Abstract

The stock market is infamous for its unpredictability and volatility. The huge amount of data and changes of data that govern this volatility means that making accurate and profitable predictions is very hard and therefore this task is one of the most challenging tasks in time series forecasting.

So, this project's main goal is to make use of deep learning techniques to the stock market to predict the behaviour of stocks and thereby attempting to minimise investment risk. The main objective of this project is to be able to predict the end-of-day stock price for a company on the National Stock Exchange by using data of earlier days.

To achieve this, we collect historical stock data for our project using python packages. We then pre-process this data by cleaning it and adding technical indicators to the data. The prepared data is split into a train and test set and then fed to the model to train it. The process of training and testing repeats till satisfactory results are achieved. Once this has been achieved the model can then be used for its intended purpose.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Introduction

The stock market is known for its volatility, randomness and unpredictability [1]. It is a chaotic place with a massive and continuously changing stream of data which makes predicting and acting on those predictions to make a profit very hard. It is one of the most challenging tasks in time series forecasting.

Analysis of financial time series forecasting and as such, predicting future stock price movements has been an active area of research for a long period of time. There are researchers who believe that it is impossible to forecast stock movement due to the volatility and unpredictability of it. Alternatively, there are researchers who believe that it is possible to reliably predict the prices of stocks using models. There are some propositions that focus on the decomposition of time series for stock prediction that have laid a good basis for stock price prediction. [2]

Therefore, the goal of this project is to try and observe and try and understand the stock market, identify parameters and then try to bring some sense to this volatility and help mitigate the risk associated with stock trading.

1.2. Aims & Objectives

The project's main aim is to create an interface by applying machine learning techniques to the stock market in order to predict stock behaviour in an effort to minimize the risk associated with stock trading, especially for beginner traders.

To this end, many tools will be used to reach the objective of this project. Frameworks such as TensorFlow will be used to incorporate a selected neural network model trained to accurately predict the stock price of various companies on the National Stock Exchange. The application will be bundled by creating a front-end using technologies such as React.JS and Flask.

In the interest of the ever-changing nature of the stock market, the project will be able to be trained on different stocks for versatility and those models will be able to be used to predict prices for the respective stocks.

1.3. Scope

The scope of the project is to deliver a prediction model by making use of a deep neural network, more specifically making use of a Recurrent neural Network (RNN) model for making the predictions.

The stock market has for long deterred potential investors and has had a negative impact on informed investors. This has led to stigma that the stock market is only for the "elites" and not for the common man. This application, will at the very least provide a gateway to bridge the gap between the layman and help them learn and understand about investing.

The model should achieve the following:

1. The targeted price prediction will be near term. More, specifically the model should be able to predict the end-of-day stock price for the following day and to a certain extent for a specified number of days.

2. The proposed program should use varied data consisting of historical stock data and various technical indicators
3. The program should be capable of making predictions for a wide range of stocks on the National Stock Exchange

Chapter 2

Review of Literature

2.1. Domain Explanation

The project is rooted within two main domains:

- Machine Learning
- Financial Markets

Machine Learning:

Machine Learning has long been used to solve problems regarding classification and prediction as well. This has led to the formulation of many algorithms in order to solve complex machine learning problems [3]

Time series forecasting is one such important area of machine learning [2] [4]. It is important because there are so many prediction problems that involve a time component. However, while the time component adds additional information, it also makes time series problems more difficult to handle compared to many other prediction tasks. Time series data, as the name indicates, differ from other types of data in the sense that the temporal aspect is important. On a positive note, this gives us additional information that can be used when building our machine learning model — that not only the input features contain useful information, but also the changes in input/output over time.

Furthermore, with the advent of multi-layer neural networks in the form of deep learning, the ability of solving temporal problems has greatly increased. Deep learning methods offer a lot of promise for time series forecasting, such as the automatic learning of temporal dependence and the automatic handling of temporal structures like trends and seasonality [5]. Certain methods such as Recurring Neural Networks and its derivatives further enhance this capability due to their inherent abilities of handling temporal data as well as their capability of remembering long-term information.

Therefore, machine learning and more specifically, deep learning is the right avenue to take for the project since this capability of handling time series data is essential.

Financial Markets:

The financial markets are a massive area of application where machine learning techniques can be used. While Machine learning is used in the financial markets, its uptake has been slow. This is mainly because of the following reasons:

- Businesses often have completely unrealistic expectations towards machine learning and its value for their organizations.
- R&D in machine learning is costly.
- The shortage of DS/ML engineers is another major concern. The figure below illustrates an explosive growth of demand for AI and machine learning skills.
- Financial incumbents are not agile enough when it comes to updating data infrastructure.

Despite the challenges, many financial companies have made use of machine learning. The main reasons being:

- Reduced operational costs.
- Increased revenues thanks to better productivity and enhanced user experience.
- Better compliance and security.

As seen in figure 2.1., Large FinTech companies are increasingly investing in AI.

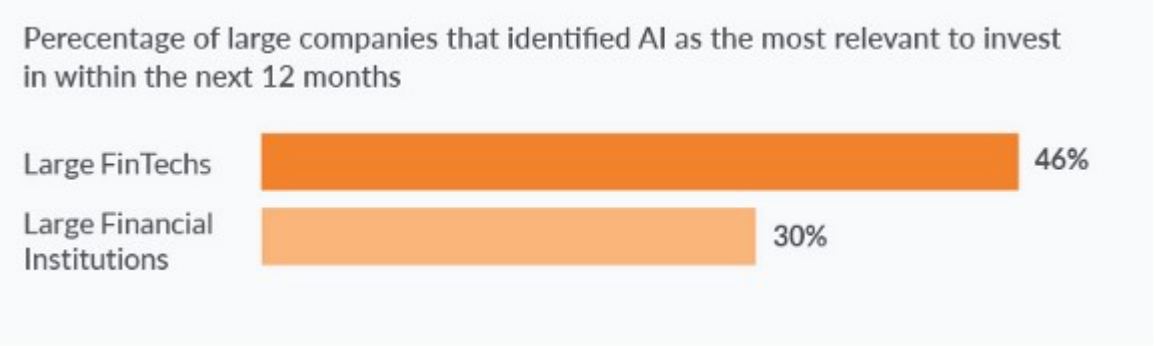


Figure 2.1. Uptake of AI in the Financial Sector

One domain where this project will be useful is in algorithmic trading. In algorithmic trading, machine learning helps to make better trading decisions. A mathematical model monitors the news and trade results in real-time and detects patterns that can force stock prices to go up or down. It can then act proactively to sell, hold, or buy stocks according to its predictions. In our case, the model predicts the end-of-day share price for a particular stock [6].

2.2. Existing Solution

There are many approaches to predicting the stock market [7].Traditional approaches to stock market analysis and stock price prediction include fundamental analysis, which consider factors such as the stock's past performance as well as the company's credibility along with statistical analysis. This method is primarily concerned with crunching numbers and identifying patterns in the variation.

The artificial NN (ANN) [8] is considered as an efficient optimization tool for predicting the time series, and also, it predicts the hidden and the unknown records. The advancements in the deep learning algorithm have produced various trading algorithms to predict the stock price changes with more accuracy.

There also exist time series models such as the Kalman filter, ARIMA, regression et cetera., that are used to analyse the stock market. These models make use of a single point and find it difficult to fit data in non-linear nature, as is the case in stock markets [9]. Furthermore, the stock market is usually noisy in nature. To deal with this, algorithms such as SVM and ANN are used. These models provide better results but are not suitable for fuzzy data [10].

Deep learning methods on the other hand extract features from a large set of raw data, which makes it potentially attractive for stock market prediction. By considering the drawbacks in the existing stock market prediction techniques, this project proposes a different method to stock price predictions.

2.3. Hardware & Software Requirements

The following are the recommended system requirements:

- Processor: Intel Core i5-7200U
- RAM: 4GB
- OS: Windows(64 Bit)
- Stable Internet Connection

The necessary recommended python version and packages are given below:

- Python: 3.6+
- TensorFlow: 2.2.0
- MatPlotLib: 3.2.0
- Pandas: 0.25.3
- Numpy: 1.18.1
- NSEpy: 0.8
- Nsetools: 1.0.11
- TA-Lib: 0.4.17
- Flask: 1.1.2
- React.js: 17.0.1

Chapter 3

Analysis

3.1. Functional Requirements

- The user should be able to enter the symbol of the stock required.
- The Application should allow the user to search the current stock price of a stock.
- The system should retrieve stock data of the chosen stock, calculate parameters and train the model on that data.
- Such a model should be created for every new stock
- A trained model can be chosen by passing the stock symbol. The model should be able to predict the price for the next day for that stock
- Multi-day predictions should be possible

3.2. Non-Functional Requirements

- Reliability: The system will consistently and reliably accomplish its goals
- Scalability: The system will be scalable for large scale use
- Performance: The designed project will predict the next day closing price of stock with great accuracy
- Efficient Response Time
- The accuracy of multiday predictions will be within bounds

3.3. Proposed System

The proposed system is an application to serve to the user, the prediction models as well as other features to help the user in their endeavour. Figure 3.1. provides an overview of the features that will be included in the application.

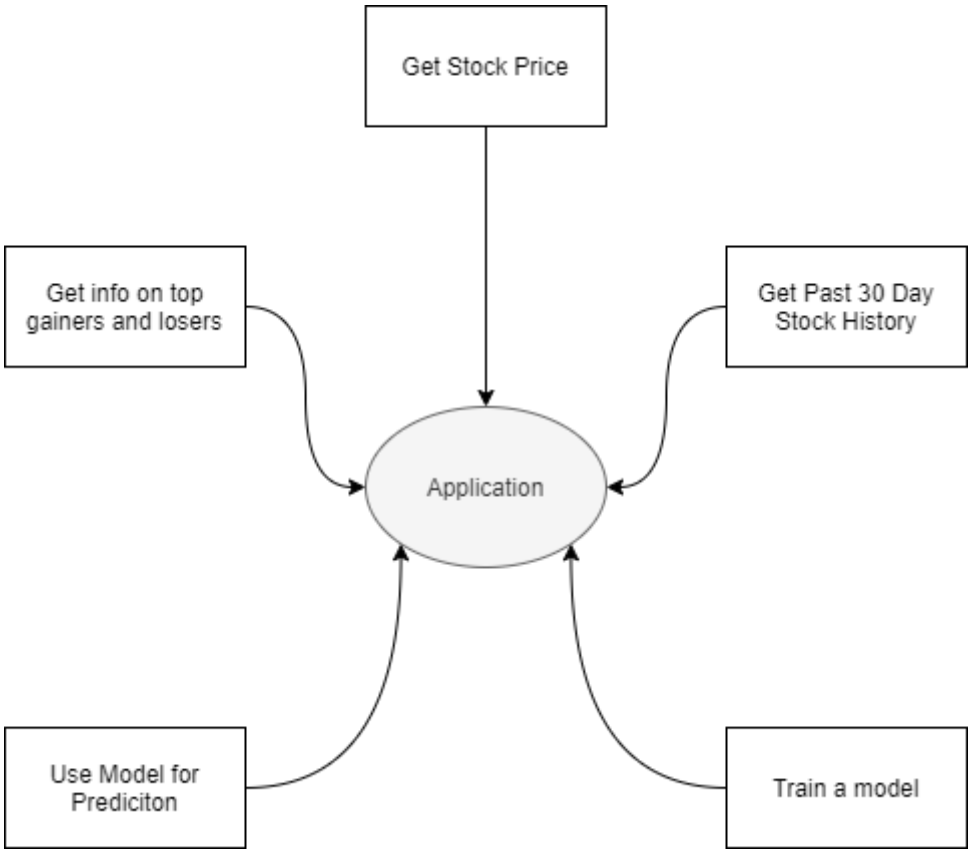


Figure 3.1. Proposed System

3.3.1. Flow of Model

The proposed system is a CNN-LSTM neural network that is trained on stock data of the past 15 years and thereafter can be used to predict the end-of-day closing price of a stock for the following day.

