

# **STOCK PRICE PREDICTION**

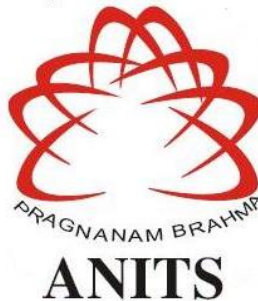
*A Project report submitted in partial fulfillment of the requirements for  
the award of the degree of*

## **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE ENGINEERING**

### *Submitted by*

Somaraju Dinesh	-	317126510170
Adduri Maruthi Siva Rama Raju	-	318126510L25
Sasumana Rahul	-	317126510167
Oruganti Naga Sandeep	-	318126510L29

**Under the guidance of  
N D S S Kiran Relangi  
Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES  
(UGC AUTONOMOUS)**

*(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade)*  
Sangivalasa, bheemili mandal, visakhapatnam dist.(A.P)  
2017-2021

## ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **N D S S KIRAN RELANGI**, Assistant Professor, Department of Computer Science and Engineering, ANITS, for his/her guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Dr.R.Sivaranjani**, Head of the Department, Computer Science and Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, ANITS, Sangivalasa**, for their encouragement and cooperation to carry out this work.

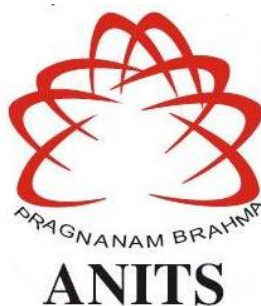
We express our thanks to Project Coordinator **Dr.V.Usha Bala**, for her Continuous support and encouragement. We thank all **teaching faculty** of Department of CSE, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank **S. Sajahan** of the Department of CSE, ANITS for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

## PROJECT STUDENTS

Somaraju Dinesh	- 317126510170
Adduri Maruthi Siva Rama Raju	- 318126510L25
Sasumana Rahul	- 317126510167
Oruganti Naga Sandeep	- 318126510L29

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES**  
**(UGC AUTONOMOUS)**  
*(Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade)*  
**Sangivalasa, bheemili mandal, visakhapatnam dist.(A.P)**



**CERTIFICATE**

This is to certify that the project report entitled “**STOCK PRICE PREDICTION**” submitted by **Somaraju Dinesh (317126510170)**, **Adduri Maruthi Siva Rama Raju (318126510L25)**, **Sasumana Rahul (317126510167)**, **Oruganti Naga Sandeep (318126510L29)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science Engineering** of Anil Neerukonda Institute of technology and sciences (A), Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.

**Project Guide**

**N D S S KIRAN RELANGI**  
ASSISTANT PROFESSOR  
Department of CSE  
ANITS

**Head of the Department**

**Dr. R. SIVARANJANI**  
Department of CSE  
ANITS

## DECLARATION

We, **SOMARAJU DINESH, ADDURI MARUTHI SIVA RAMA RAJU, SASUMANA RAHUL, OORUGANTI NAGA SANDEEP**, of final semester B.Tech., in the department of Computer Science and Engineering from ANITS, Visakhapatnam, hereby declare that the project work entitled **STOCK PRICE PREDICTION** is carried out by us and submitted in partial fulfillment of the requirements for the award of **Bachelor of Technology in Computer Science Engineering** , under Anil Neerukonda Institute of Technology & Sciences(A) during the academic year 2017-2021 and has not been submitted to any other university for the award of any kind of degree.

Somaraju Dinesh	-	317126510170
Adduri Maruthi Siva Rama Raju	-	318126510L25
Sasumana Rahul	-	317126510167
Oruganti Naga Sandeep	-	318126510L29

## **ABSTRACT**

In this project we attempt to implement machine learning approach to predict stock prices. Machine learning is effectively implemented in forecasting stock prices. The objective is to predict the stock prices in order to make more informed and accurate investment decisions. We propose a stock price prediction system that integrates mathematical functions, machine learning, and other external factors for the purpose of achieving better stock prediction accuracy and issuing profitable trades.

There are two types of stocks. You may know of intraday trading by the commonly used term "day trading." Interday traders hold securities positions from at least one day to the next and often for several days to weeks or months. LSTMs are very powerful in sequence prediction problems because they're able to store past information. This is important in our case because the previous price of a stock is crucial in predicting its future price. While predicting the actual price of a stock is an uphill climb, we can build a model that will predict whether the price will go up or down.

**Keywords:** LSTM, CNN, ML, DL, Trade Open, Trade Close, Trade Low, Trade High

# CONTENTS

<b>ABSTRACT</b>	v
<b>LIST OF SYMBOLS</b>	viii
<b>LIST OF FIGURES</b>	ix
<b>LIST OF TABLES</b>	x
<b>LIST OF ABBREVIATIONS</b>	xi
<b>CHAPTER 1 INTRODUCTION</b>	01
1.1 Motivation for work	
1.2 Problem statement	
<b>CHAPTER 2 LITERATURE SURVEY</b>	03
2.1 Introduction	
2.2 Existing methods	
2.2.1 Stock Market Prediction Using Machine Learning	
2.2.2 Forecasting the Stock Market Index Using Artificial Intelligence Techniques	
2.2.3 Indian stock market prediction using artificial neural network	
2.2.4 The Stock Market and Investment	
2.2.5 Automated Stock Price Prediction Using Machine Learning	
2.2.6 Stock Price Correlation Coefficient Prediction with ARIMA-LSTM Hybrid Model	
2.2.7 Event Representation Learning Enhanced with External Common-sense Knowledge	
2.2.8 Forecasting directional movements of stock prices for trading using LSTM and random forests	
2.2.9 A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance	
2.2.10 An innovative neural network approach for stock market prediction	
2.2.11 An Intelligent Technique for Stock Market Prediction	

