Stock Market Prediction
using
Long Short-Term Memory (LSTM)

Sayak Chakraborty
saychakr13@gmail.com

Sreetam Ganguly
sreetamneverrests@gmail.com

Shidur Sharma Durba
shidurdurbacst@gmail.com

Under the supervision of: Nirnay Ghosh

Department of Computer Science and Technology
Indian Institute of Engineering Science and Technology (IIEST), Shibpur
Howrah, India

November 29th, 2019

Abstract

The Stock Market, with all its complications, is largely unpredictable in nature. Prices can move up or down depending on a large number of factors. In this report, we aim to investigate the viability of Long Short-Term Memory (LSTM) model to predict the fluctuations in the market.

1 Introduction

1.1 What Is the Stock Market?

A stock market, equity market or share market is the aggregation of buyers and sellers (a loose network of economic transactions, not a physical facility or discrete entity) of stocks (also called shares), which represent ownership claims on businesses; these may include *securities* listed on a public stock exchange, as well as stock that is only traded privately. Examples of the latter include shares of private companies which are sold to investors through equity crowdfunding platforms. Stock exchanges list shares of common equity as well as other security types, e.g. corporate bonds and convertible bonds[1].

Stock markets are inherently complex in nature with hundreds of companies being listed and the number of daily transactions often in the order of $\sim 10^7$[1]. This might prevent an easy analysis of the condition of stock market. Fortunately, any market has an “Index” value. The Stock exchange takes into account the performance of the top performers and the effect of the most influential companies and corporations listed on the stock exchange and thus calculates the Index value. It is computed from the prices of selected stocks (typically a weighted average). It is a tool used by investors and financial managers to describe the market, and to compare the return on specific investments.

1.2 What Affects the Stock Market?

Several factors may affect the stock market such as the socio economic factors including inflation, price of commodities and raw materials, the condition and stability of the government. There are human factors affecting the stock market such as psychological factors and general mood of the public, as well as the tendency of people to invest in stock market as a group (group thinking)[2].

In a paper tittled *"Speculative Dynamics"* by David M. Cutler, James M. Poterba, Lawrence H. Summers, the authors draw an analogy with gambling. In normal times the market behaves like a game of roulette; the probabilities are known and largely independent of the investment decisions of the different players. In times of market stress, however, the game becomes more like poker (herding behavior takes over). The players now must give heavy weight to the psychology of other investors and how they are likely to react psychologically[3].

1.3 Motivation for Studying the Stock Market

The understanding of the stock market is crucial for analysing the risk of investments. International organisations, governments and private entities can profit from an accurate prediction and analysis of the stock market. Both organisations and individuals can thus take steps to prevent economic disasters by looking at the analysis. There might be several reasons to study the stock market. The following reasons are some of them:

- Studying the stock markets help policy makers to take steps to preserve the fairness of the market and prevent any foul play.
- It can help in the determination of poorly performing market sectors. This can aid governments by pointing out which market sectors need government subsidies and which do not.
- Predicting the future of the stock market accurately helps the general public make informed descisions. Thus, market disasters and bubbles can be avoided.
- Predicting and probing the economic future of any asset is crucial in deciding the risk of investments. This helps in the detection and prevention of the existence of shell companies.

Stock market disasters like the Real Estate crash of 2008, Dot Com bubble of the 1990's and the 1920's market crash could have been avoided if the general public was presented with more accurate prediction of the future of the market.

With the recent advancements in machine learning and Recurrent Neural Networks (RNN's), stock market prediction is becoming more of a feasibility and a necessity.

1.4 Objective

To build a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN), which can predict the values of the indices of the stock market at a future date and to test the viability and accuracy of such a model.

2 Related Study

A similar study was conducted by Armando Bernal, Sam Fok, Rohit Pidaparthi in *Financial Market Time Series Prediction with Recurrent Neural Networks*[4]. There, employing Echo State Network

Implementation (ESN), a very good prediction of the stock market was obtained. A comparison with the Kalman Filter was done and the ESN method proved to make the error ~0.1 times the error in the Kalman Filter technique. This demonstrated the viability of a Neural Network architecture predicting the stock market with very less error.

