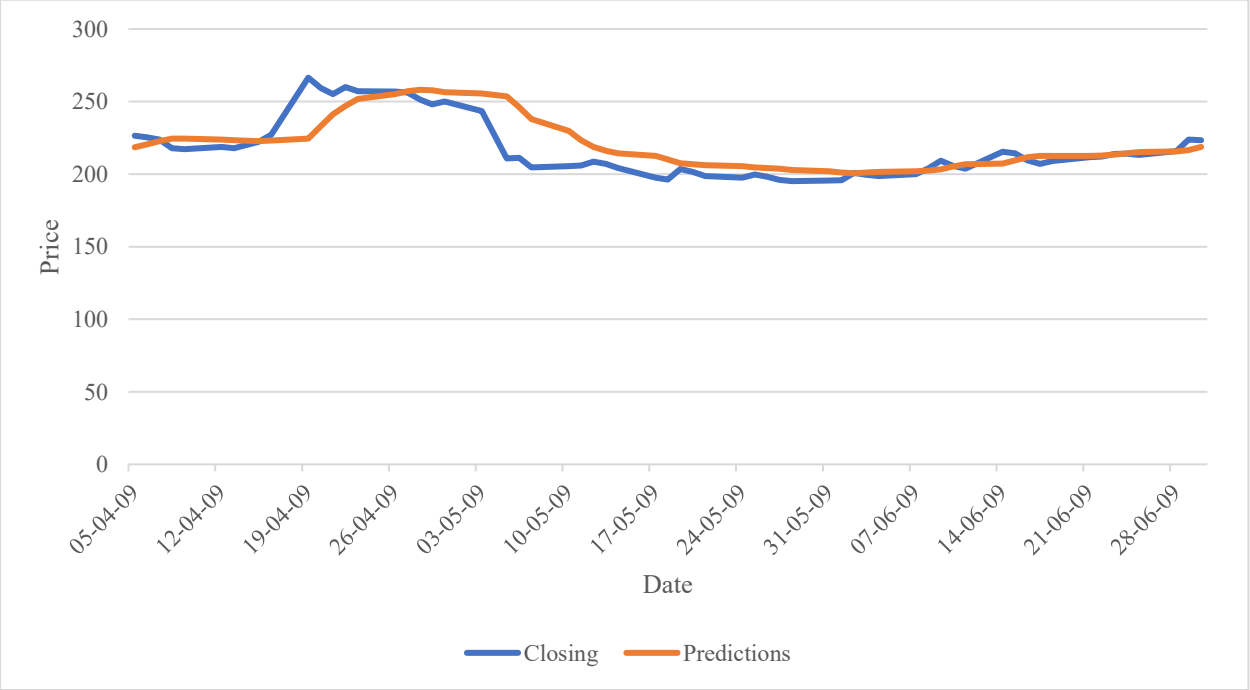

Figure 6. BATBC stock price actual movement vs. LSTM predictions for first 3 months in test data

The result is generated from the following generated data (for first 3 months)-

Date	Closing	Predictions
05-04-09	226.6	218.5422
06-04-09	225.3	220.6361
07-04-09	223.8	222.8292
08-04-09	217.8	224.4063

09-04-09	217.2	224.4357
12-04-09	218.8	223.8148
13-04-09	217.8	223.3501
15-04-09	222.2	222.8385
16-04-09	227.1	223.1699
19-04-09	266.5	224.5902
20-04-09	259.4	233.0465
21-04-09	255.1	241.1791
22-04-09	260	247.022
23-04-09	257.2	251.9011
26-04-09	257	255.0796
27-04-09	256.2	257.0644
28-04-09	251.5	258.1372
29-04-09	248.1	257.877
30-04-09	250	256.6094
03-05-09	243.5	255.558
05-05-09	210.8	253.6339
06-05-09	211.2	246.0258
07-05-09	204.6	237.8975
10-05-09	205.5	229.8737
11-05-09	206	223.4783
12-05-09	208.6	218.8807
13-05-09	207	216.1923
14-05-09	204.1	214.3742
17-05-09	197.5	212.6822
18-05-09	196.2	210.1812
19-05-09	203.5	207.6143
20-05-09	201.6	206.7493
21-05-09	198.8	206.2787
24-05-09	197.7	205.5294
25-05-09	199.9	204.6143
26-05-09	198.3	204.177
27-05-09	196.1	203.6879
28-05-09	195.2	202.8713
31-05-09	195.5	201.9182
01-06-09	195.9	201.1286