For a particular stock, the data fetched and prepared by the program. The program is then is then trained on that data and the model is saved.

In figure 3.2. we can see an overview of the proposed system.

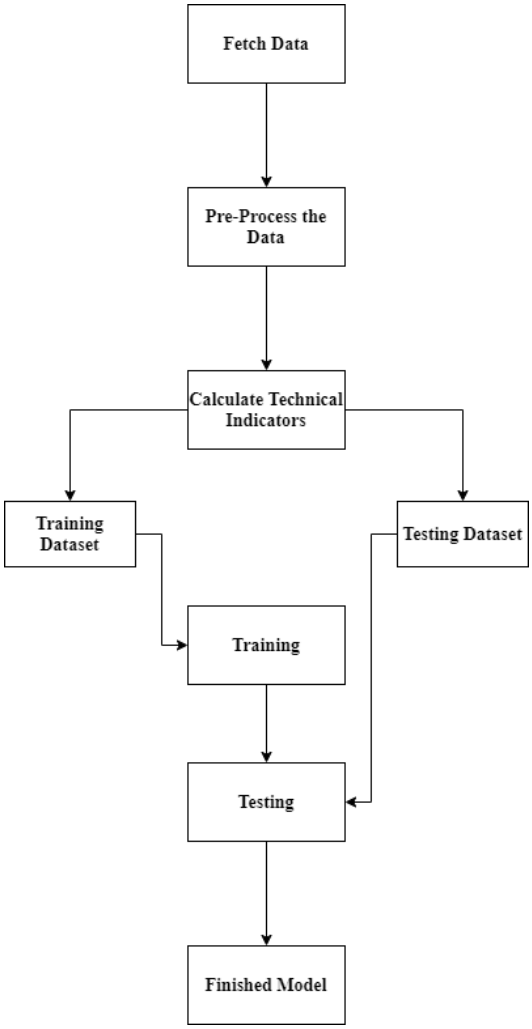


Figure 3.2. Flow of Model

3.3.2. Prediction

To use a model for predictions, we ask the users, the number of days he wants to predict for and the model which he wants to use. Figure 3.3. shows the flow of the prediction system. The model will predict the High Price, Low Price, Open Price and Closing Price.

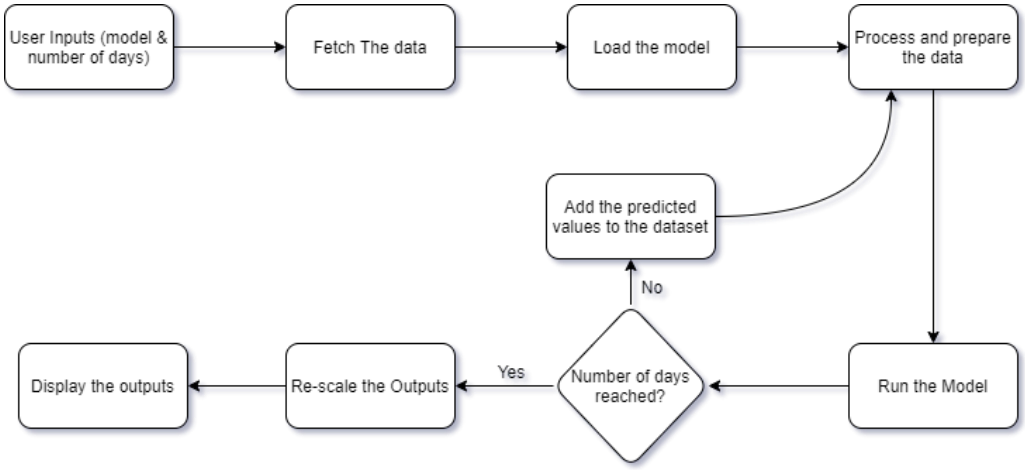


Figure 3.3. Flow of Prediction System

Chapter 4

Design

4.1. Design Consideration

For the design of our model, we have incorporated an MVC architecture to serve up our deep learning model to the user.

MVC Architecture

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects. Figure 4.1. shows the working of an MVC architecture.

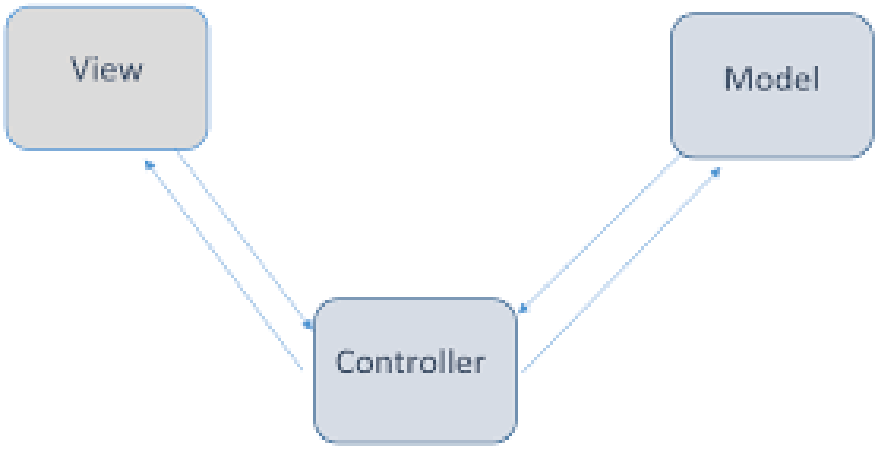


Figure 4.1. MVC Architecture

MVC Architecture has 3 components:

- **Model:** The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View

and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

- **View:** The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

- **Controller:** Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

4.2. Design Details

Using the MVC architecture earlier, we form a connect between the python code and the user interface for the user. It consists of the following components:

- React.js: The entirety of the UI front-end is based on React.js library. React is a declarative Javascript library that finds extensive use in web development for the creation of various client-side modules. React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called “components” [11]. React provides a greater deal of flexibility and applications created through React are more compact as compared to development using Angular.js. React is an incredibly modern web development framework. The various advantages provided by React include its flexibility, server-side communication capabilities, multiple asynchronous functions, and generators provided along with its compatibility with several javascript libraries [12]. To make the application more appealing and to make it compatible for all kinds of devices, we have followed the Maerial Design Guidelines laid out by Google [13].

- Flask: Flask is an micro framework offering basic features of web app. This framework has no dependencies on external libraries. The framework offers extensions for form validation, object-relational mappers, open authentication systems, uploading mechanism, and several other tools. We use flask to be the "controller" or the bridge between the UI and the python backend. It acts as a backend API server through which we retrieve information from the python models by using GET api calls in our frontend.

- Python: Python makes up the crux of our project. Using python and its associated libraries we create the software for the model and all the other required features.

- TensorFlow: It is a machine learning system that operates on a large scale. TensorFlow uses dataflow graphs to represent computation, shared state, and the operations that mutate that state. It maps the nodes of a dataflow graph across many machines in a cluster, and within a machine across multiple computational devices, including multicore CPUs, general-purpose GPUs, and custom-designed ASICs known as Tensor Processing Units (TPUs). TensorFlow enables developers to experiment with novel optimizations and training algorithms. TensorFlow supports a variety of applications, with a focus on training and inference on deep neural networks [14].

4.3. GUI Details

Figure 4.2. shows the design of our application.

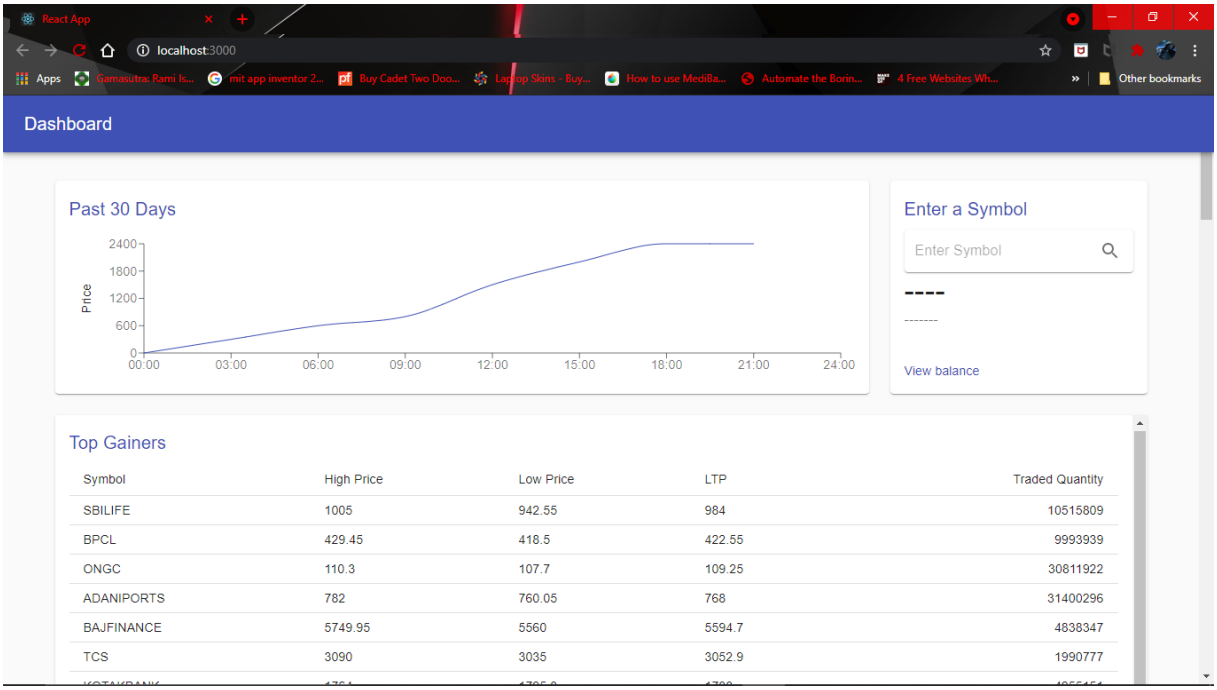


Figure 4.2. GUI Overview

The GUI consists of 4 components:

- The first component is the “Stock Search” component (Figure 4.3.). Here, the user can search for the current price of a stock based on its symbol and can get the current stock price and the past 30 day closing price history of the stock.