2.1 Gaps in the State-of-the-Art

In the above study, Indian stock exchanges was not analysed. Besides that, the study did not employ the LSTM architecture for its findings or analyse any results of the LSTM architecture for the prediction of Stock Exchange Indices.

3 Prediction Model

3.1 Recurring Neural Networks (RNNs) and their use in Predicticting Time Dependent Data

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.

The term "recurrent neural network" is used indiscriminately to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behavior. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that can not be unrolled.

Both finite impulse and infinite impulse recurrent networks can have additional stored state, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph, if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of long short-term memory networks (LSTMs) and gated recurrent units.This is also called Feedback Neural Network.

Recurrent neural networks have loops. These loops enable them to predict future data taking into account previous data. This makes them exceedingly useful in prediction models where memory retention is an issue.

However, with each “pass” or “calculation”, that a RNN does, the effect of previous data becomes more and more obscure. This leads to loss of accuracy, since the model may miss or forget important aspects of data that happened a long time ago, which falls under the category of long term dependencies.

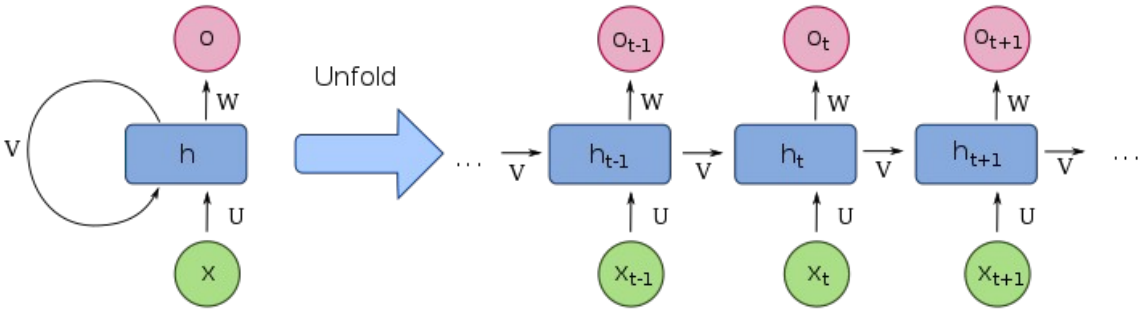


Figure 1: Diagram of a recurrent neural network (By François Deloche - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=60109157)

In theory, RNNs are absolutely capable of handling such "long-term dependencies." A human could carefully pick parameters for them to solve toy problems of this form. Sadly, in practice, RNNs don't seem to be able to learn them. The problem was explored in depth by Hochreiter (1991) [German] and Bengio, et al. (1994)[5][][6], who found some pretty fundamental reasons why it might be difficult.

3.2 Long Short-Term Memory(LSTM): It's Purpose In Prediction Of Stock Market Data

Long Short-Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997)[7], and were refined and popularized by several other people. They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn.

3.3 How LSTM Works?

Unlike standard feedforward neural networks, LSTM has feedback connections. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.