02-06-09	200.8	200.5959
03-06-09	199.6	201.0971
04-06-09	198.8	201.6488
07-06-09	200	201.9765
08-06-09	204	202.3742
09-06-09	209.2	203.4203
10-06-09	205.8	205.41
11-06-09	203.8	206.7723
14-06-09	215.4	207.3254
15-06-09	214.4	209.5302
16-06-09	209.4	211.7308
17-06-09	207	212.6643
18-06-09	209	212.5749
21-06-09	211.7	212.4958
22-06-09	212.2	212.9039
23-06-09	214	213.4954
24-06-09	214.2	214.3644
25-06-09	213.3	215.2139
28-06-09	216	215.7523
29-06-09	223.9	216.5649
30-06-09	223.4	218.7146

Table 6. First few rows of LSTM model predictions for BATBC dataset

5.3 Performance Comparison

As we have discarded 20 companies, we have 340 companies on which we have performed prediction using both ARIMA and LSTM models. To calculate RMSE for all the companies with both the models, we have used a python script. Here are the first 10 rows that we have generated.

Company	LSTM	ARIMA
AAMRANET	98.88733	6.65985
AAMRATECH	3.929954	0.901877
ABB1STMF	0.09201	0.022498

ABBANK	10.60286	7.833885
ACI	1058.243	146.4122
ACIFORMULA	144.9034	31.74873
ACMELAB	8.328857	2.536109
ACTIVEFINE	41.99166	1.829533
ADVENT	17.17759	4.563646
AFCAGRO	7.371071	2.372637
AFTABAUTO	10586.23	88.72616
AGNISYSL	1.519027	1.344983
AGRANINS	0.936153	0.548774
AIBL1STIMF	0.098409	0.048032
AIL	141.5319	10.27553
AL-HAJTEX	27.92287	22.27259
ALARABANK	1.644329	0.302286
ALIF	1.849929	0.546069
ALLTEX	1.035904	1.315914
AMANFEED	31.91061	4.419708

Table 7. First few rows of final error output data (full output appear in the appendix)

In case of 329 companies out of 340, ARIMA has produced less error than LSTM. Meaning, 96.8% of the time, ARIMA performed better.

Chapter 6

Discussion

As we have seen, our LSTM model performed terribly awful in comparison to ARIMA. There are some possible causes behind this.

Firstly, deep learning algorithms performs brilliantly on huge datasets. Although our datasets cannot be called small, when we actually see the number of training data rows in the oldest companies (BATBC = 2312, BATASHOE = 2309), it is quite negligible compared to the amount of data these algorithms are designed to be trained upon.

Secondly, in our LSTM model, we used 60 parameters. For older companies with high data rows, it might not hurt that much, but the newer companies with low number of data rows has suffered with lower degrees of freedom. Furthermore, we have used a generalized approach to split the datasets into 50% training and 50% test data, whereas small datasets should have had 70% training data, so that 30% test data could have been used to test both the LSTM model and ARIMA model.

Thirdly, there are different variations of LSTMs. LSTM has been proved to effective in developed countries, where markets follow the random walk model. Traditional methods like ARIMA cannot perform better there. Highly volatile markets like ours might need some other variation of LSTM with some more fine tuning to make prediction.

Chapter 7

Conclusion and Future Works

ARIMA has been reliably used in Econometrics to forecast stock market movement for decades. It is no surprise that it performed so well. Although, deep learning algorithms are now so widely used across sectors and they perform so well in their tasks, it is surprising to see it performing so bad here. In future, we will try to come up with a fine-tuned LSTM variant with some other optimizers (here we have used Adam optimizer). Apart from that, there is a scope to blend ARIMA and LSTM together, where we can create ARIMA models for earlier data and use them as training set to develop a LSTM model to generalize them for test data.

Moreover, this work is not yet ready to have an applied usage. An online tool with dynamically created the models and forecasting features is possible with the work that we have done here. These all rests as future works.

References

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice. OTexts.

Bepari, M.K. and Mollik, A. (2008). Bangladesh Stock Market Growing? a key indicators based assessment. Journal of Business Administration Online (JBAO), Issue 8, 2008, Arkansas: School of Business, Arkansas Tech University.