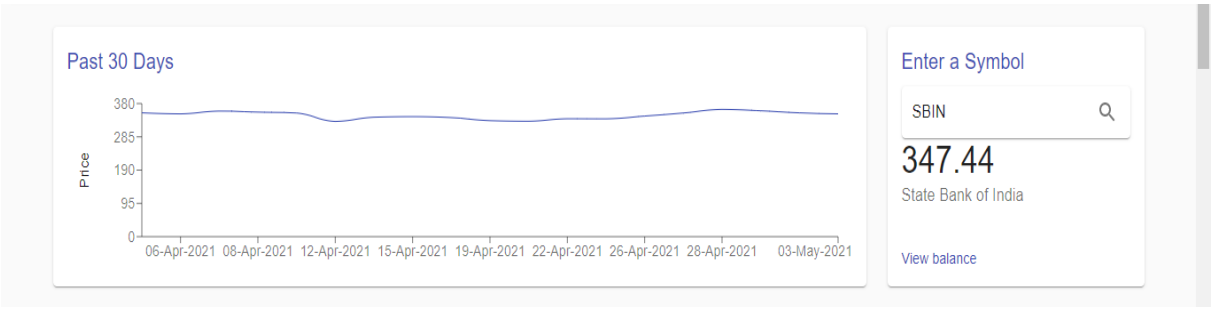
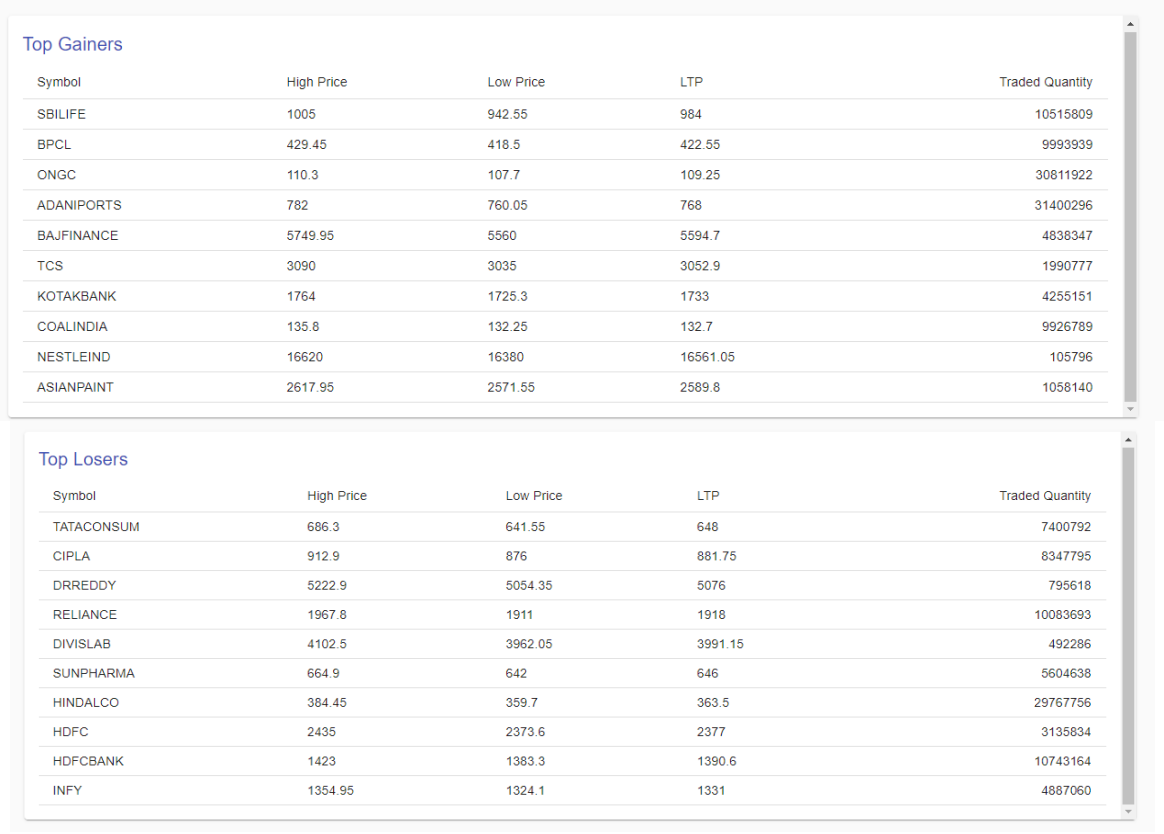


Figure 4.3. Stock Search Component

- The second component is the Gives us the Top Gainers and Losers (Figure 4.4.). It updates every 10 seconds so that the user is always akin to the top performers of the day.



Top Gainers

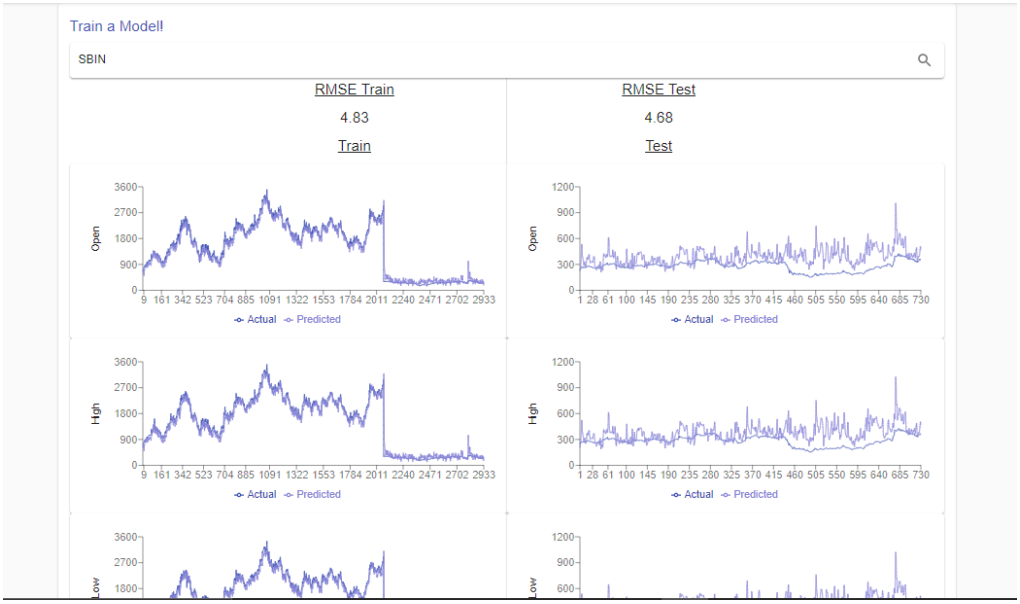
Symbol	High Price	Low Price	LTP	Traded Quantity
SBILIFE	1005	942.55	984	10515809
BPCL	429.45	418.5	422.55	9993939
ONGC	110.3	107.7	109.25	30811922
ADANIPORTS	782	760.05	768	31400296
BAJFINANCE	5749.95	5560	5594.7	4838347
TCS	3090	3035	3052.9	1990777
KOTAKBANK	1764	1725.3	1733	4255151
COALINDIA	135.8	132.25	132.7	9926789
NESTLEIND	16620	16380	16561.05	105796
ASIANPAINT	2617.95	2571.55	2589.8	1058140

Top Losers

Symbol	High Price	Low Price	LTP	Traded Quantity
TATACONSUM	686.3	641.55	648	7400792
CIPLA	912.9	876	881.75	8347795
DRREDDY	5222.9	5054.35	5076	795618
RELIANCE	1967.8	1911	1918	10083693
DIVISLAB	4102.5	3962.05	3991.15	492286
SUNPHARMA	664.9	642	646	5604638
HINDALCO	384.45	359.7	363.5	29767756
HDFC	2435	2373.6	2377	3135834
HDFCBANK	1423	1383.3	1390.6	10743164
INFY	1354.95	1324.1	1331	4887060

Figure 4.4. Top Gainers and Losers

- The third component is the Training component (Figure 4.5.). The user inputs the symbol of the stock and then a request is made to the API. The API then calls the training function to fetch the data and train the model. The application then displays the error of the trained model and displays the graph of comparison of the actual and predicted values of the Open price, Close Price, High Price and Low price for the training and testing dataset.



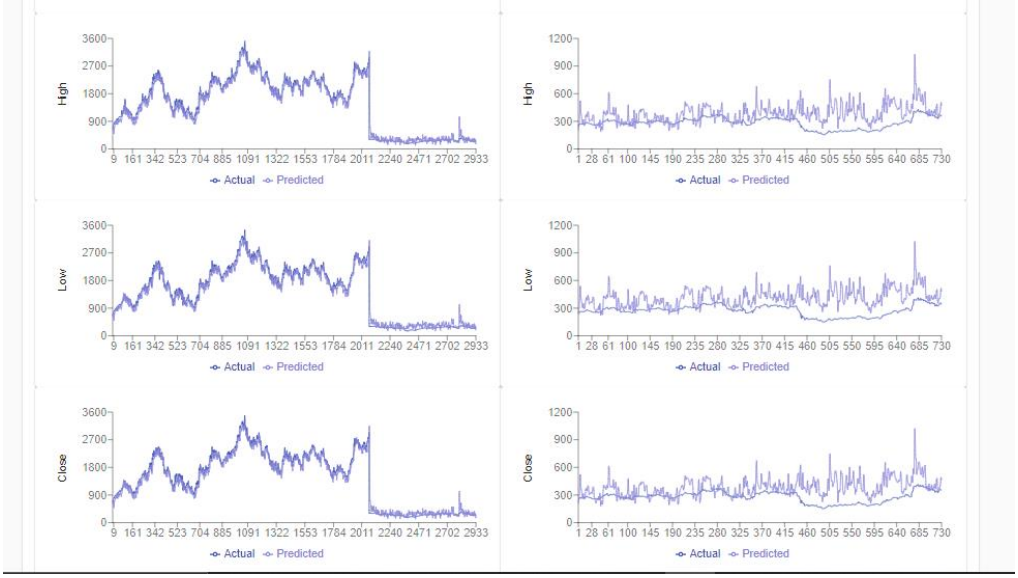


Figure 4.5. Training Component

- The last component is the prediction component (Figure 4.6.). Here the user inputs the number of days for which he wants to predict and the model which he wants to use. The API call is made and the prediction model is run. The UI then displays the closing price as well as a graph with the predicted price of every day for the number of days mentioned.