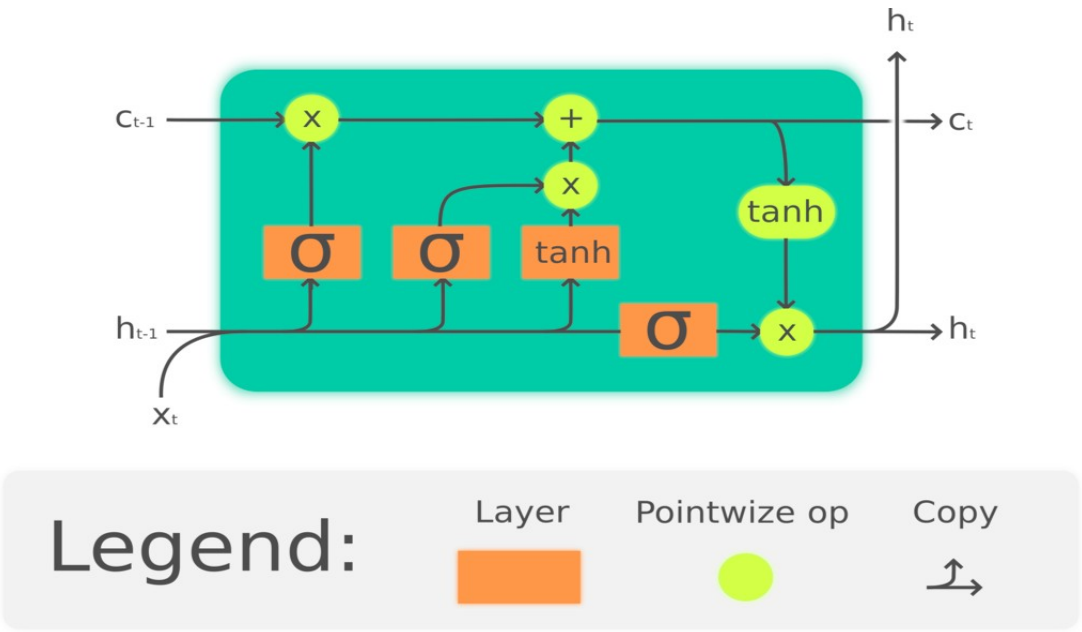


Figure 2: Diagram of a LSTM node (By Guillaume Chevalier - Own work, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=71836793)

4 Results And Discussion

4.1 Training And Testing Of The Model On Large Datasets

The dataset used is from the official website of S&P BSE. The data consists of daily stock market data, except holidays, of different indices. The indices used in this case are:

1. *S&P BSE India Government Bond Index* – It is a rules-based, transparent index that seeks to measure the performance of the 10-year Indian sovereign bond. The bond has a fixed coupon and a remaining maturity of close or equal to 10 years.

2. *S&P BSE SENSEX* – It is a free-float market-weighted stock market index of 30 well-established and financially sound companies listed on Bombay Stock Exchange. The 30 component companies which are some of the largest and most actively traded stocks, are representative of various industrial sectors of the Indian economy.

3. *S&P BSE SENSEX 50 TR* – It is a transparent, rules-based index that is designed to measure the performance of the top 50 largest and liquid stocks in the S&P BSE LargeMidCap by float-adjusted market capitalization.

4. *S&P BSE 100* – It was formerly known as the *BSE* National index. This Index has 1983-84 as the base year and was launched in 1989. In line with the shift of the *BSE* Indices to the globally accepted Free-Float methodology, *BSE-100* was shifted to Free-Float methodology effective from April 5, 2004.

5. *S&P BSE Bharat 22 Index TR* – It is designed to measure the performance of 22 select companies disinvested by the central government of India.

The datasets listed above from 1-5 are in the range of July 1 2009 to 31 December 2018. The dataset has been converted into .csv format not excluding any anomalous values (During market shutdowns and holidays). During importing of the data from .csv to usable dataframe, Pandas, a Python library is used.

```
# Importing the training set
dataset = pd.read_csv('nifty50twoyearsrecentdata.csv')

cols = dataset.columns.tolist()

# Dropping the effective date. Its not needed for now
dataset = dataset.drop(cols[0], 1)
k = len(dataset)

cols = dataset.columns.tolist()

# Now I will drop the base values. I want to work with real values
for i in cols:
    if i.find('(Base value)') != -1:
        dataset = dataset.drop(i, 1)

cols = dataset.columns.tolist()

# Removing the nan or 0 values
for i in cols:
    for j in range(1, len(dataset)):
        if dataset[i][j] == 0 or np.isnan(dataset[i][j]):
            dataset[i][j] = dataset[i][j - 1]
```

Code Snippet 1: Importing the dataset and converting the NAN values to previous day's entries.