Solnik, B. (1973). Note on the Validity of the Random Walk for European Stock Prices. Journal of Finance, 28: 1151–1159.

Ojah, K. & Karemera, D. (1999). Random walks and market efficiency tests of Latin American emerging equity markets. The Financial Review, 34(1), 57-72.

Branes, P. (1986). Thin trading and stock market efficiency: A case of the Kuala Lumpur Stock Exchange. Journal of Business Finance & Accounting, volume 13(4) winter, pp. 609- 617.

Khababa, N. (1998), Behavior of stock prices in the Saudi Arabian Financial Market: Empirical research findings. Journal of Financial Management & Analysis, vol.11(1), pp.48-55, Jan-June.

Harvey, C. R., (1994), "Conditional Asset allocation in Emerging Markets", Working Paper No. 4623, Cambridge, MA.

Irfan, M., Irfan, M. & Awais, M. (2010). Investigating the weak form efficiency of an emerging market using parametric tests: Evidence from Karachi Stock Market of Pakistan. Electronic Journal of Applied Statistical Analysis, 3(1), 52-64.

Rahman, S. & Hossain, F. (2006), Weak-Form Efficiency: Testimony of Dhaka Stock Exchange. Journal of Business Research, 8, pp.1-12.

Jia, H. (2016). Investigation into the effectiveness of long short term memory networks for stock price prediction. arXiv preprint arXiv:1603.07893.

Mittal, A., & Goel, A. (2012). Stock prediction using twitter sentiment analysis. Standford University, CS229 (2011 http://cs229. stanford. edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis. pdf), 15.

Bao W, Yue J, & Rao Y (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLOS ONE 12(7): e0180944. https://doi.org/10.1371/journal.pone.0180944

Di Persio, L., & Honchar, O. (2016). Artificial neural networks architectures for stock price prediction: Comparisons and applications. International Journal of Circuits, Systems and Signal Processing, 10, 403-413.

Roondiwala, M., Patel, H., & Varma, S. (2017). Predicting stock prices using LSTM. International Journal of Science and Research (IJSR), 6(4), 1754-1756.

Siami-Namini, S., & Namin, A. S. (2018). Forecasting economics and financial time series: Arima vs. lstm. arXiv preprint arXiv:1803.06386.

Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of ARIMA and artificial neural networks models for stock price prediction. Journal of Applied Mathematics, 2014.

Olah, C. (2015, August 27). Understanding LSTM Networks. Retrieved March 22, 2019, from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Haider, A. S. (2009). Forecasting Dhaka Stock Exchange (DSE) return: An Autoregressive Integrated Moving Average (ARIMA) Approach. North South Business Review. 3 and 4. 36-54.