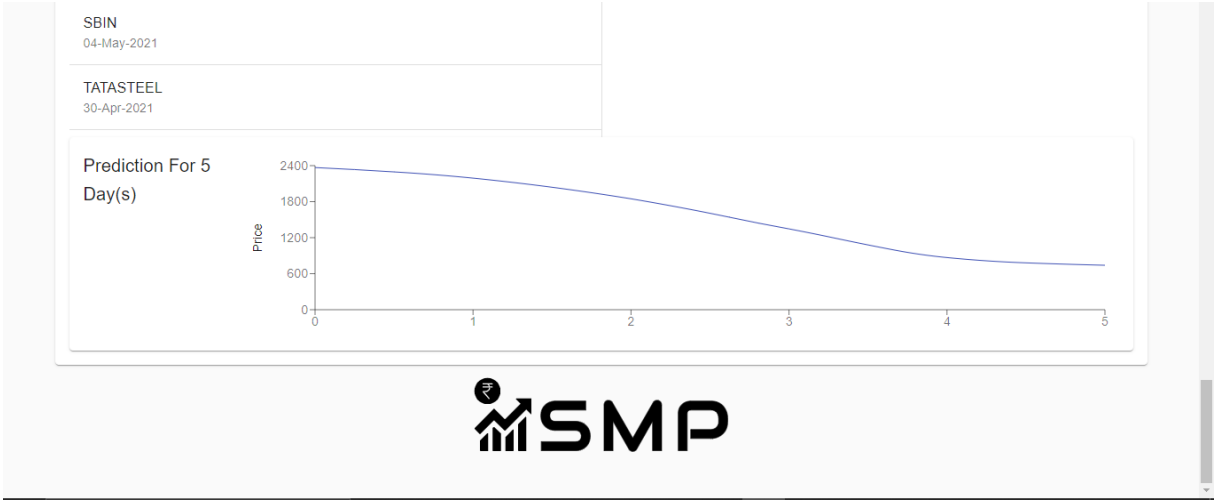


Figure 4.6. Prediction Model

Chapter 5

Implementation

5.1. Dataset Description

The model requires diverse data in order to train the model. For this reason, we try to incorporate various types of data in the project. For each stock that the model is trained on, we use 12 parameters. The data used by the project comes from two places:

- Historical Stock Data
- Technical Indicators

5.1.1. Historical Stock Data:

Historical stock data is the data we directly fetch from various API. This consists of the various stock data that is publicly available. This consists of:

- Opening Price
- Closing Price
- The day’s low and high price
- Difference in Prices

5.1.2. Technical Indicators:

Technical indicators are heuristic or pattern-based signals produced by the price, volume, and/or open interest of a security or contract used by traders who follow technical analysis. By analyzing historical data, technical analysts use indicators to predict future price movements [15]. They give an idea of the momentum of the stock, trend, strength of the trend and volume shift. Indicators used in the project are:

- Bollinger Bands: Bollinger Bands consist of three lines. The middle band is a simple moving average (generally 20 periods) of the typical price (TP). The upper and lower bands are *F* standard deviations (generally 2) above and below the middle band. The bands widen and narrow when the volatility of the price is higher or lower, respectively.

  Bollinger Bands do not, in themselves, generate buy or sell signals; they are an indicator of overbought or oversold conditions. When the price is near the upper or lower band it indicates that a reversal may be imminent. The middle band becomes a support or resistance level. The upper and lower bands can also be interpreted as price targets. When the price bounces off of the lower band and crosses the middle band, then the upper band becomes the price target [16] [17].

$$TP = \frac{high + low + close}{3}$$
$$MidBand = SimpleMovingAverage(TP)$$
$$UpperBand = MidBand + F \times \sigma(TP)$$
$$LowerBand = MidBand - F \times \sigma(TP)$$

- Relative Strength Index (RSI): The relative strength index was created by J. Welles Wilder Jr. in the late 1970s; his "New Concepts in Trading Systems" (1978) is now an investment-lit classic. On a chart, RSI assigns stocks a value between 0 and 100. Once these numbers are charted, analysts compare them against other factors, such as the undersold or underbought values. To reach the best evaluation, experts generally chart the RSI on a daily time frame rather than hourly. However, sometimes shorter hourly periods are charted to indicate whether it is a good idea to make a short-term asset purchase [16] [17] [18].

$$RSI = 100 - \left(\frac{100}{1 + RS}\right)$$

$$RS = \frac{Average\ of\ x\ days'up\ closes}{Average\ of\ x\ days'down\ closes}$$

- Stochastic Oscillator: A stochastic oscillator is a momentum indicator comparing a particular closing price of a security to a range of its prices over a certain period of time. The sensitivity of the oscillator to market movements is reducible by adjusting that time period or by taking a moving average of the result. It is used to generate overbought and oversold trading signals, utilizing a 0–100 bounded range of values [17] [19].

$$\%K = \frac{close - LowestLow_{last\ n\ periods}}{HighestHigh_{last\ n\ periods} - LowestLow_{last\ n\ periods}}$$

- Moving Average Convergence Divergence (MACD): It is the difference between two Exponential Moving Averages. The Signal line is an Exponential Moving Average of the MACD. The MACD signals trend changes and indicates the start of new trend direction. High values indicate overbought conditions, low values indicate oversold conditions. Divergence with the price indicates an end to the current trend, especially if the MACD is at extreme high or low values. When the MACD line crosses above the signal line a buy signal is generated. When the MACD crosses below the signal line a sell signal is generated. To confirm the signal, the MACD should be above zero for a buy, and below zero for a sell [17] [16].

$$shortEMA = 0.15 \times price + 0.85 \times shortEMA_{-1}$$
$$longEMA = 0.075 \times price + 0.925 \times longEMA_{-1}$$

- Exponential Moving Average: The Exponential Moving Average is a staple of technical analysis and is used in countless technical indicators. In a Simple Moving Average, each value in the time period carries equal weight, and values outside of the time period are not included in the average. However, the Exponential Moving Average is a cumulative calculation, including all data. Past values have a diminishing contribution to the average, while more recent values have a greater contribution. This method allows the moving average to be more responsive to changes in the data [16] [17].

$$K = \frac{2}{n+1}$$

$$EMA = EMA_{-1} + K \times (input - EMA_{-1})$$

- Average Directional Index: The average directional index (ADX) is a technical analysis indicator used by some traders to determine the strength of a trend. The trend can be either up or down, and this is shown by two accompanying indicators, the negative directional indicator (-DI) and the positive directional indicator (+DI) [17] [20].

$$+DI = \left(\frac{Smoothed + DM}{ATR}\right) \times 100$$

$$-DI = \left(\frac{Smoothed - DM}{ATR}\right) \times 100$$

$$DX = \left(\frac{|+DI - -DI|}{|+DI + -DI|}\right)$$

$$ADX = \frac{(Prior\ ADX \times 13) + Current\ ADX}{14}$$

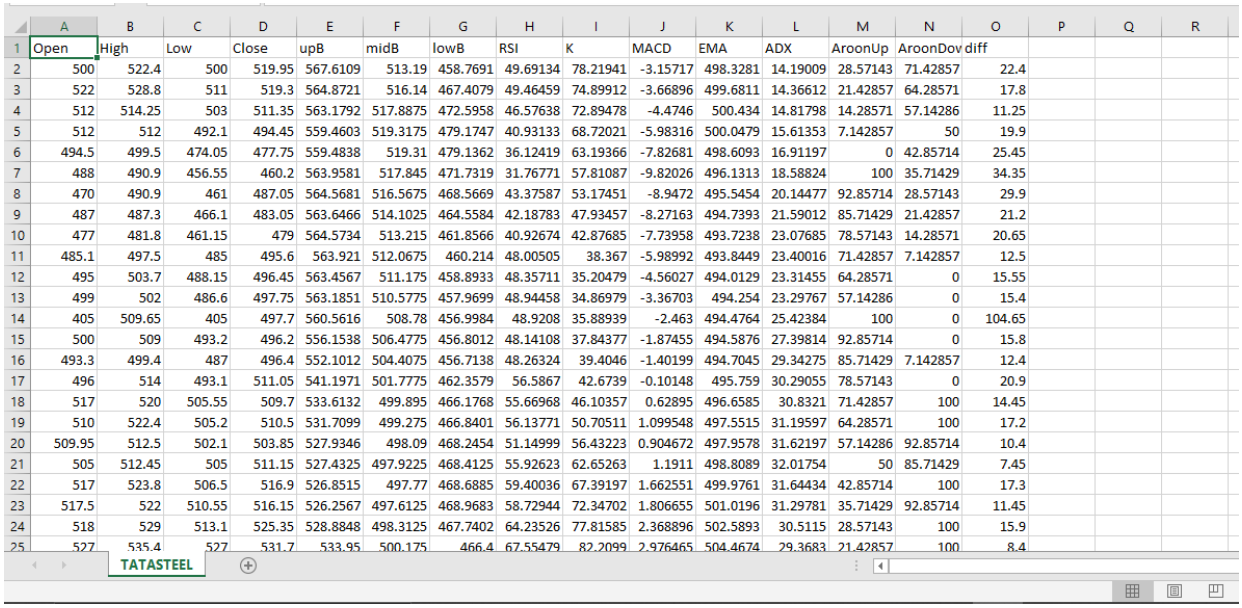
- Aroon Oscillator: The Aroon Oscillator is a trend-following indicator that uses aspects of the Aroon Indicator (Aroon Up and Aroon Down) to gauge the strength of a current trend and the likelihood that it will continue. Readings above zero indicate that an uptrend is present, while readings below zero indicate that a downtrend is present. Traders watch for zero-line crossovers to signal potential trend changes. They also watch for big moves, above 50 or below -50 to signal strong price moves [17] [21].

$$Aroon\ Oscillator = Aroon\ Up - Aroon\ Down$$

$$Aroon\ Up = 100 \times \frac{(25 - Periods\ Since\ 25\ Period\ High)}{25}$$

$$Aroon\ Down = 100 \times \frac{(25 - Periods\ Since\ 25\ Period\ Low)}{25}$$

Figure 5.1. shows us a snapshot of the data used for the Tata Steel (TATASTEEL) stock:



	Open	High	Low	Close	upB	midB	lowB	RSI	K	MACD	EMA	ADX	AroonUp	AroonDov	diff
2	500	522.4	500	519.95	567.6109	513.19	458.7691	49.69134	78.21941	-3.15717	498.3281	14.19009	28.57143	71.42857	22.4
3	522	528.8	511	519.3	564.8721	516.14	467.4079	49.46459	74.89912	-3.66896	499.6811	14.36612	21.42857	64.28571	17.8
4	512	514.25	503	511.35	563.1792	517.8875	472.5958	46.57638	72.89478	-4.4746	500.434	14.81798	14.28571	57.14286	11.25
5	512	512	492.1	494.45	559.4603	519.3175	479.1747	40.93133	68.72021	-5.98316	500.0479	15.61353	7.142857	50	19.9
6	494.5	499.5	474.05	477.75	559.4838	519.31	479.1362	36.12419	63.19366	-7.82681	498.6093	16.91197	0	42.85714	25.45
7	488	490.9	456.55	460.2	563.9581	517.845	471.7319	31.76771	57.81087	-9.82026	496.1313	18.58824	100	35.71429	34.35
8	470	490.9	461	487.05	564.5681	516.5675	468.5669	43.37587	53.17451	-8.9472	495.5454	20.14477	92.85714	28.57143	29.9
9	487	487.3	466.1	483.05	563.6466	514.1025	464.5584	42.18783	47.93457	-8.27163	494.7393	21.59012	85.71429	21.42857	21.2
10	477	481.8	461.15	479	564.5734	513.215	461.8566	40.92674	42.87685	-7.73958	493.7238	23.07685	78.57143	14.28571	20.65
11	485.1	497.5	485	495.6	563.921	512.0675	460.214	48.00505	38.367	-5.98992	493.8449	23.40016	71.42857	7.142857	12.5
12	495	503.7	488.15	496.45	563.4567	511.175	458.8933	48.35711	35.20479	-4.56027	494.0129	23.31455	64.28571	0	15.55
13	499	502	486.6	497.75	563.1851	510.5775	457.9699	48.94458	34.86979	-3.36703	494.254	23.29767	57.14286	0	15.4
14	405	509.65	405	497.7	560.5616	508.78	456.9984	48.9208	35.88939	-2.463	494.4764	25.42384	100	0	104.65
15	500	509	493.2	496.2	556.1538	506.4775	456.8012	48.14108	37.84377	-1.87455	494.5876	27.39814	92.85714	0	15.8
16	493.3	499.4	487	496.4	552.1012	504.4075	456.7138	48.26324	39.4046	-1.40199	494.7045	29.34275	85.71429	7.142857	12.4
17	496	514	493.1	511.05	541.1971	501.7775	462.3579	56.5867	42.6739	-0.10148	495.759	30.29055	78.57143	0	20.9
18	517	520	505.55	509.7	533.6132	499.895	466.1768	55.66968	46.10357	0.62895	496.6585	30.8321	71.42857	100	14.45
19	510	522.4	505.2	510.5	531.7099	499.275	466.8401	56.13771	50.70511	1.099548	497.5515	31.19597	64.28571	100	17.2
20	509.95	512.5	502.1	503.85	527.9346	498.09	468.2454	51.14999	56.43223	0.904672	497.9578	31.62197	57.14286	92.85714	10.4
21	505	512.45	505	511.15	527.4325	497.9225	468.4125	55.92623	62.65263	1.1911	498.8089	32.01754	50	85.71429	7.45
22	517	523.8	506.5	516.9	526.8515	497.77	468.6885	59.40036	67.39197	1.662551	499.9761	31.64434	42.85714	100	17.3
23	517.5	522	510.55	516.15	526.2567	497.6125	468.9683	58.72944	72.34702	1.806655	501.0196	31.29781	35.71429	92.85714	11.45
24	518	529	513.1	525.35	528.8848	498.3125	467.7402	64.23526	77.81585	2.368896	502.5893	30.5115	28.57143	100	15.9
25	527	535.4	527	531.7	533.95	500.175	466.4	67.55479	82.2099	2.976465	504.4674	29.3683	21.42857	100	8.4

TATASTEEL

Figure 5.1. Dataset Example

5.2. Neural Network Architecture

As stated in the proposed system. The network used is a CNN-LSTM architecture. The CNN Long Short-Term Memory Network or CNN-LSTM for short is an LSTM architecture specifically designed for sequence prediction problems with spatial inputs, like images or videos

This is a two-fold model with two parts, i.e. the Convolutional Neural Network (CNN) and the Long Short-Term Memory (LSTM) network. The CNN Model for feature extraction and the LSTM Model for interpreting the features across time steps. Figure 5.2. represents the flow of a CNN-LSTM model.

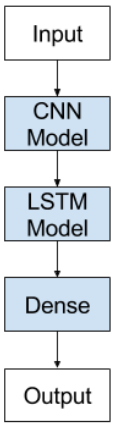


Figure 5.2. CNN-LSTM Architecture

5.2.1. Long Short-Term Memory (LSTM)

The main crux of the model is the LSTM network [22]. The LSTM network is a type of Recurrent Neural Network (RNN) that solves some of the problems that plague the original RNN model. The biggest of these problems is the problem of long-term dependencies i.e. an LSTM model can remember information from far back in time to help with predicting the output at the current time. This is due to the LSTM structure which has 3 activation gates, namely the 'Input Gate', 'Output Gate' and 'Forget Gate' that can selectively remember and forget information that is irrelevant [23].

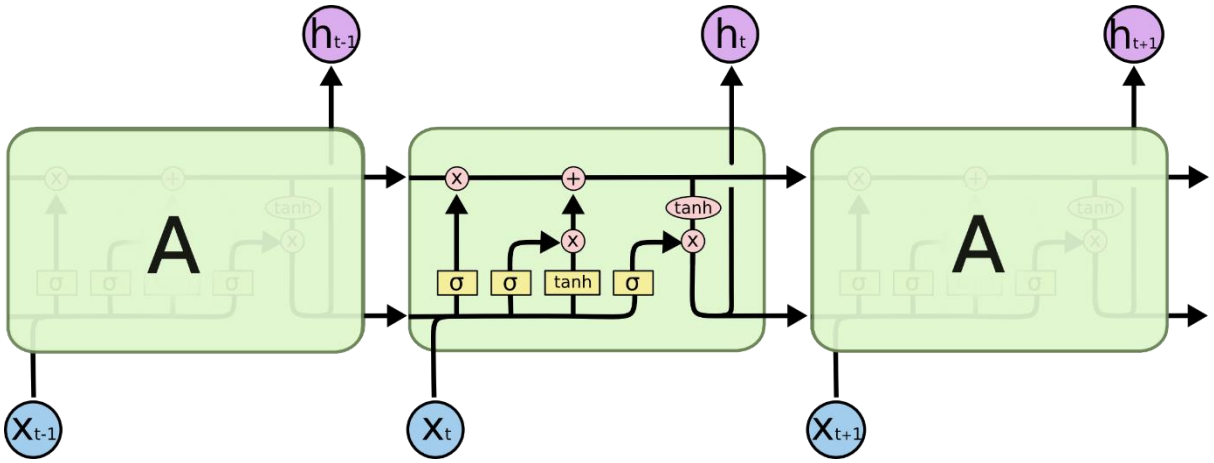


Figure 5.3. LSTM Chain

5.2.2. Convolutional Neural Network (CNN)

A Convolutional Neural Network is a network made up of multiple layers of artificial neurons which are basically mathematical functions that calculate the weighted sum of multiple inputs and outputs as an activation value. The behavior of each neuron is defined by its weights. When fed with the pixel values, the artificial neurons of a CNN pick out various visual features.

When you input an image into a ConvNet, each of its layers generates several activation maps. Activation maps highlight the relevant features of the image. Each of the neurons takes a patch of pixels as input, multiplies their color values by its weights, sums them up, and runs them through the activation function.

The first (or bottom) layer of the CNN usually detects basic features such as horizontal, vertical, and diagonal edges. The output of the first layer is fed as input of the next layer, which extracts more complex features, such as corners and combinations of edges. As you move deeper into the convolutional neural network, the layers start detecting higher-level features such as objects, faces, and more. Figure 5.4. gives us an example of a CNN [24].

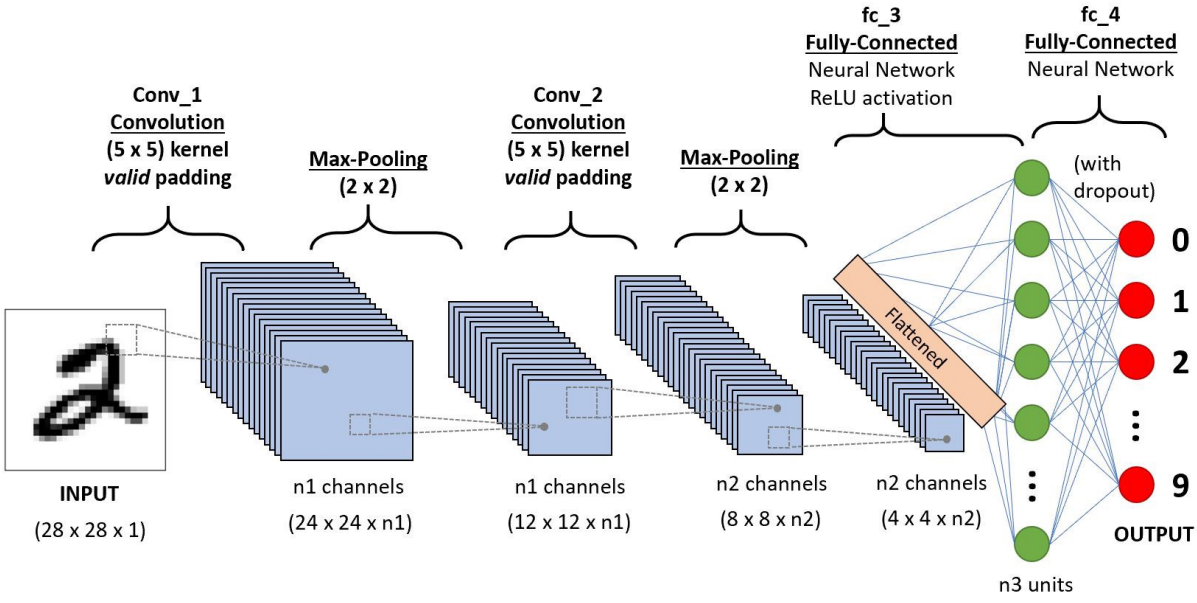


Figure 5.4. Example of a CNN

5.3. Configuration of Model

The configuration of our model is as follows:

Table 5.1. Network Detail of CNN-LSTM

Layer	Configuration
Convolutional * 2	Filter: 18
	Kernal Size: 3,4
	Stride: 1
	Padding: No
LSTM	Units: 100
	Dropout: 0.4
	Kernel Regularizer: L1(0.001)
LSTM	Units: 50
	Dropout: 0.4
	Activity Regularizer: L1(0.001)
LSTM	Units: 25
	Dropout: 0.2
	Activity Regularizer: L1(0.001)
Fully Connected	Units: 4

5.4. Methodology

The system consists of various steps from start to finish. These steps are:

- Data Retrieval
- Data Processing

- Training the Model
- Testing the Model
- Prediction

5.4.1. Data Retrieval

Rather than using a static dataset, for this project we dynamically fetch a dataset for each new stock. This means that we save on disk space since we do not have to keep a repository of stock information for many stocks.

The user will input the symbol of the stock and then the program will retrieve the historical data of that stock. This is done using the NSEpy package which retrieves data in the form of a pandas dataframe.

5.4.2. Data Processing

In the data processing step, we do many functions such as cleaning the data, calculating the technical indicators, normalization and splitting the data into the train and test set. The flowchart below gives an overview of the flow of the data processing step.

To calculate the technical indicators, we use the TA-Lib package that provides us with the functionality of many indicators used for technical analysis.

Since the model requires a large amount of training data, 85% of the dataset is used as the training dataset and 15% of it is used as the testing dataset. In Figure 5.5. we can see the flowchart that represents the data processing phase.