<b>CHAPTER 3 METHODOLOGY</b>	10
3.1 proposed system	
3.1.1 system architecture	
3.2 Hardware Requirements	
3.3 Software Requirements	
3.4 Functional Requirements	
3.5 Non-Functional Requirements	
<b>CHAPTER 4 DESIGN</b>	18
4.1 Structure chart	
4.2 UML Diagrams	
4.2.1 use case diagram	
4.2.2 sequence diagram	
4.2.3 activity diagram	
4.2.4 collaboration diagram	
4.2.5 flow chart diagram	
4.2.6 component diagram	
<b>CHAPTER 5 EXPERIMENTAL ANALYSIS AND RESULTS</b>	26
5.1 system configuration	
5.1.1 hardware requirements	
5.1.2 software requirements	
5.2 sample code	
5.3 input and output	
5.3.1 input	
5.3.2 output	
5.4 Website Pages	
5.5 Performance Measure	
<b>CHAPTER 6 CONCLUSION AND FUTURE WORK</b>	57
<b>REFERENCES</b>	58

## LIST OF SYMBOLS

$X_t$	Input at current state
$X(t-1)$	Input at Previous state
$C_t$	Current Cell State
$C(t-1)$	Previous Cell State
$h_t$	Current hidden/output State
$h(t-1)$	Previous hidden/output State
$\sigma$	Sigmoid Function
$\tanh$	Hyperbolic tangent function



## LIST OF FIGURES

<b>Fig.No.</b>	<b>Topic Name</b>	<b>Page No.</b>
1	LMS Inputs and Outputs	11
2	LMS updating weights	11
3	LMS updating weights	12
4	LSTM Architecture	14
5	Pre-processing of data	17
6	Overall Architecture	17
7	Structure Chart	18
8	Use case diagram	20
9	Sequence diagram	21
10	Activity diagram	22
11	Collaboration diagram	23
12	Flow chart	24
13	Component diagram	25
14	Home page	51
15	Training page	51
16	Training Page: After Selecting the dataset	52
17	Training Page: While Training	52
18	Training Page: Training Completed	53
19	Predictions Page	53
20	Predictions Page: After selecting the model	54
20	Team Page	54

## LIST OF TABLES

Table No.	Topic Name	Page No.
1	Min and Max of columns in Google Dataset	49
2	Min and Max of columns in Nifty50 Dataset	50
3	Min and Max of columns in Reliance Dataset	50
4	Sample input	50
5	Epochs for Google Dataset using LSTM	55
6	Epochs for Nifty50 Dataset using LSTM	55
7	Epochs for Reliance Dataset using LSTM	55
8	Epochs for Google Dataset using LSTM with LMS	56
9	Epochs for Nifty50 Dataset using LSTM with LMS	56
10	Epochs for Reliance Dataset using LSTM with LMS	56

## **LIST OF ABBREVIATIONS**

LSTM	Long Short-Term Memory
ATS	Automated Trading System
GRU	Gated Recurrent Unit
ML	Machine Learning
SVM	Support Vector Machine
EMH	Efficient Market hypothesis
AI	Artificial Intelligence
NN	Neural Networks
ARMA	Autoregressive Moving Average
DRL	Deep Reinforcement Learning
LMS	Least Mean Square
UML	Unified modelling Language
MSE	Mean Squared Error
RMSE	Root Mean Squared Error

# **CHAPTER 1**

## **INTRODUCTION**

The financial market is a dynamic and composite system where people can buy and sell currencies, stocks, equities and derivatives over virtual platforms supported by brokers. The stock market allows investors to own shares of public companies through trading either by exchange or over the counter markets. This market has given investors the chance of gaining money and having a prosperous life through investing small initial amounts of money, low risk compared to the risk of opening new business or the need of high salary career. Stock markets are affected by many factors causing the uncertainty and high volatility in the market. Although humans can take orders and submit them to the market, automated trading systems (ATS) that are operated by the implementation of computer programs can perform better and with higher momentum in submitting orders than any human. However, to evaluate and control the performance of ATSs, the implementation of risk strategies and safety measures applied based on human judgements are required. Many factors are incorporated and considered when developing an ATS, for instance, trading strategy to be adopted, complex mathematical functions that reflect the state of a specific stock, machine learning algorithms that enable the prediction of the future stock value, and specific news related to the stock being analysed.

Time-series prediction is a common technique widely used in many real-world applications such as weather forecasting and financial market prediction. It uses the continuous data in a period of time to predict the result in the next time unit. Many time-series prediction algorithms have shown their effectiveness in practice. The most common algorithms now are based on Recurrent Neural Networks (RNN), as well as its special type - Long-short Term Memory (LSTM) and Gated Recurrent Unit (GRU). Stock market is a typical area that presents time-series data and many researchers study on it and proposed various models. In this project, LSTM model is used to predict the stock price.

## **1.1 MOTIVATION FOR WORK**

Businesses primarily run over customer's satisfaction, customer reviews about their products. Shifts in sentiment on social media have been shown to correlate with shifts in stock markets. Identifying customer grievances thereby resolving them leads to customer satisfaction as well as trustworthiness of an organization. Hence there is a necessity of an unbiased automated system to classify customer reviews regarding any problem. In today's environment where we're justifiably suffering from data overload (although this does not mean better or deeper insights), companies might have mountains of customer feedback collected; but for mere humans, it's still impossible to analyse it manually without any sort of error or bias. Oftentimes, companies with the best intentions find themselves in an insights vacuum. You know you need insights to inform your decision making and you know that you're lacking them, but don't know how best to get them. Sentiment analysis provides some answers into what the most important issues are, from the perspective of customers, at least. Because sentiment analysis can be automated, decisions can be made based on a significant amount of data rather than plain intuition.

## **1.2 PROBLEM STATEMENT**

Time Series forecasting & modelling plays an important role in data analysis. Time series analysis is a specialized branch of statistics used extensively in fields such as Econometrics & Operation Research. Time Series is being widely used in analytics & data science. Stock prices are volatile in nature and price depends on various factors. The main aim of this project is to predict stock prices using Long short term memory (LSTM).

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 INTRODUCTION**

"What other people think" has always been an important piece of information for most of us during the decision-making process. The Internet and the Web have now (among other things) made it possible to find out about the opinions and experiences of those in the vast pool of people that are neither our personal acquaintances nor well-known professional critics — that is, people we have never heard of. And conversely, more and more people are making their opinions available to strangers via the Internet. The interest that individual users show in online opinions about products and services, and the potential influence such opinions wield, is something that is driving force for this area of interest. And there are many challenges involved in this process which needs to be walked all over in order to attain proper outcomes out of them. In this survey we analysed basic methodology that usually happens in this process and measures that are to be taken to overcome the challenges being faced.