Appendix A

	Company	LSTM Error	ARIMA Error
1	AAMRANET	98.88733142	6.6598496
2	AAMRATECH	3.929954041	0.901876984
3	ABB1STMF	0.092010453	0.022497907
4	ABBANK	10.60286068	7.833884981
5	ACI	1058.242671	146.4121743
6	ACIFORMULA	144.9034038	31.74873096
7	ACMELAB	8.328856975	2.536109223
8	ACTIVEFINE	41.99165562	1.829533125
9	ADVENT	17.17758575	4.563645829
10	AFCAGRO	7.371071266	2.372636858
11	AFTABAUTO	10586.22518	88.72616058
12	AGNISYSL	1.519027197	1.344982773
13	AGRANINS	0.936153109	0.548774315
14	AIBL1STIMF	0.098408723	0.048032381
15	AIL	141.5319331	10.27552572
16	AL-HAJTEX	27.92286873	22.27258705
17	ALARABANK	1.644328701	0.302285938
18	ALIF	1.849929208	0.546069322
19	ALLTEX	1.03590376	1.315913542
20	AMANFEED	31.91060618	4.419708098
21	AMBEEPHA	616.0797522	512.9411415
22	AMCL(PRAN)	140.0777874	43.95830416
23	ANLIMAYARN	6.529018333	1.354773997
24	ANWARGALV	16.3507385	6.519891715
25	APEXFOODS	26.55859003	29.732046
26	APEXFOOT	203.5625464	108.814142
27	APEXSPINN	353.3865419	22.96977748
28	APEXTANRY	74.3599686	29.08725654
29	APOLOISPAT	0.78859196	0.246368557
30	ARAMIT	528.72264	316.2668544
31	ARAMITCEM	11.56361139	5.514133221
32	ARGONDENIM	4.208438259	0.864184387
33	ASIAINS	4.913638759	0.468096461
34	ASIAPACINS	4.064647532	0.705236379
35	ATCSLGF	0.246675391	0.047885208
36	ATLASBANG	141.390108	121.3552337
37	AZIZPIPES	65.5689957	53.43548749
38	BANGAS	325.1362952	292.3019637
39	BANKASIA	10.23225846	0.606972103
40	BARKAPOWER	4.321542522	1.015582698
41	BATASHOE	1092.522226	387.2797209
42	BATBC	3883.148075	1770.846931
43	BAYLEASING	8.616053732	0.766406141
44	BBS	5.437145604	1.89722186
45	BBSCABLES	85.29397016	23.77632118
46	BDAUTOCA	169.320176	59.90321436
47	BDCOM	3.216796556	1.640169114
48	BDFINANCE	6.740352739	0.487013836
49	BDLAMPS	56.88838254	54.69409761
50	BDTHAI	56.24441968	6.730138136
51	BDWELDING	44.4903013	36.7977679
52	BEACHHATCH	1.270715993	0.862552244
53	BEACONPHAR	3.204187579	0.297143714
54	BENGALWTL	3.945111143	1.818123538
55	BERGERPBL	15581.23203	4626.863283
56	BEXIMCO	369.5147595	54.64314642
57	BGIC	9.400822364	3.63348104
58	BIFC	0.780779015	0.360948671
59	BNICL	1.24202994	0.229601283
60	BPML	205.9024375	15.17166773
61	BRACBANK	11.86373475	3.948660278
62	BSC	3670.370043	1362.167714
63	BSCCL	19.31893297	8.311901772
64	BSRMLTD	34.58481438	8.262722599
65	BSRMSTEEL	10.76528128	5.124841511
66	BXPHARMA	30.27311082	10.49291872
67	BXSYNTH	0.32786942	0.261809415
68	CAPMBDBLMF	0.405000812	0.073491589
69	CAPMIBBLMF	0.430039121	0.210174669
70	CENTRALINS	10.67712258	0.617759022
71	CENTRALPHL	1.369540215	0.780161169
72	CITYBANK	9.551488246	1.963498903
73	CITYGENINS	1.89078529	0.364316063
74	CNATEX	1.169912615	0.10989242
75	CONFIDCEM	27.49304742	19.94993222
76	CONTININS	1.188540923	0.549539558
77	CVOPRL	1389.642852	426.5145811
78	DACCADYE	1.980425019	0.283307147
79	DAFODILCOM	1.719811997	1.174112529
80	DBH	11.91326688	6.086832832
81	DBH1STMF	0.244857904	0.022583604
82	DELTALIFE	1111.58342	104.7828528
83	DELTASPINN	2.372139407	0.979753326
84	DESCO	59.64125073	3.189093692
85	DESHBANDHU	30.73342493	0.401878104
86	DHAKABANK	2.366529258	2.376271751
87	DHAKAINS	1.639314028	0.463376467
88	DOREENPWR	55.46200186	9.421349951
89	DSHGARME	84.37023964	78.05126609
90	DSSL	9.167464906	4.02553732
91	DULAMIACOT	3.7951851	1.78225499
92	DUTCHBANGL	1296.669945	9.222647023
93	EASTERNINS	3.085087469	1.100835767
94	EASTLAND	9.096190647	1.506442576
95	EASTRNLUB	2186.622473	1819.844044
96	EBL	4.842727238	4.206497378
97	EBL1STMF	0.076767655	0.042574229
98	EBLNRBMF	0.04392519	0.031750658
99	ECABLES	195.8016926	37.50673171
100	EHL	19.40162069	12.19165241
101	EMERALDOIL	3.928009799	1.193100803
102	ENVOYTEX	4.173214848	0.79555406
103	ETL	4.288391357	0.446747116
104	EXIM1STMF	0.046845456	0.028822827
105	EXIMBANK	0.719729327	0.228137199
106	FAMILYTEX	0.202450591	0.055810968
107	FARCHEM	3.154655733	0.541027027
108	FAREASTFIN	0.55368444	0.182334197
109	FAREASTLIF	14.77897035	10.1550722
110	FASFIN	4.112103025	0.34009872
111	FBFIF	0.067097046	0.029712769
112	FEDERALINS	0.88030562	0.297312818
113	FEKDIL	0.812981546	0.599982126
114	FINEFOODS	2.691051535	1.777872499
115	FIRSTFIN	14.25986668	0.873330594
116	FIRSTSBANK	0.414186543	0.126967335
117	FORTUNE	23.48875454	1.596062701
118	FUWANGCER	0.889114278	0.405232796
119	FUWANGFOOD	8.4426648	6.567506435
120	GBBPOWER	2.846982089	0.301366037
121	GEMINISEA	3875.009487	1741.9417