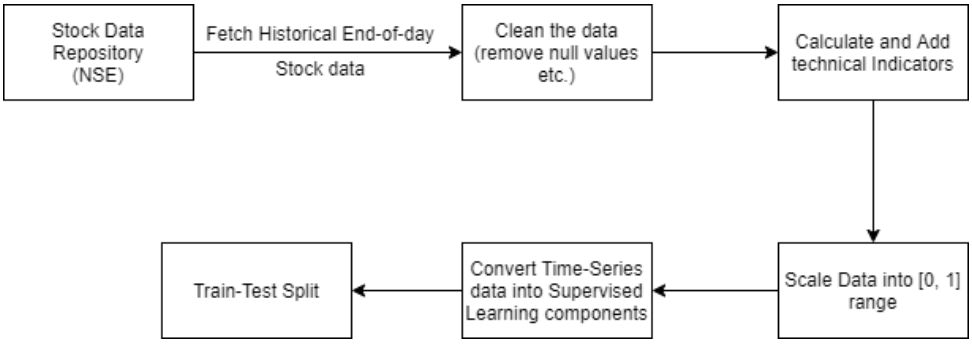


Figure 5.5. Data Processing

5.4.3. Training the Data

After preparing the data, we store it in an excel file. This data is also then fed to the model to and the model is trained on this data. We train the model for 250 iterations. For the purpose of training, we first need to set two parameters: a loss function and an optimizer.

Loss Function

Deep learning models are usually trained under stochastic gradient descent. Therefore, as part of the optimization process, the error for the current state has to be estimated repeatedly. For this purpose, we require a loss function to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation. In our project, we use the Mean Squared Error(MSE) loss metric.

Mean squared error (MSE) is the most commonly used loss function for regression. The loss is the mean overseen data of the squared differences between true and predicted values, or writing it as a formula.

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y - \hat{y}_i)^2$$

Optimizer

Deep learning is a highly iterative process requiring a lot of hyperparameter tuning and variations. Therefore, it is paramount that we have a way to train the model quicker without penalizing the cost. This job is done by something known as “optimizers” [25]. For our application, we make use of the AdaDelta [26] optimization algorithm.

Adadelta [26] optimization is a stochastic gradient descent method that is based on adaptive learning rate per dimension to address two drawbacks:

- The continual decay of learning rates throughout training
- The need for a manually selected global learning rate

Adadelta is a more robust extension of Adagrad that adapts learning rates based on a moving window of gradient updates, instead of accumulating all past gradients. This way, Adadelta continues learning even when many updates have been done. Compared to Adagrad, in the original version of Adadelta you don't have to set an initial learning rate.

5.4.4. Testing the Data

Once the model is trained, we assess the predicted values to the actual values and if the performance is not satisfactory, we go back and adjust the hyperparameters and train the model again. This process repeats till the performance is satisfactory

5.4.5. Prediction

After the training and testing of the model is complete, we then save the model. This model can then be used by passing the symbol of the stock to the system. On passing the symbol, the program should return the Opening Price, Closing Price, High of the day and low of the day. Since all the parameters exhibit a similar error, we are only plotting the closing prices

below. The errors mentioned in Figure 5.2. are the average errors of all the 4 predicted parameters.

5.5. Results

Based on our model, we have tested a few stocks. Namely,

- Tata Steel (TATASTEEL)
- Reliance Industries Ltd (RELIANCE)
- State Bank of India (SBIN)

We have chosen two parameters to evaluate our models. (All multi-day predctions use a simple error formula):

- Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Predicted_i - Observed_i)^2}{n}}$$

- Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|$$

Table 5.2. shows us the accuracies of the trained model.

Table 5.2. Accuracy Table

Symbol Accuracy Measure	SBIN	TATASTEEL	RELIANCE
RMSE (Train)	0.023	0.015	0.028
RMSE (Test)	0.031	0.027	0.059
MAE (Train)	0.024	0.014	0.015

MAE(Test)	0.029	0.019	0.052

5.5.1. Tata Steel

Below we can see the result of the model on the Training Set (Figure 5.6), Testing Set (Figure 5.7.) and performance on a small sample of multiple day predictions (Figure 5.8.).

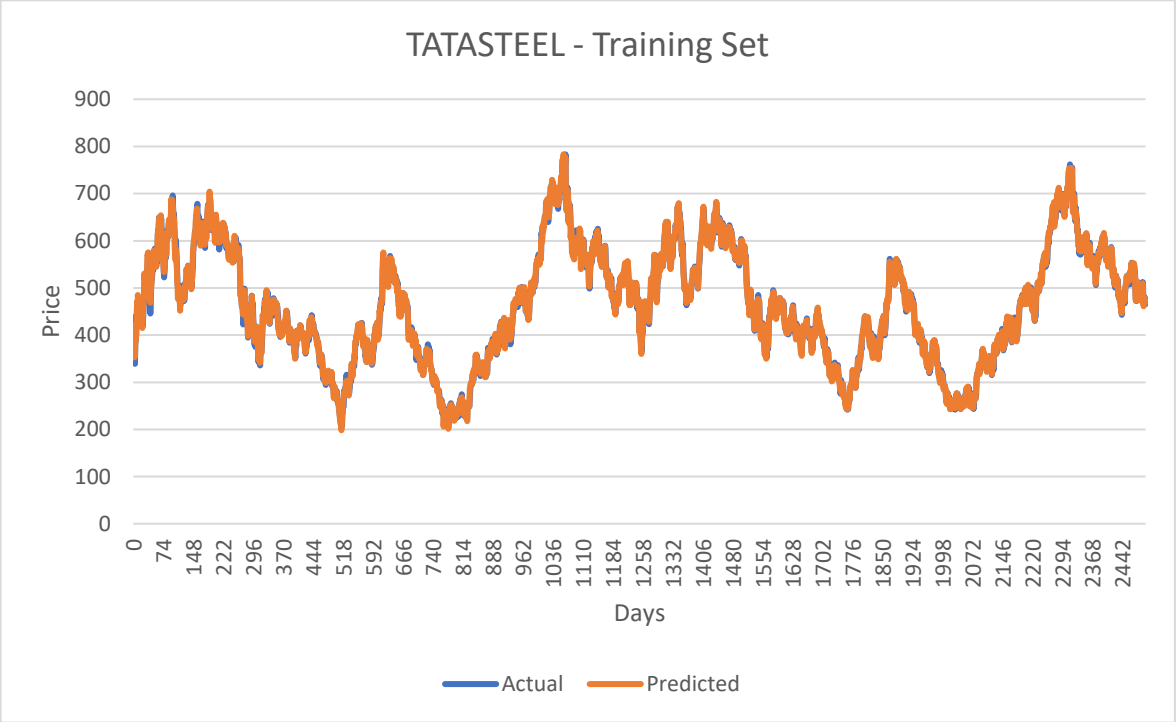


Figure 5.6. TATASTEEL – Train

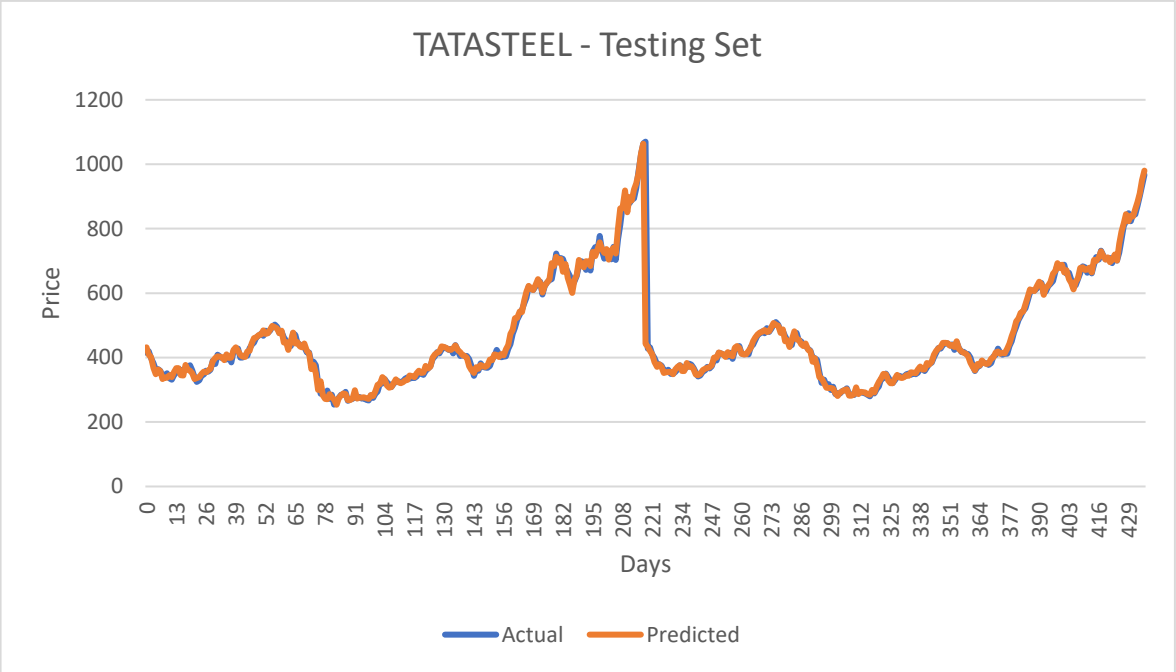


Figure 5.7. TATASTEEL – Test

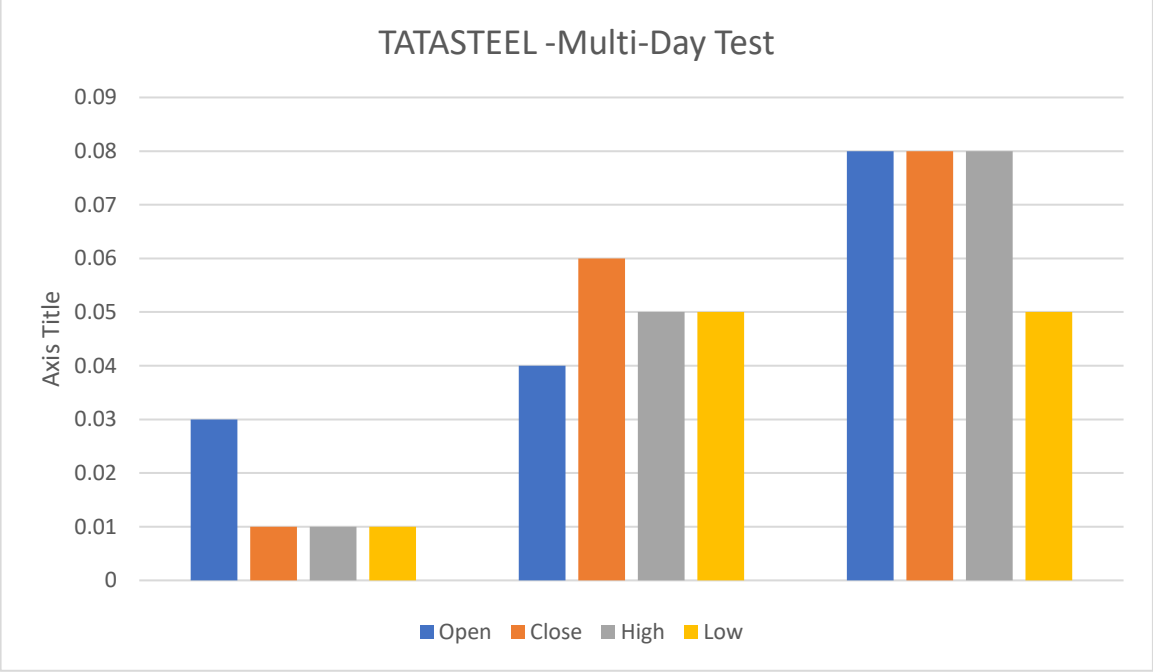


Figure 5.8. TATASTEEL – Multi-Day Prediction

5.5.2. State Bank of India

Below we can see the result of the model for the State Bank of India on the Training Set (Figure 5.9), Testing Set (Figure 5.10.) and performance on a small sample of multiple day predictions (Figure 5.11.).