The nan values are thus converted to the values of the previous day. This makes sense since market transactions ordered are generally made based on the most recent data. Since the amount and type of transactions determine the nature of the market, this value can give a more accurate prediction of the market.

The datafields are as follows:

- S&P BSE India Government Bond Index (Base Value)
- S&P BSE India Government Bond Index (Real Value)
- S&P BSE SENSEX (Base Value)
- S&P BSE SENSEX (Real Value)
- S&P BSE SENSEX 50 TR (Base Value)
- S&P BSE SENSEX 50 TR (Real Value)
- S&P BSE 100 (Base Value)
- S&P BSE 100 (Real Value)
- S&P BSE Bharat 22 Index TR (Base Value)
- S&P BSE Bharat 22 Index TR (Real Value)

There are two kinds of values mentioned above: Base Value and Real Value. The Real Value is the value of the Index on the corresponding day, as mentioned in daily newspapers or in the media. The Base Value is a scaled value. The first value of the dataset is taken to be 100 and the rest of the values are computed based on the original value.

Base Value(t) = {Real value(t) / Real value(0)} x 100

Where Base value(t) implies the Base value of the index for t^{th} day, Real value(t) implies the Real value of the index for t^{th} day and Real value(0) implies the Real value of the index for 1^{st} day of the dataset.

4.2 Set Up Of The LSTM Model

At first, certain parameters are chosen which determine the parameters of the LSTM model.

```
# dr is the divide ratio. It divides the dataset into dr:1 ratio
dr = 19

# f_param is the timestep function
f_param = 30

# Parameters of the RNN
num_of_middle_layers = 1
middle_layer_units = 100
middle_layer_dropouts = 0.01
RNN_epochs = 100

# Importing the training set
dataset = pd.read_csv('nifty50twoyearsrecentdata.csv')

cols = dataset.columns.tolist()

# Dropping the effective date. Its not needed for now
dataset = dataset.drop(cols[0], 1)
k = len(dataset)

cols = dataset.columns.tolist()
```

Code Snippet 2: Setting up the parameters of the LSTM model.

The parameters are self explanatory by their names. This setting of parameter values provides flexibility to the code and aids in debugging and modification.

The parameters were empirically found. These particular parameters provide a balance between training time, accuracy and error, and loss. The optimizer is set to ADAM to make training faster.

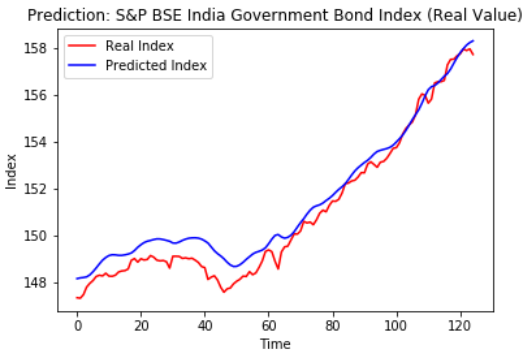
The model is set in the following code snippet.

```
# Dividing the dataset into training set and testing set
dl = int((dr / (dr + 1)) * k)
training_set = dataset.iloc[0:dl, pos:(pos + 1)].values
testing_set = dataset.iloc[(dl - 1):, pos:(pos + 1)].values

# Feature Scaling
sc = skp.MinMaxScaler(feature_range=(0, 1))
training_set_scaled = sc.fit_transform(training_set)

# Creating a data structure with f_param timesteps and 1 output
x_train = []
y_train = []
for i in range(f_param, len(training_set)):
    x_train.append(training_set_scaled[i - f_param:i, 0])
    y_train.append(training_set_scaled[i, 0])
x_train, y_train = np.array(x_train), np.array(y_train)

# Reshaping
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
#                             no of rows........timestep.........features

# Building the RNN

# Initialising the RNN
regressor = Sequential()

# Adding the first LSTM layer and some Dropout regularisation. Its the input layer
regressor.add(LSTM(units=70, return_sequences=True, input_shape=(x_train.shape[1], 1)))
regressor.add(Dropout(0.01))

# Adding the middle layers
for i in range(0, num_of_middle_layers):
    regressor.add(LSTM(units=middle_layer_units, return_sequences=True))
    regressor.add(Dropout(middle_layer_dropouts))
```