122	GENNEXT	0.124037874	0.072206343
123	GHAIL	3.061196527	2.201372249
124	GHCL	1.91360544	1.74556773
125	GLAXOSMITH	5651.638108	1209.67742
126	GLOBALINS	3.524956913	0.625743323
127	GOLDENSON	12.24950794	1.635064894
128	GP	143.5296546	44.39883618
129	GPHISPAT	2.882773664	1.99910985
130	GQBALLPEN	79.48105635	46.3717969
131	GRAMEENS2	1.199021249	0.099443241
132	GREENDELMF	0.026126917	0.019756723
133	GREENDELT	60.79779153	41.22495223
134	GSPFINANCE	1.185413352	0.699682361
135	HAKKANIPUL	13.61400126	5.796225746
136	HEIDELBCEM	2227.840641	105.7947618
137	HFL	0.983353974	0.58839941
138	HRTEX	12.36840426	2.267297049
139	HWAWELLTEX	2.378556644	0.842783512
140	IBBLPBOND	110.9175234	126.666249
141	IBNSINA	296.2701021	41.42260736
142	ICB	886.1923794	36.97490139
143	ICB2NDNRB	0.089667652	0.060123372
144	ICB3RDNRB	0.208629961	0.023936412
145	ICBAGRANI1	0.167537249	0.019483865
146	ICBAMCL2ND	1.270308871	0.044256254
147	ICBEPMF1S1	0.1054974	0.037044619
148	ICBIBANK	0.285951878	0.041283322
149	ICBSONALI1	0.056739464	0.043780947
150	IDLC	149.3857509	119.1238355
151	IFADAUTOS	23.42495431	18.70758348
152	IFIC	18.59204341	7.090147442
153	IFIC1STMF	0.236172725	0.020784989
154	IFILISLMF1	0.027581758	0.029406286
155	ILFSL	5.982166441	0.331677374
156	IMAMBUTTON	1.428733259	1.304243353
157	INTECH	2.832004245	1.946049184
158	INTRACO	9.966997914	3.384521134
159	IPDC	3.149567706	1.49854394
160	ISLAMIBANK	39.39712118	2.848771123
161	ISLAMICFIN	12.4146283	0.597498582
162	ISLAMIINS	1.365435485	0.525687128
163	ISNLTD	2.529618144	1.709431124
164	ITC	49.96313198	3.461817075
165	JAMUNABANK	3.318584346	0.32599378
166	JAMUNAOIL	40.90863914	24.07450188
167	JANATAINS	0.772089636	0.374041062
168	JMISMDL	387.6230993	43.44520637
169	JUTESPINN	83.14829385	33.150319
170	KARNAPHULI	0.533812813	0.309466828
171	KBPPWBIL	0.777890271	0.347934068
172	KDSALTD	18.25617332	10.52036697
173	KEYACOSMET	12.31253086	10.20341014
174	KOHINOOR	263.3183725	170.5478483
175	KPCL	31.17241482	8.118133644
176	KPPL	1.161238169	0.769381116
177	LANKABAFIN	13.54383608	4.741454904
178	LEGACYFOOT	41.89939761	22.73294191
179	LHBL	49.71360984	7.02534115
180	LIBRAINFU	1774.240864	1481.522937
181	LINDEBD	670.744543	486.4638263
182	LRGLOBMF1	0.056781799	0.028952107
183	MAKSONSPIN	4.696363598	0.119475737
184	MALEKSPIN	0.97454662	0.299257549
185	MARICO	1051.448789	845.9006567
186	MATINSPINN	1.380109443	0.705216294