#### **2.2 EXISTING METHODS**

##### **2.2.1 Stock Market Prediction Using Machine Learning**

The research work done by V Kranthi Sai Reddy Student, ECM, Sreenidhi Institute of Science and Technology, Hyderabad, India. In the finance world stock trading is one of the most important activities. Stock market prediction is an act of trying to determine the future value of a stock other financial instrument traded on a financial exchange. This paper explains the prediction of a stock using Machine Learning. The technical and fundamental or the time series analysis is used by the most of the stockbrokers while making the stock predictions. The programming language is used to predict the stock market using machine learning is Python. In this paper we propose a Machine Learning (ML) approach that will be trained from the available stocks data and gain intelligence and then uses the acquired knowledge for an accurate prediction. In this context this study uses a machine learning technique called Support Vector Machine (SVM) to predict stock prices for the large and small capitalizations and in the three different markets, employing prices with both daily and up-to-the-minute frequencies.

### **2.2.2 Forecasting the Stock Market Index Using Artificial Intelligence Techniques**

The research work done by Lufuno Ronald Marwala A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering. The weak form of Efficient Market hypothesis (EMH) states that it is impossible to forecast the future price of an asset based on the information contained in the historical prices of an asset. This means that the market behaves as a random walk and as a result makes forecasting impossible. Furthermore, financial forecasting is a difficult task due to the intrinsic complexity of the financial system. The objective of this work was to use artificial intelligence (AI) techniques to model and predict the future price of a stock market index. Three artificial intelligence techniques, namely, neural networks (NN), support vector machines and neuro-fuzzy systems are implemented in forecasting the future price of a stock market index based on its historical price information. Artificial intelligence techniques have the ability to take into consideration financial system complexities and they are used as financial time series forecasting tools.

Two techniques are used to benchmark the AI techniques, namely, Autoregressive Moving Average (ARMA) which is linear modelling technique and random walk (RW) technique. The experimentation was performed on data obtained from the Johannesburg Stock Exchange. The data used was a series of past closing prices of the All Share Index. The results showed that the three techniques have the ability to predict the future price of the Index with an acceptable accuracy. All three artificial intelligence techniques outperformed the linear model. However, the random walk method outperformed all the other techniques. These techniques show an ability to predict the future price however, because of the transaction costs of trading in the market, it is not possible to show that the three techniques can disprove the weak form of market efficiency. The results show that the ranking of performances support vector machines, neuro-fuzzy systems, multilayer perceptron neural networks is dependent on the accuracy measure used.

### **2.2.3 Indian stock market prediction using artificial neural networks on tick data**

The research work done by Dharmaraja Selvamuthu, Vineet Kumar and Abhishek Mishra Department of Mathematics, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India. A stock market is a platform for trading of a company's stocks and derivatives at an agreed price. Supply and demand of shares drive the stock market. In any country stock market is one of the most emerging sectors. Nowadays, many people are indirectly or directly related to this sector. Therefore, it becomes essential to know about market trends. Thus, with the development of the stock

market, people are interested in forecasting stock price. But, due to dynamic nature and liable to quick changes in stock price, prediction of the stock price becomes a challenging task. Stock m Prior work has proposed effective methods to learn event representations that can capture syntactic and semantic information over text corpus, demonstrating their effectiveness for downstream tasks such as script event prediction. On the other hand, events extracted from raw texts lacks of common-sense knowledge, such as the intents and emotions of the event participants, which are useful for distinguishing event pairs when there are only subtle differences in their surface realizations. To address this issue, this paper proposes to leverage external common-sense knowledge about the intent and sentiment of the event.

Experiments on three event-related tasks, i.e., event similarity, script event prediction and stock market prediction, show that our model obtains much better event embeddings for the tasks, achieving 78% improvements on hard similarity task, yielding more precise inferences on subsequent events under given contexts, and better accuracies in predicting the volatilities of the stock market<sup>1</sup>. Markets are mostly a non-parametric, non-linear, noisy and deterministic chaotic system (Ahanger et al. 2010). As the technology is increasing, stock traders are moving towards to use Intelligent Trading Systems rather than fundamental analysis for predicting prices of stocks, which helps them to take immediate investment decisions. One of the main aims of a trader is to predict the stock price such that he can sell it before its value decline, or buy the stock before the price rises. The efficient market hypothesis states that it is not possible to predict stock prices and that stock behaves in the random walk. It seems to be very difficult to replace the professionalism of an experienced trader for predicting the stock price. But because of the availability of a remarkable amount of data and technological advancements we can now formulate an appropriate algorithm for prediction whose results can increase the profits for traders or investment firms. Thus, the accuracy of an algorithm is directly proportional to gains made by using the algorithm.

#### **2.2.4 The Stock Market and Investment**

The research work done by Manh Ha Duong Boriss Siliverstovs. Investigating the relation between equity prices and aggregate investment in major European countries including France, Germany, Italy, the Netherlands and the United Kingdom. Increasing integration of European financial markets is likely to result in even stronger correlation between equity prices in different European countries. This process can also lead to convergence in economic development across European countries if developments in stock markets influence real economic components, such as investment and consumption. Indeed, our vector autoregressive models suggest that the positive correlation between changes equity prices and investment is, in general, significant. Hence,



monetary authorities should monitor reactions of share prices to monetary policy and their effects on the business cycle.

### **2.2.5 Automated Stock Price Prediction Using Machine Learning**

The research work done by Mariam Moukalled Wassim El-Hajj Mohamad Jaber Computer Science Department American University of Beirut. Traditionally and in order to predict market movement, investors used to analyse the stock prices and stock indicators in addition to the news related to these stocks. Hence, the importance of news on the stock price movement. Most of the previous work in this industry focused on either classifying the released market news as (positive, negative, neutral) and demonstrating their effect on the stock price or focused on the historical price movement and predicted their future movement. In this work, we propose an automated trading system that integrates mathematical functions, machine learning, and other external factors such as news' sentiments for the purpose of achieving better stock prediction accuracy and issuing profitable trades. Particularly, we aim to determine the price or the trend of a certain stock for the coming end-of-day considering the first several trading hours of the day. To achieve this goal, we trained traditional machine learning algorithms and created/trained multiple deep learning models taking into consideration the importance of the relevant news. Various experiments were conducted, the highest accuracy (82.91%) of which was achieved using SVM for Apple Inc. (AAPL) stock.