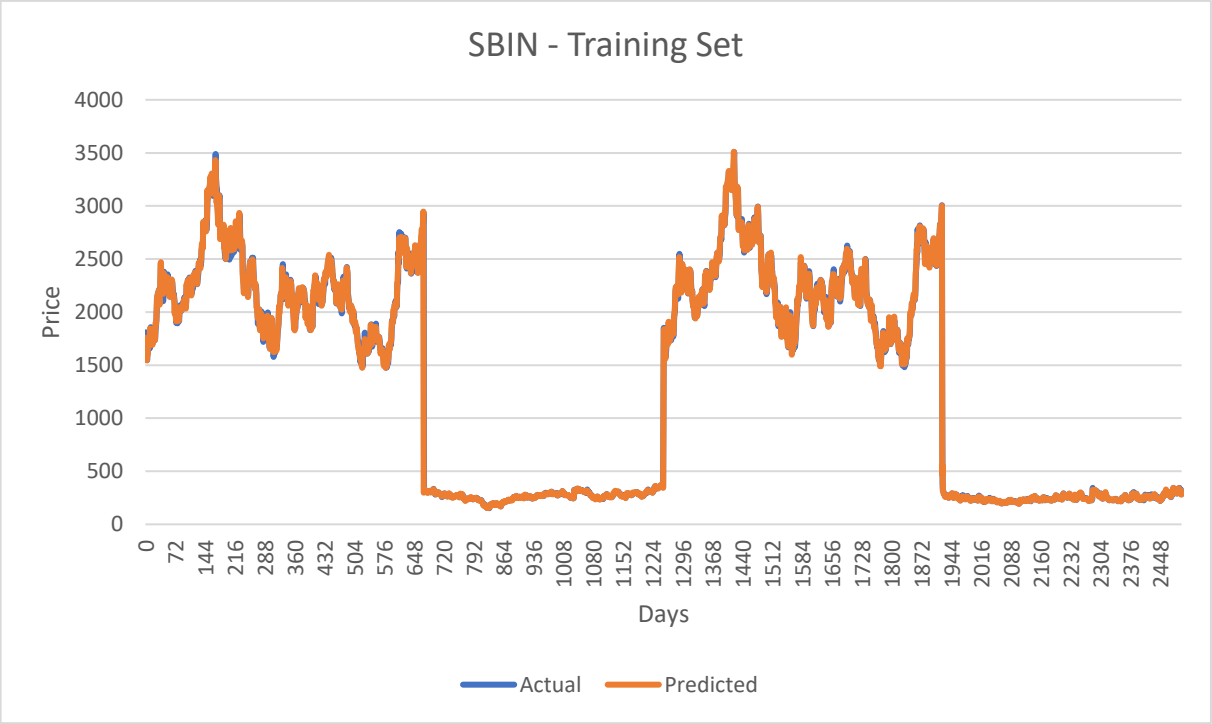


Figure 5.9. SBIN – Train

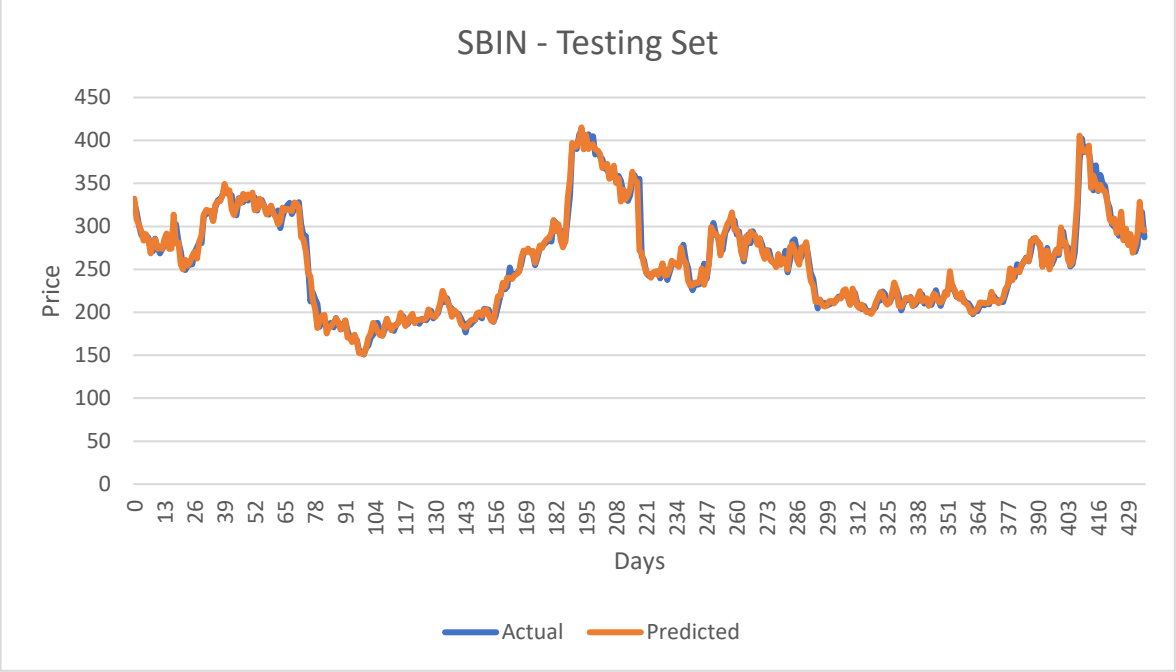


Figure 5.10. SBIN -Test

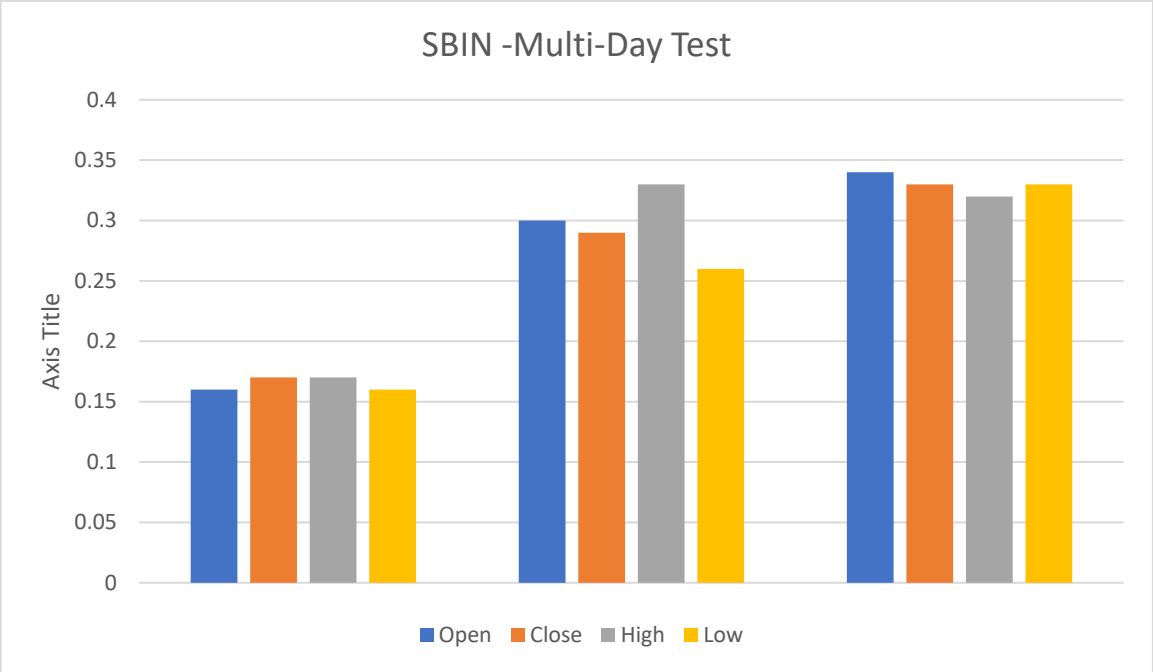


Figure 5.11. SBIN – Multi-Day Prediction

5.5.3. Reliance Industries Ltd.

Below we can see the result of the model for Reliance Industries Ltd. on the Training Set (Figure 5.12), Testing Set (Figure 5.13.) and performance on a small sample of multiple day predictions (Figure 5.14.).