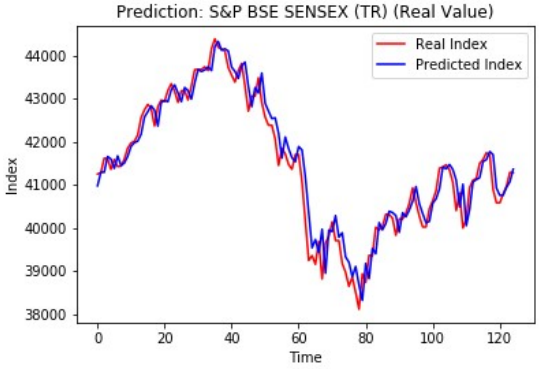
```
# Adding the penultimate LSTM layer and some Dropout regularisation
regressor.add(LSTM(units=70))
regressor.add(Dropout(0.01))

# Adding the output layer
regressor.add(Dense(units=1))

# Compiling the RNN
# instead of adam we could also have used rmsprop as the optimizer
# This is a regression problem so we use mean_squared_error
# In case of classification we could have used binary cross entropy or categorical cross entropy
regressor.compile(optimizer='adam', loss='mean_squared_error')
```

Code Snippet 3: Building the LSTM model

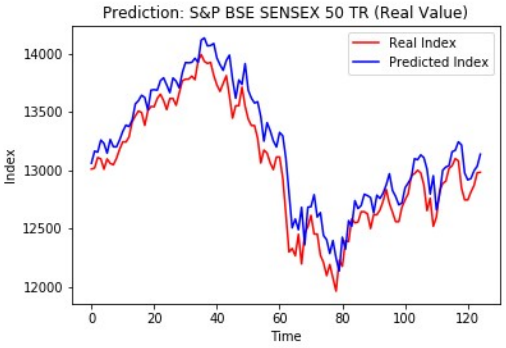
The results of the prediction model was calculated and exported into .csv format, which was later imported into a dataframe and plotted using Matplotlib library. The plots are shown in the next page.



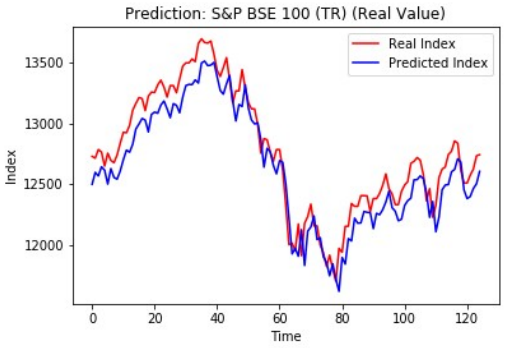
(a)



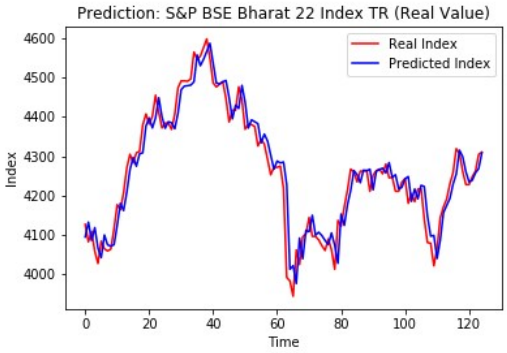
(b)



(c)



(d)



(e)

(From top to down, From left to right)

(a) Figure 1: Predicted and corresponding real values of S&P BSE India Government Bond Index (Real Value) plotted

(b) Figure 2: Predicted and corresponding real values of S&P BSE SENSEX (TR) (Real Value) plotted

(c) Figure 3: Predicted and corresponding real values of S&P BSE SENSEX 50 (TR) (Real Value) plotted

(d) Figure 4: Predicted and corresponding real values of S&P BSE 100 (Real Value) plotted