### **2.2.6 Stock Price Correlation Coefficient Prediction with ARIMA-LSTM Hybrid Model**

The research work done by Hyeong Kyu Choi, B.A Student Dept. of Business Administration Korea University Seoul, Korea. Predicting the price correlation of two assets for future time periods is important in portfolio optimization. We apply LSTM recurrent neural networks (RNN) in predicting the stock price correlation coefficient of two individual stocks. RNN's are competent in understanding temporal dependencies. The use of LSTM cells further enhances its long-term predictive properties. To encompass both linearity and nonlinearity in the model, we adopt the ARIMA model as well. The ARIMA model filters linear tendencies in the data and passes on the residual value to the LSTM model. The ARIMA-LSTM hybrid model is tested against other traditional predictive financial models such as the full historical model, constant correlation model, single-index model and the multi-group model. In our empirical study, the predictive ability of the ARIMA-LSTM model turned out superior to all other financial models by a significant scale. Our work implies that it is worth considering the ARIMALSTM model to forecast correlation coefficient for portfolio optimization.

### **2.2.7 Event Representation Learning Enhanced with External Common-sense Knowledge**

The research work done by Xiao Ding, Kuo Liao, Ting Liu, Zhongyang Li, Junwen Duan Research Center for Social Computing and Information Retrieval Harbin Institute of Technology, China. Prior work has proposed effective methods to learn event representations that can capture syntactic and semantic information over text corpus, demonstrating their effectiveness for downstream tasks such as script event prediction. On the other hand, events extracted from raw texts lacks of common-sense knowledge, such as the intents and emotions of the event participants, which are useful for distinguishing event pairs when there are only subtle differences in their surface realizations. To address this issue, this paper proposes to leverage external common-sense knowledge about the intent and sentiment of the event. Experiments on three event-related tasks, i.e., event similarity, script event prediction and stock market prediction, show that our model obtains much better event embeddings for the tasks, achieving 78% improvements on hard similarity task, yielding more precise inferences on subsequent events under given contexts, and better accuracies in predicting the volatilities of the stock market.

### **2.2.8 Forecasting directional movements of stock prices for intraday trading using LSTM and random forests**

The research work done by Pushpendu Ghosh, Ariel Neufeld, Jajati Keshari Sahoo Department of Computer Science & Information Systems, BITS Pilani K.K.Birla Goa campus, India bDivision of Mathematical Sciences, Nanyang Technological University, Singapore cDepartment of Mathematics, BITS Pilani K.K.Birla Goa campus, India. We employ both random forests and LSTM networks (more precisely CuDNNLSTM) as training methodologies to analyse their effectiveness in forecasting out-of-sample directional movements of constituent stocks of the S&P 500 from January 1993 till December 2018 for intraday trading. We introduce a multi-feature setting consisting not only of the returns with respect to the closing prices, but also with respect to the opening prices and intraday returns. As trading strategy, we use Krauss et al. (2017) and Fischer & Krauss (2018) as benchmark and, on each trading day, buy the 10 stocks with the highest probability and sell short the 10 stocks with the lowest probability to outperform the market in terms of intraday returns – all with equal monetary weight. Our empirical results show that the multi-feature setting provides a daily return, prior to transaction costs, of 0.64% using LSTM networks, and 0.54% using random forests. Hence, we outperform the single-feature setting in Fischer & Krauss (2018) and Krauss et al. (2017) consisting only of the daily returns with respect to the closing prices, having corresponding daily returns of 0.41% and of 0.39% with respect to LSTM and random forests, respectively. 1 Keywords: Random forest, LSTM, Forecasting, Statistical Arbitrage, Machine learning, Intraday trading.

### **2.2.9 A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance**

The research work done by Xiao-Yang Liu<sup>1</sup> Hongyang Yang, Qian Chen<sup>4</sup>, Runjia Zhang, Liuqing Yang, Bowen Xiao, Christina Dan, Wang, Electrical Engineering, <sup>2</sup>Department of Statistics, <sup>3</sup>Computer Science, Columbia University, <sup>3</sup>AI4Finance LLC., USA, Ion Media Networks, USA, Department of Computing, Imperial College, <sup>6</sup>New York University (Shanghai). As deep reinforcement learning (DRL) has been recognized as an effective approach in quantitative finance, getting hands-on experiences is attractive to beginners. However, to train a practical DRL trading agent that decides where to trade, at what price, and what quantity involves error-prone and arduous development and debugging. In this paper, we introduce a DRL library FinRL that facilitates beginners to expose themselves to quantitative finance and to develop their own stock trading strategies. Along with easily-reproducible tutorials, FinRL library allows users to streamline their own developments and to compare with existing schemes easily.

Within FinRL, virtual environments are configured with stock market datasets, trading agents are trained with neural networks, and extensive back testing is analysed via trading performance. Moreover, it incorporates important trading constraints such as transaction cost, market liquidity and the investor's degree of risk-aversion. FinRL is featured with completeness, hands-on tutorial and reproducibility that favors beginners: (i) at multiple levels of time granularity, FinRL simulates trading environments across various stock markets, including NASDAQ-100, DJIA, S&P 500, HSI, SSE 50, and CSI 300; (ii) organized in a layered architecture with modular structure, FinRL provides fine-tuned state-of-the-art DRL algorithms (DQN, DDPG, PPO, SAC, A2C, TD3, etc.), commonly used reward functions and standard evaluation baselines to alleviate the debugging workloads and promote the reproducibility, and (iii) being highly extendable, FinRL reserves a complete set of user-import interfaces. Furthermore, we incorporated three application demonstrations, namely single stock trading, multiple stock trading, and portfolio allocation. The FinRL library will be available on GitHub at link <https://github.com/AI4Finance-LLC/FinRL-Library>.