187	MBL1STMF	0.058059045	0.034098495
188	MEGCONMILK	1.979722123	0.869055041
189	MEGHNACEM	373.2260899	63.40334518
190	MEGHNALIFE	21.85678868	7.98978881
191	MEGHNAPET	1.10346304	0.628776688
192	MERCANBANK	0.955815765	0.488649632
193	MERCINS	1.28217485	0.563263319
194	METROSPIN	0.346120467	0.217694145
195	MHSML	0.925674711	0.66626487
196	MICEMENT	20.59464178	3.569664441
197	MIDASFIN	25.11484098	15.76787032
198	MIRACLEIND	2.17877693	2.825478434
199	MITHUNKNIT	12.22856094	10.46100926
200	MJLBD	72.60702343	5.967671914
201	MONNOCERA	40.19745091	41.89023856
202	MONNOSTAF	40068.49708	40143.28457
203	MPETROLEUM	39.1004485	35.45300728
204	MTB	2.137897219	0.78321065
205	NAHEEACP	182.8534234	12.05043584
206	NATLIFEINS	182.1908283	88.67718664
207	NAVANACNG	12.22004387	3.57157042
208	NBL	63.06452148	16.60421946
209	NCCBANK	2.328179317	2.198232535
210	NCCBLMF1	0.067205852	0.032272533
211	NFML	2.198747509	0.413790698
212	NHFIL	4.96918776	2.995687783
213	NITOLINS	7.642711642	1.057158771
214	NLI1STMF	0.719094512	0.046840519
215	NORTHERN	1835.489899	560.4969435
216	NORTHRNINS	5.821127789	0.88435778
217	NPOLYMAR	15.46029561	9.817683058
218	NTC	1380.347144	486.8270975
219	NTLTUBES	50.06018984	24.97918542
220	NURANI	1.466706634	0.723489332
221	OAL	1.225093421	0.268273168
222	OIMEX	9.59193424	3.373645937
223	OLYMPIC	97.87788648	57.20131481
224	ONEBANKLTD	5.863692697	0.98081385
225	ORIONINFU	13.67391445	3.974069594
226	ORIONPHARM	1.494781855	0.9121048
227	PADMALIFE	3.229494408	2.176896621
228	PADMAOIL	1584.592809	43.46868296
229	PARAMOUNT	2.976450636	0.394158875
230	PDL	2.468617601	1.021144764
231	PENINSULA	6.367972307	2.131541432
232	PEOPLESINS	7.657514761	0.922921348
233	PF1STMF	0.045860434	0.029557408
234	PHARMAID	2451.99155	221.6736365
235	PHENIXINS	25.27927334	1.155848076
236	PHOENIXFIN	25.6284631	0.894359119
237	PHPMF1	0.037222075	0.022977327
238	PIONEERINS	78.84068129	2.529596831
239	PLFSL	1.933157204	0.820954563
240	POPULAR1MF	0.062037418	0.020129297
241	POPULARLIF	1097.180206	42.07975571
242	POWERGRID	11.31412291	1.994658833
243	PRAGATIINS	13.47234818	2.033770927
244	PRAGATILIF	47.15738602	31.23099791
245	PREMIERBAN	0.208373694	0.131343942
246	PREMIERCEM	7.760846788	5.134021365
247	PREMIERLEA	3.273242765	0.212624199
248	PRIME1ICBA	0.062763816	0.030144129
249	PRIMEBANK	2.852254676	2.988634983
250	PRIMEFIN	109.3523458	0.833778131
251	PRIMEINSUR	9.766871408	0.797762198