(e) Figure 5: Predicted and corresponding real values of S&P BSE Bharat 22 Index TR (Real Value) plotted

The above results are plotted with a kink in the y-axis. The accuracy of the prediction is calulated as

error (k_{ne}) = (Root Mean Square Error (n x $E^{0.5}$) x 100) / (Average of the real values (n x s))

The values are calulated in a separate function

```
def errorcalc(dat):
    dataset = pd.read_csv('data/Prediction:_'+ dat +'.csv')
    cols = dataset.columns.tolist()

    # Dropping the effective date. Its not needed for now
    dataset = dataset.drop(cols[0], 1)
    k = len(dataset)

    cols = dataset.columns.tolist()
    E = 0
    s = 0

    for i in range(k):
        e = dataset[cols[0]][i] - dataset[cols[1]][i]
        e = e*e
        s = s + dataset[cols[0]][i]
        E = E + e


    kne = (math.sqrt(E)*100)/s
    return kne
```

Code Snippet 4: Function to calculate the error of the prediction by the model

This function calculates and returns the value to the calling function, and the resulant value is exported.

	Data	Error (%)
0	S&P BSE India Government Bond Index (Real Value)	0.038412246356934
1	S&P BSE SENSEX (TR) (Real Value)	0.080711779819695
2	S&P BSE 100 (TR) (Real Value)	0.124062764136015
3	S&P BSE Bharat 22 Index TR (Real Value)	0.092884458481529
4	S&P BSE SENSEX 50 TR (Real Value)	0.136435304210502

Table 1: The percentage error made in prediction

As observed, the error is quite low, compared to the considered value to be predicted. The error in the case of S&P BSE SENSEX and S&P BSE 100 is higher than the others. This is because these are largely based on the private sector and their corresponding shares. These values are inherently more volatile since a company may face loss or gain unpredictably. Whereas, the S&P BSE India Government Bond Index has the lowest error, which makes sense since this Index is calculated based on the prices of Government Bonds, which often have a minimum holding period. This restriction on the bonds' frequent sale make them often immune to market crashes or market fluctuations. Thus, the model has less difficulty predicting the exponential-like-graph of S&P BSE India Government Bond Index.

This error may seem small when calculated based on a very large dataset (~3250 data entries). However, when calculated on a smaller dataset (~720 data entries), the accuracy drops and the data trend predicted is often erroneous.

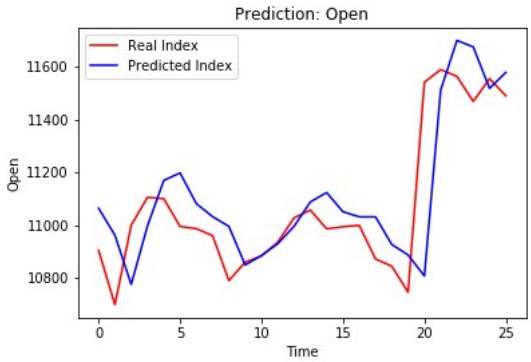
4.3 Training And Testing Of The Model By Smaller Number Of Data Entries

To test the model in this case, two years of NIFTY 50 data is considered. The dataset in this case is in the range from October 3, 2017 to September 30, 2019. The NIFTY 50 index is the National Stock Exchange of India's benchmark broad based stock market index for the Indian equity market. Full form of NIFTY is National Stock Exchange Fifty. The data fields are:

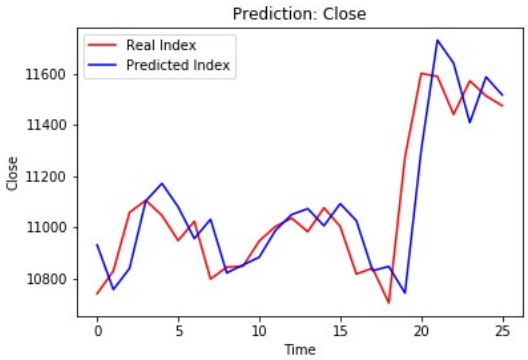
- *Open* – This is the value of the index at the start of the corresponding bussiness day.
- *Low* – This is the lowest value achieved by the index or the minima for the corresponding bussiness day.
- *High* – This is the highest value achieved by the index or the maxima for the corresponding bussiness day.
- *Close* – This is the value of the index when the stock exchange closed for the corresponding day.
- *Shares traded* – This is the number of shares traded in the day.
- *Turnover (in crores of rupees)* – This the total turnover generated (in crores of rupees) for the corresponding business day.