### **2.2.10 An innovative neural network approach for stock market prediction**

The research work done by Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin. To develop an innovative neural network approach to achieve better stock market predictions. Data were obtained from the live stock market for real-time and off-line analysis and results of visualizations and analytics to demonstrate Internet of Multimedia of Things for stock analysis. To study the influence of market characteristics on stock prices, traditional neural network algorithms may incorrectly predict the stock market, since the initial weight of the random selection problem can be easily prone to incorrect predictions.

Based on the development of word vector in deep learning, we demonstrate the concept of “stock vector.” The input is no longer a single index or single stock index, but multi-stock high-dimensional historical data. We propose the deep long short-term memory neural network (LSTM) with embedded layer and the long short-term memory neural network with automatic encoder to predict the stock market. In these two models, we use the embedded layer and the automatic encoder, respectively, to vectorize the data, in a bid to forecast the stock via long short-term memory neural network. The experimental results show that the deep LSTM with embedded layer is better. Specifically, the accuracy of two models is 57.2 and 56.9%, respectively, for the Shanghai A-shares composite index. Furthermore, they are 52.4 and 52.5%, respectively, for individual stocks. We demonstrate research contributions in IMMT for neural network-based financial analysis.

### 2.2.11 An Intelligent Technique for Stock Market Prediction

### **2.2.11 An Intelligent Technique for Stock Market Prediction**

The research work done by M. Mekayel Anik · M. Shamsul Arefin (B) Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong, Bangladesh. A stock market is a loose network of economic transactions between buyers and sellers based on stocks also known as shares. In stock markets, stocks represent the ownership claims on businesses. These may include securities listed on a stock exchange as well as those only traded privately. A stock exchange is a place where brokers can buy and/or sell stocks, bonds, and other securities. Stock market is a very vulnerable place for investment due to its volatile nature. In the near past, we faced huge financial problems due to huge drop in price of shares in stock markets worldwide. This phenomenon brought a heavy toll on the international as well as on our national financial structure. Many people lost their last savings of money on the stock market. In 2010–2011 financial year, Bangladeshi stock market faced massive collapse [1]. This phenomenon can be brought under control especially by strict monitoring and instance stock market analysis. If we can analyse stock market correctly in time, it can become a field of large profit and may become comparatively less vulnerable for the investors.

Stock market is all about prediction and rapid decision making about investment, which cannot be done without thorough analysis of the market. If we can predict the stock market by analysing historical data properly, we can avoid the consequences of serious market collapse and to be able to take necessary steps to make market immune to such situations.

## **CHAPTER 3**

### **METHODOLOGY**

#### **3.1 PROPOSED SYSTEMS**

The prediction methods can be roughly divided into two categories, statistical methods and artificial intelligence methods. Statistical methods include logistic regression model, ARCH model, etc. Artificial intelligence methods include multi-layer perceptron, convolutional neural network, naive Bayes network, back propagation network, single-layer LSTM, support vector machine, recurrent neural network, etc. They used Long short-term memory network (LSTM).

##### **Long short-term memory network:**

Long short-term memory network (LSTM) is a particular form of recurrent neural network (RNN).

##### **Working of LSTM:**

LSTM is a special network structure with three “gate” structures. Three gates are placed in an LSTM unit, called input gate, forgetting gate and output gate. While information enters the LSTM’s network, it can be selected by rules. Only the information conforms to the algorithm will be left, and the information that does not conform will be forgotten through the forgetting gate.

The experimental data in this paper are the actual historical data downloaded from the Internet. Three data sets were used in the experiments. It is needed to find an optimization algorithm that requires less resources and has faster convergence speed.

- Used Long Short-term Memory (LSTM) with embedded layer and the LSTM neural network with automatic encoder.
- LSTM is used instead of RNN to avoid exploding and vanishing gradients.
- In this project python is used to train the model, MATLAB is used to reduce dimensions of the input. MySQL is used as a dataset to store and retrieve data.
- The historical stock data table contains the information of opening price, the highest price, lowest price, closing price, transaction date, volume and so on.
- The accuracy of this LSTM model used in this project is 57%.

## LMS filter:

The LMS filter is a kind of adaptive filter that is used for solving linear problems. The idea of the filter is to minimize a system (finding the filter coefficients) by minimizing the least mean square of the error signal.

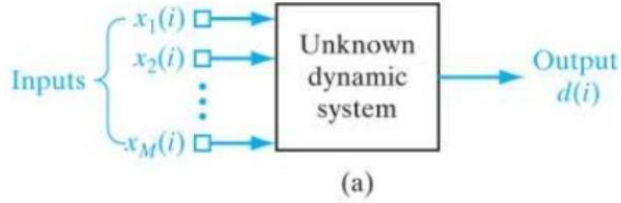


Fig. 1: LMS Inputs and Outputs

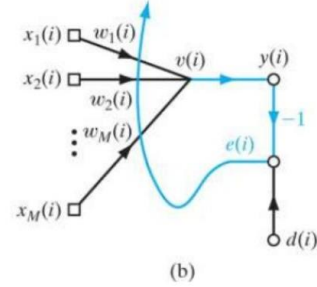


Fig 2: LMS updating weights

---

### Algorithm 1: LMS

---

**Input:**

$x$  : input vector  
 $d$ : desired vector  
 $\mu$ : learning rate  
 $N$ : filter order

**Output:**

$y$ : filter response  
 $e$ : filter error

**begin**

```

     $M = \text{size}(x)$  ;
     $x_n(0) = w_n(0) = [0 \ 0 \ \dots \ 0]^T$ ;
    while  $n < M$  do
         $x_{n+1} = [x(n); x_n(1 : N)]$ ;
         $y(n) = w_n^H * x_n$ ;
         $e(n) = d(n) - y(n)$ ;
         $w_{n+1} = w_n + 2\mu e(n)x_n$ ;
    
```

**end**

**end**

---

In general, we don't know exactly if the problem can be solved very well with linear approach, so we usually test a **linear** and a **non-linear** algorithm. Since the internet always shows non-linear approaches, we will use LMS to prove that stock market prediction **can** be done with linear algorithms with a **good precision**.