252	PRIMELIFE	111.1468632	7.770429198
253	PRIMETEX	4.682235417	1.212762862
254	PROGRESLIF	51.00457855	15.96540073
255	PROVATIINS	1.548271411	0.373875438
256	PTL	14.35912393	2.325628357
257	PUBALIBANK	24.23695636	4.474781926
258	PURABIGEN	12.63604966	0.400068936
259	QUASEMIND	24.27396366	15.97453111
260	QUEENSOUTH	37.46637164	6.231210897
261	RAHIMTEXT	207.4083427	169.1447901
262	RAKCERAMIC	5.52469639	2.93037458
263	RANFOUNDRY	16.46138979	15.47257097
264	RDFOOD	1.167503057	0.371715028
265	RECKITTBEN	77404.11727	2847.658443
266	REGENTTEX	1.560126018	0.955626766
267	RELIANCE1	0.088822599	0.054042599
268	RELIANCINS	14.33435792	3.812579983
269	RENATA	1389.909537	1011.059315
270	RENWICKJA	3438.946616	633.0545429
271	REPUBLIC	3.44444995	1.126609422
272	RNSPIN	7.30562137	0.834646079
273	RSRMSTEEL	12.62932547	7.109917652
274	RUPALIBANK	13.27445569	9.892375274
275	RUPALIINS	4.637281994	0.758475032
276	RUPALILIFE	91.19894604	3.435295634
277	SAFKOSPINN	1.93813444	0.735004481
278	SAIFPOWER	16.39217456	3.121314
279	SAIHAMCOT	1.121328109	0.438822033
280	SAIHAMTEX	5.217945706	1.272988603
281	SALAMCRST	8.008923345	1.367826821
282	SALVOCHEM	1.933961265	0.482702563
283	SAMATALETH	30.3793324	4.992593213
284	SAMORITA	6.736163071	5.787301156
285	SANDHANINS	145.844461	5.12470764
286	SAPORTL	5.524322345	3.402875377
287	SAVAREFR	267.6291682	42.40310971
288	SEBL1STMF	0.0395723	0.036735246
289	SEMLIBBLSF	0.5660901	0.181923027
290	SEMLLECMF	0.988226864	0.08168936
291	SHAHJABANK	2.800894095	0.339106114
292	SHASHADNIM	47.36077088	5.075017902
293	SHEPHERD	4.814635435	2.229407394
294	SHURWID	2.337587998	0.841593566
295	SHYAMPSUG	7.404026148	3.501067633
296	SIBL	1.511712747	0.300380129
297	SIMTEX	5.70398389	1.768566448
298	SINGERBD	339.7307293	240.7105236
299	SINOBANGLA	5.615119462	5.96310237
300	SONALIANSH	449.9305053	179.8653013
301	SONARBAINS	1.234540708	0.512782314
302	SONARGAON	3.887128391	0.702721285
303	SOUTHEASTB	13.84370223	1.370633196
304	SPCERAMICS	2.447828965	0.34905338
305	SPCL	21.21744102	11.49087062
306	SQUARETEXT	40.16020791	20.10168023
307	SQURPHARMA	321.7366031	64.35062621
308	STANCERAM	25.13311793	13.72883593
309	STANDARINS	1.172205909	0.930076163
310	STANDBANKL	1.271135273	0.365041678
311	STYLECRAFT	60855.92185	27909.23299
312	SUMITPOWER	2.399727548	1.86169799
313	SUNLIFEINS	1.845121167	0.859053765
314	TAKAFULINS	1.23137304	0.611398148
315	TALLUSPIN	4.217510119	0.918493968
316	TITASGAS	3.269982477	1.920126889
317	TOSRIFA	2.091586213	0.919390078
318	TRUSTB1MF	0.190289817	0.025007464
319	TRUSTBANK	1.109959586	0.687206047
320	TUNGHAI	0.729301389	0.26364037
321	UCB	35.86512595	0.790168968
322	UNIONCAP	121.676716	0.576950392
323	UNIQUEHRL	2.75797476	1.863629101
324	UNITEDAIR	1.259968031	0.072424182
325	UNITEDFIN	31.0067733	13.25883358
326	UNITEDINS	9.224746337	1.512242819
327	UPGDCL	677.7177712	79.27193471
328	USMANIAGL	31.34245279	32.86533993
329	UTTARABANK	4686.249756	9.262677713
330	UTTARAFIN	117.0930096	72.66955054
331	VAMLBDMF1	0.171504065	0.06788428
332	VAMLRBBF	0.19545407	0.038244538
333	WATACHEM	906.3076049	126.6764608
334	WMSHIPYARD	5.457250787	1.912590029
335	YPL	3.00913949	0.388195834
336	ZAHEENSPIN	2.084066341	0.354380123
337	ZAHINTEX	1.49110117	0.39978249
338	ZEALBANGLA	10.60309477	3.937025822
339	1JANATAMF	0.040317379	0.02393425
340	1STPRIMFMF	1.282009781	0.216366048

Table 8: The final error data of our analysis.