The values “Open”, “Low”, “High” and “Close” go up and down together. This is expected since these values are almost directly proportional to the absolute value of the Index for that day. Besides, the values “Shares Traded” and “Turnover (in crores of rupees)” go up and down with the value of the index, since, with a booming share market, the number of shares traded goes up, and so does the turnover and the absolute value of the index for the corresponding day.

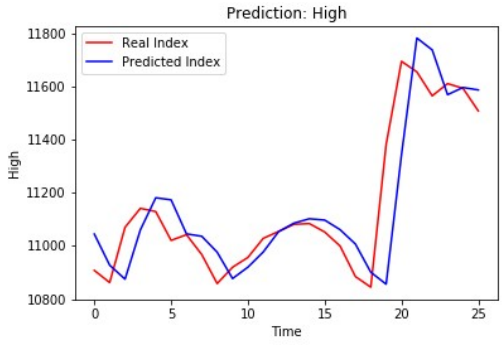
The data is similarly imported, treated and modified as before. The results are plotted in the next page.



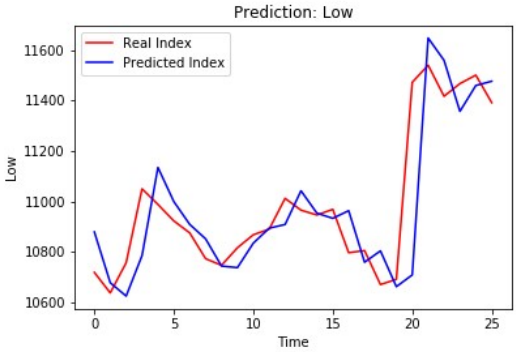
(a)



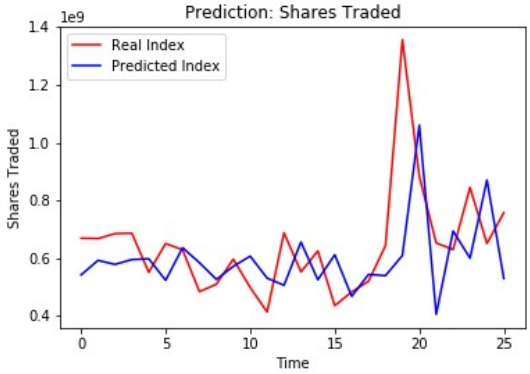
(b)



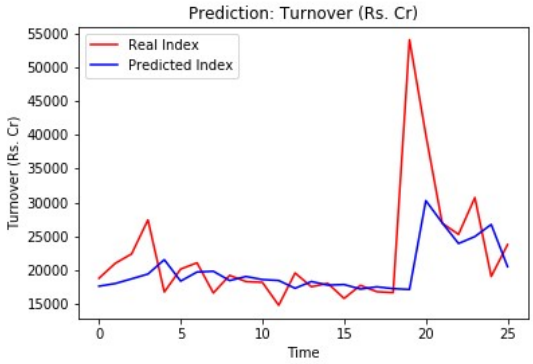
(c)



(d)



(e)



(f)

(From top to down, From left to right)
(a) Figure 6: Predicted and corresponding "Open" values of NIFTY 50 plotted
(b) Figure 7: Predicted and corresponding "Close" values of NIFTY 50 plotted
(c) Figure 8: Predicted and corresponding "High" values of NIFTY 50 plotted
(d) Figure 9: Predicted and corresponding "Low" values of NIFTY 50 plotted
(e) Figure 10: Predicted and corresponding "Shares Traded" values of NIFTY 50 plotted
(f) Figure 11: Predicted and corresponding "Turnover (Rs. Cr)" values of NIFTY 50 plotted