But this filter **mimetizes** a system, that is, if we apply this filter in our data, we will have the **filter coefficients** trained, and when we input a new vector, our filter coefficients will output a response that the original system would (in the best case). So we just have to do a *tricky* modification for using this filter to predict data.

### The system:

First, we will delay our input vector by  $l$  positions, where  $l$  would be the quantity of days we want to predict, this  $l$  new positions will be filled by **zeros**.

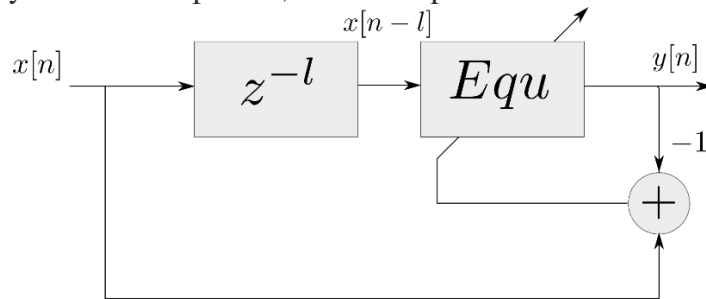


Fig. 3: LMS updating weights

When we apply the LMS filter, we will train the filter to the first 178 data. After that, we will set the error as zero, so the system will start to output the answers as the original system to the last  $l$  values. We will call the *tricky* modification as the **LMSPred algorithm**.

---

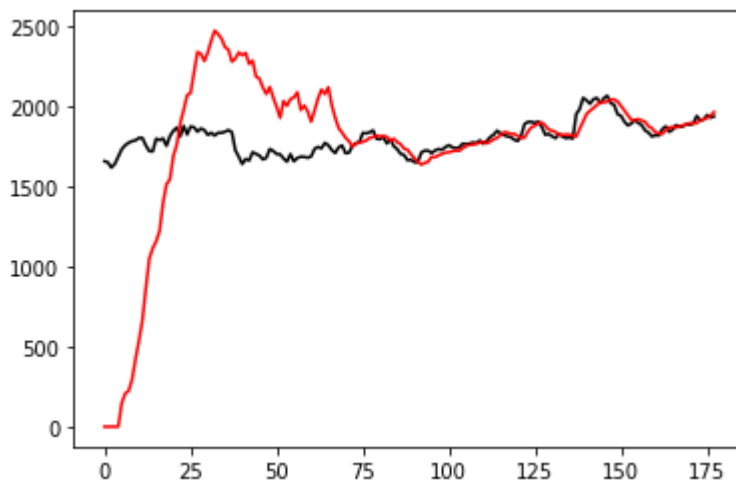
**Algorithm 2: LMSPred**

---

**Input:** $x$ : input vector $l$ : quantity of days to predict $\mu$ : learning rate $N$ : filter order**Output:** $y$ : filter response**begin** $M = \text{size}(x_d)$ ; $x_n(0) = w_n(0) = [0 \ 0 \ \dots \ 0]$ ; $x_d = [0 \ 0 \ \dots \ 0 \ x]$ ;**while**  $n < M$  **do** $x_{n+1} = [x_d(n); x_n(1 : N)]$ ; $y(n) = w_n^H * x_n$ ;**if**  $n > M - l$  **then** $e = 0$ ;**else** $e(n) = d(n) - y(n)$ ;**end** $w_{n+1} = w_n + 2\mu e(n)x_n$ ;**end****end**

---

## Results



One example of stock market prediction result



## LSTM Architecture

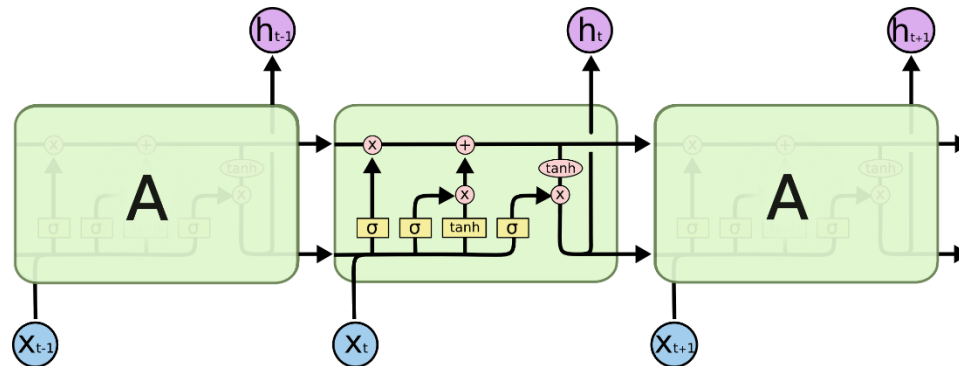


Fig. 4: LSTM Architecture

### Forget Gate:

A forget gate is responsible for removing information from the cell state.

- The information that is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter.
- This is required for optimizing the performance of the LSTM network.
- This gate takes in two inputs;  $h_{t-1}$  and  $x_t$ .  $h_{t-1}$  is the hidden state from the previous cell or the output of the previous cell and  $x_t$  is the input at that particular time step.

### Input Gate:

1. Regulating what values need to be added to the cell state by involving a sigmoid function. This is basically very similar to the forget gate and acts as a filter for all the information from  $h_{t-1}$  and  $x_t$ .
2. Creating a vector containing all possible values that can be added (as perceived from  $h_{t-1}$  and  $x_t$ ) to the cell state. This is done using the tanh function, which outputs values from -1 to +1.
3. Multiplying the value of the regulatory filter (the sigmoid gate) to the created vector (the tanh function) and then adding this useful information to the cell state via addition operation.