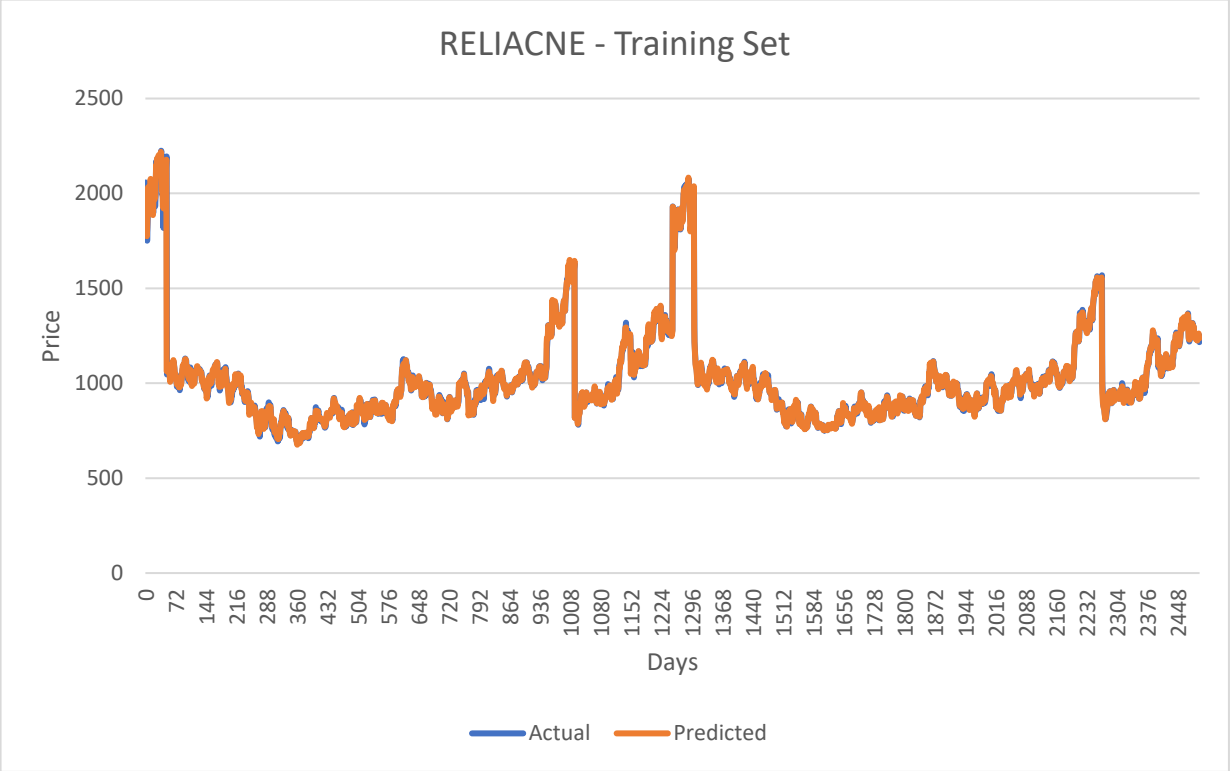


Figure 5.12. RELIANCE – Train

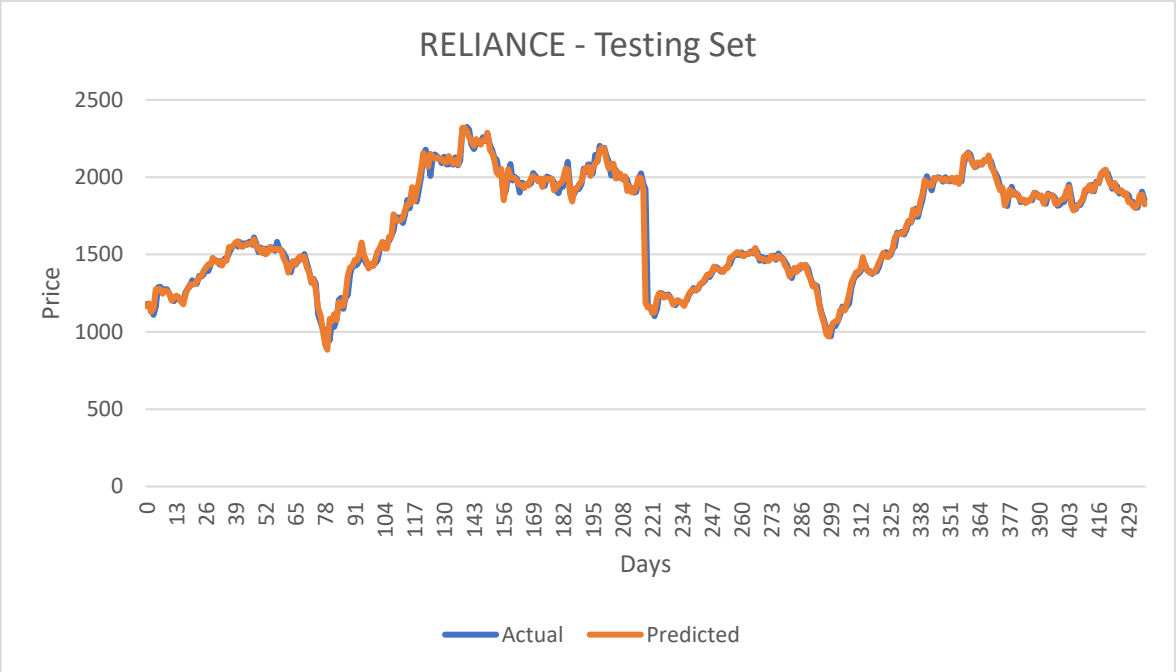


Figure 5.13. RELIANCE -Test

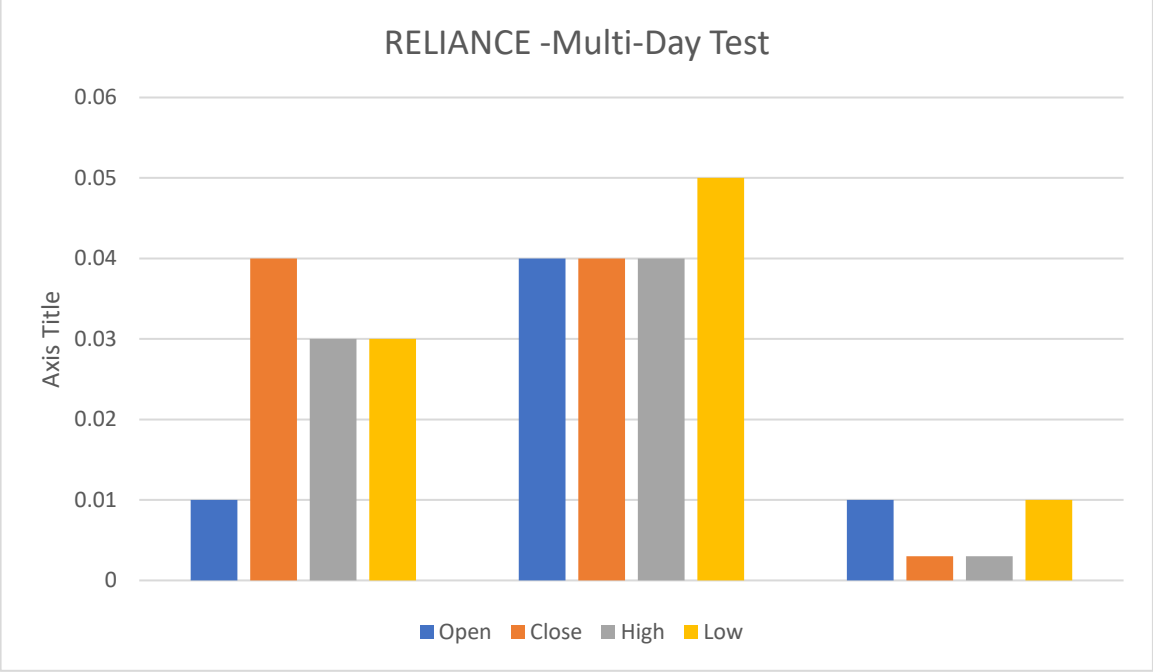


Figure 5.14. RELIANCE – Multi-Day Prediction

Chapter 6

Conclusion

In conclusion, to predict the stock prices we make use of a CNN-LSTM neural network. An LSTM network was chosen due to its ability to learn long-term dependencies and its ability to work on temporal data. For the data we used a mix of historical data fetched using an API and technical indicators calculated on that historical data.

The data is retrieved and processed for each stock symbol and a different model is trained for each new company entered. On doing tests on our model we were able to achieve accurate outputs, with low error for the open, close, high and low prices for a day. However, multi-day predictions were erratic in terms of accuracy and the error increased steeply as number of days were added, this is potentially an area for future research and work. Therefore, to conclude we have been successful in achieving our main goal for the project.

References

[1] V. Malepati, “A Study on Volatility in Indian Stock Market,” 2016.

[2] J. Sen and T. D. Chaudhuri, “An Alternative Framework for Time Series Decomposition and Forecasting and its Relevance for Portfolio Choice: A Comparative Study of the Indian Consumer Durable and Small Cap Sectors,” *Journal of Economics Library,* vol. 3, no. 2, June 2016.

[3] S. Kotsiantis, I. Zaharakis and P. Pintelas, “Machine learning: A review of classification and combining techniques,” *Artificial intelligence Review,* vol. 26, no. 3, 2006.

[4] J. Brownlee, “What Is Time Series Forecasting?,” 21 August 2019. [Online]. Available: https://machinelearningmastery.com/time-series-forecasting/.

[5] J. Brownlee, Deep Learning for Time Series Forecasting.

[6] K. Didur, “Machine learning in finance: Why, what & how,” Towards Data Science, 11 Jul 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-in-finance-why-what-how-d524a2357b56.

[7] M. M. M. P. a. R. M. P. Aparna Nayak, “Prediction Models for Indian Stock Market,” in *Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016)*, Manipal, 2016.

[8] P. Vijaya, G. Raju and S. K. Ray, “Artificial neural network-based merging score for Meta search engine,” *Journal of Central South University,* vol. 23, pp. 2604-2615, 2016.

[9] C. Evans and K. Pappas, “Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation,” *Mathematical and Computer Modelling,* vol. 58, no. 5-6, pp. 1249-1266, 2013.

[10] B. Sun, H. Guo, H. R. Karimi, Y. Ge and S. Xiong, “Prediction of stock index futures prices based on fuzzy sets and multivariate fuzzy time series,” *Neurocomputing,* vol. 151, no. 3, pp. 1528-1536, 2015.

[11] Facebook, “React,” Facebook, [Online]. Available: https://reactjs.org/. [Accessed 15 March 2020].

[12] C. J. Siddegowda, Gitu Mani Borah and K. A. Chandru, “REACT FRAMEWORK (CREATING A WEB APPLICATION WITH REACT NATIVE),”

INTERNATIONAL JOURNAL OF RECENT TRENDS IN ENGINEERING & RESEARCH, vol. 4, no. 3, 02 04 2018.

[13] Google, “Introduction,” Google, [Online]. Available: https://material.io/design/introduction. [Accessed 05 March 2021].

[14] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, M. Sherry, D. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu and X. Zheng, “TensorFlow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, USENIX Association, 2016, pp. 265-283.

[15] H. Nicholls, “7 Popular Technical Indicators and How to Use Them to Increase Your Trading Profits,” 09 April 2018. [Online]. Available: https://medium.com/@harrynicholls/7-popular-technical-indicators-and-how-to-use-them-to-increase-your-trading-profits-7f13ffeb8d05.

[16] FM Labs, Inc., “Indicator Reference,” FM Labs, Inc., [Online]. Available: https://www.fmlabs.com/reference/. [Accessed 18 11 2020].

[17] TicTacTec LLC., “Technical analysis documentation and forums for traders,” 03 22 08. [Online]. [Accessed 13 11 2020].

[18] T. Segal, “Momentum Indicators,” 1 May 2021. [Online]. Available: https://www.investopedia.com/investing/momentum-and-relative-strength-index/. [Accessed 5 May 2021].

[19] A. Hayes, “Investopedia,” 30 April 2021. [Online]. Available: https://www.investopedia.com/terms/s/stochasticoscillator.asp.

[20] C. Mitchell, “Average Directional Index (ADX),” 17 Febuary 2021. [Online]. Available: https://www.investopedia.com/terms/a/adx.asp.

[21] C. Mitchell, “Aroon Oscillator,” 21 April 2021. [Online]. Available: https://www.investopedia.com/terms/a/aroonoscillator.asp.

[22] S. Hochreiter and J. Schmidhuber, “Long Short-term Memory,” *Neural computation,* vol. 9, pp. 1735-80, 12 1997.

[23] C. Olah, “Understanding LSTM Networks,” 27 August 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed 1 December 2020].

[24] B. Dickson, “What are convolutional neural networks (CNN)?,” 6 January 2020. [Online]. Available: https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/.

[25] R. Agrawal, “Optimization Algorithms for Deep Learning,” 23 July 2019. [Online]. Available: https://medium.com/analytics-vidhya/optimization-algorithms-for-deep-learning-1f1a2bd4c46b#:~:text=In%20the%20context%20of%20deep,optimizing%20the%20cost%20function%20J.&text=With%20the%20help%20of%20optimization,trainable%20parameters%20W%20and%20b.. [Accessed 26 December 2020].

[26] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” *CoRR,* vol. abs/1212.5701, 2012.