Data	Error (%)
Open	0.334551838883021
High	0.25558133011403
Low	0.335949563042953
Close	0.302589479706711
Shares Traded	6.19013443491534
Turnover (Rs. Cr)	7.14826612384214

Table 2: Percentage error while predicting NIFTY 50 values

As observed, the error is quite significant. The error for Shares Traded and Turnover are greater than the others. This is expected since both are related to the number of shares traded on that particular day, which, in turn, often depends entirely on the whim of the investors and of the general public, as seen during market crashes and booms.

This error is also evident when we compare the error while predicting after testing on larger datasets.

4.3.1 Anomalies In Prediction

Anomalies in the prediction is present at the huge spike in the NIFTY 50 data at when Time is nearly equal to 20. At that point of time, due to India's Finance Ministry announcing that increased public spending will be made to jump start the country's economy, the stock market sky rocketed. This huge spike or increase in data value was purely due to conscious human intervention and was certainly not a trend. However, the model tries to catch up with the values and provides nearly accurate data, albeit a little later. This brings the weakness of this model. It can not work on smaller datasets, especially if there is an intolerable and unpredictable amount of noise in the input.

5 Conclusion

Prediction of stock market using LSTM RNN is very much possible and it often leads to high accuracy prediction. However, the prediction must be done by a model which has been trained on a large dataset containing >5000 data entries to gain decent accuracy, such that market trends can be predicted in advance. The number of epochs should be made as large as possible, subject to the computational resourses at hand. The number of middle layers, if kept low, along with a large enough number of units in the middle layer, can yield very good results on any dataset. Any sudden change in the market can throw the prediction model off balance for a little while and its accuracy can plummet. However, the accuracy increases after a very short time. This fluctuation in accuracy can be altogether eliminated if a large dataset is considered.

6 Future Work

This work on prediction of stock market using LSTM RNN can be extended and made more robust. This includes:

1. Making the prediction better by employing Echo State Network Implementation (ESN).
2. Computing optimal set ups for setting up the RNN model.
3. Testing the viability of Transfer Learning, if it is possible as a generality or as a case by case basis.
4. Finding correlation between stock market fluctuations and prices of commodities, Consumer Price Index of G20 countries, environmental hazards, political tensions, etc.

7 References

[1] David M. Cutler, James M. Poterba, Lawrence H. Summers, Speculative Dynamics, *The Review of Economic Studies*, Volume 58, Issue 3, May 1991, Pages 529–546, https://doi.org/10.2307/2298010

[2] S Morris, HS Shin, Risk management with interdependent choice, *Oxford Review of Economic Policy*, Volume 15, Issue 3, September 1999, Pages 52–62, https://doi.org/10.1093/oxrep/15.3.52

[3] Sergey Perminov, *Trendocracy and Stock Market Manipulations* (2008, ISBN 978-1-4357-5244-3).

[4] Armando Bernal, Sam Fok, Rohit Pidaparthi, *Financial Market Time Series Prediction with Recurrent Neural Networks.*

[5] Yoshua Bengio, Patrice Simard, Paolo Frasconi, *IEEE transactions on neural networks*, Volume 5, Issue 2, 1994/3/2, Pages 157-166, https://scholar.google.com/scholar?oi=bibs&cluster=5202616651584911758&btnI=1&hl=en

[6] *Untersuchungen zu dynamischen neuronalen Netzen*, Josef Hochreiter, Institut für Informatik, Technische Universität München, http://people.idsia.ch/~juergen/SeppHochreiter1991ThesisAdvisorSchmidhuber.pdf

[7] *LONG SHORT-TERM MEMORY*, Neural Computation 9(8):1735{1780, 1997, Sepp Hochreiter, Fakultät für Informatik, Technische Universität München, Jürgen Schmidhuber, IDSIA, Corso Elvezia 36.