## Output Gate:

The functioning of an output gate can again be broken down to three steps:

- Creating a vector after applying tanh function to the cell state, thereby scaling the values to the range -1 to +1.
- Making a filter using the values of  $h_{t-1}$  and  $x_t$ , such that it can regulate the values that need to be output from the vector created above. This filter again employs a sigmoid function.
- Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as a output and also to the hidden state of the next cell.

```
• # LSTM
• Inputs: dataset
• Outputs: RMSE of the forecasted data
•
• # Split dataset into 75% training and 25% testing data
• size = length(dataset) * 0.75
• train = dataset [0 to size]
• test = dataset [size to length(dataset)]
•
• # Procedure to fit the LSTM model
• Procedure LSTMAlgorithm (train, test, train_size, epochs)
•     X = train
•     y = test
•     model = Sequential ()
•     model.add(LSTM(50), stateful=True)
•     model.compile(optimizer='adam', loss='mse')
•     model.fit(X, y, epochs=epochs, validation_split=0.2)
•     return model
•
• # Procedure to make predictions
• Procedure getPredictionsFromModel (model, X)
•     predictions = model.predict(X)
•     return predictions
•
• epochs = 100
• neurons = 50
• predictions = empty
```

```

• # Fit the LSTM model
• model = LSTMAlgorithm (train, epoch, neurons)
•
• # Make predictions
• pred = model.predict(train)
•
• # Validate the model
• n = len(dataset)
•
• error = 0
• for i in range(n): error += (abs(real[i] - pred[i])/real[i]) * 100
• accuracy = 100 - error/n

```

#### **Hardware Requirements:**

- RAM: 4 GB
- Storage: 500 GB
- CPU: 2 GHz or faster
- Architecture: 32-bit or 64-bit

#### **Software Requirements:**

- Python 3.5 in Google Colab is used for data pre-processing, model training and prediction.
- Operating System: windows 7 and above or Linux based OS or MAC OS

#### **Functional requirements**

Functional requirements describe what the software should do (the functions). Think about the core operations.

Because the “functions” are established before development, functional requirements should be written in the future tense. In developing the software for Stock Price Prediction, some of the functional requirements could include:

- The software shall accept the tw\_spydata\_raw.csv dataset as input.
- The software should shall do pre-processing (like verifying for missing data values) on input for model training.
- The software shall use LSTM ARCHITECTURE as main component of the software.
- It processes the given input data by producing the most possible outcomes of a CLOSING STOCK PRICE.

Notice that each requirement is directly related to what we expect the software to do. They represent some of the core functions.

### Non-Functional requirements

Product properties

- Usability: It defines the user interface of the software in terms of simplicity of understanding the user interface of stock prediction software, for any kind of stock trader and other stakeholders in stock market.
- Efficiency: maintaining the possible highest accuracy in the closing stock prices in shortest time with available data.

Performance: It is a quality attribute of the stock prediction software that describes the responsiveness to various user interactions with it.

### 3.1.1 SYSTEM ARCHITECTURE

#### 1) Preprocessing of data



Fig. 5: Pre-processing of data

#### 2) Overall Architecture

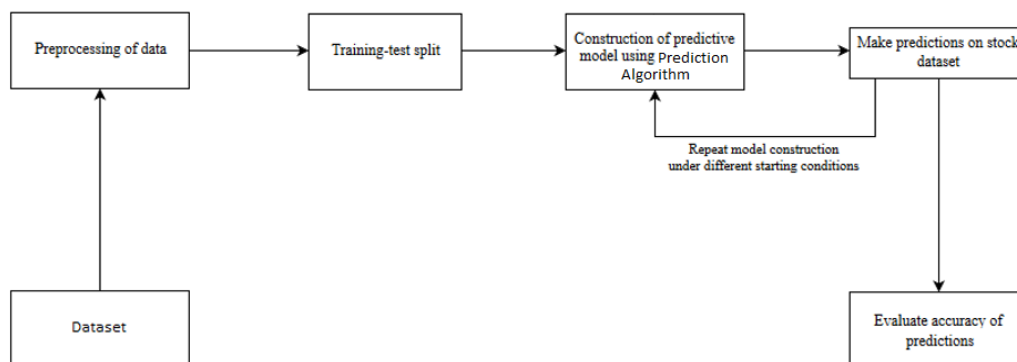


Fig. 6: Overall Architecture

## CHAPTER 4 DESIGN

### 4.1 Structure Chart

A structure chart (SC) in software engineering and organizational theory is a chart which shows the breakdown of a system to its lowest manageable levels. They are used in structured programming to arrange program modules into a tree. Each module is represented by a box, which contains the module's name.

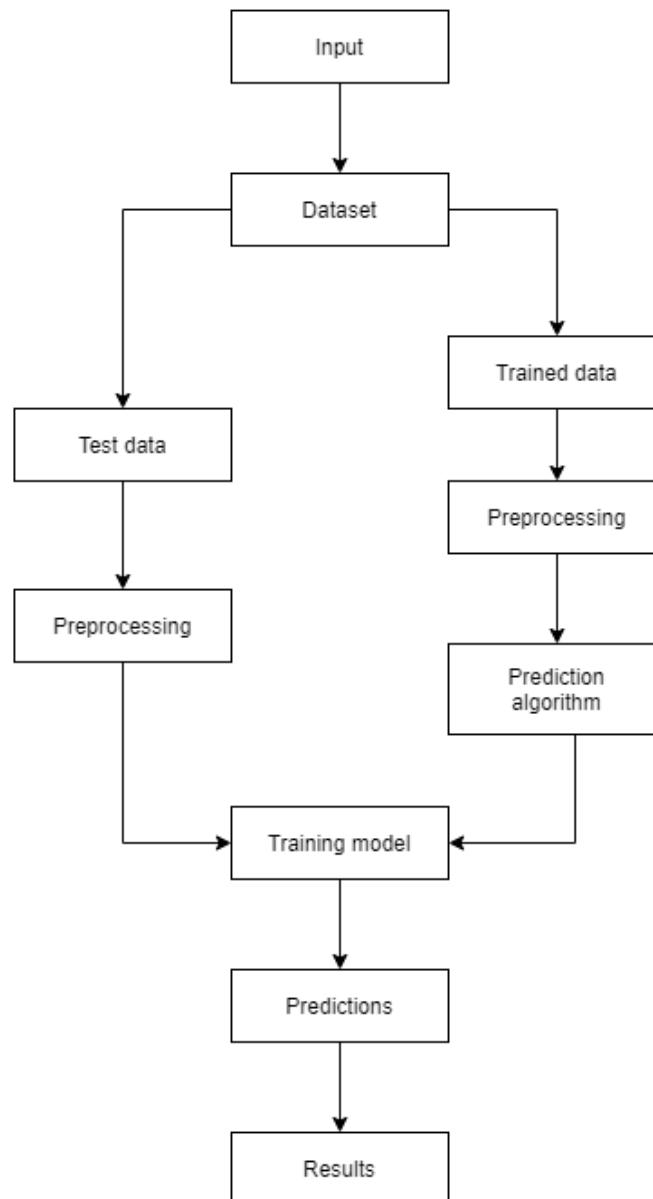


Fig. 7: Training and prediction

## 4.2 UML Diagrams

A UML diagram is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains graphical elements (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts.

The kind of the diagram is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines.

UML specification does not preclude mixing of different kinds of diagrams, e.g. to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced. At the same time, some UML Tools do restrict set of available graphical elements which could be used when working on specific type of diagram.

UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams.

Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

## 4.2.1 Use Case Diagram

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems.
- Goals that your system or application helps those entities (known as actors) achieve.
- The scope of your system.

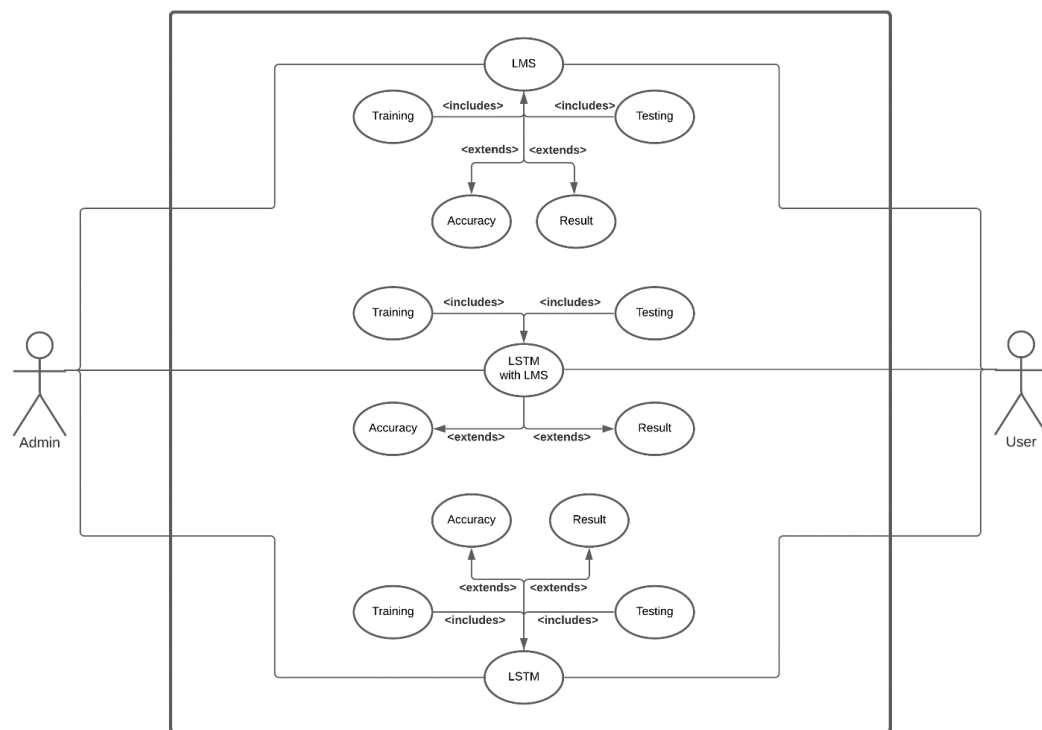


Fig. 8: Using LMS, LSTM and LSTM with LMS in the system

## 4.2.2 Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.

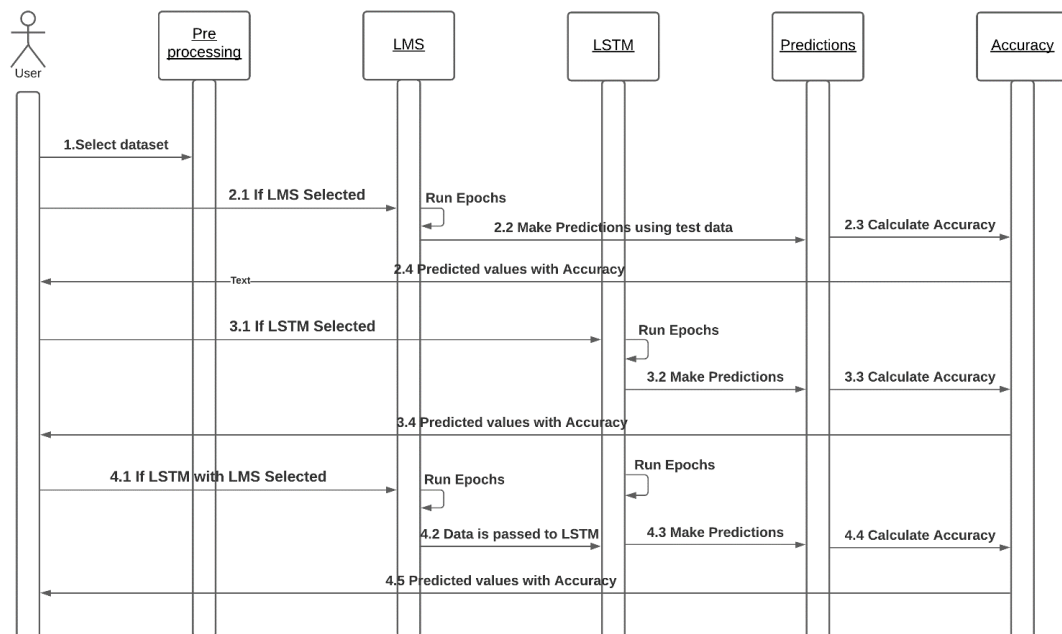


Fig. 9: Execution based on model selection



### 4.2.3 Activity Diagram

An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system.

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

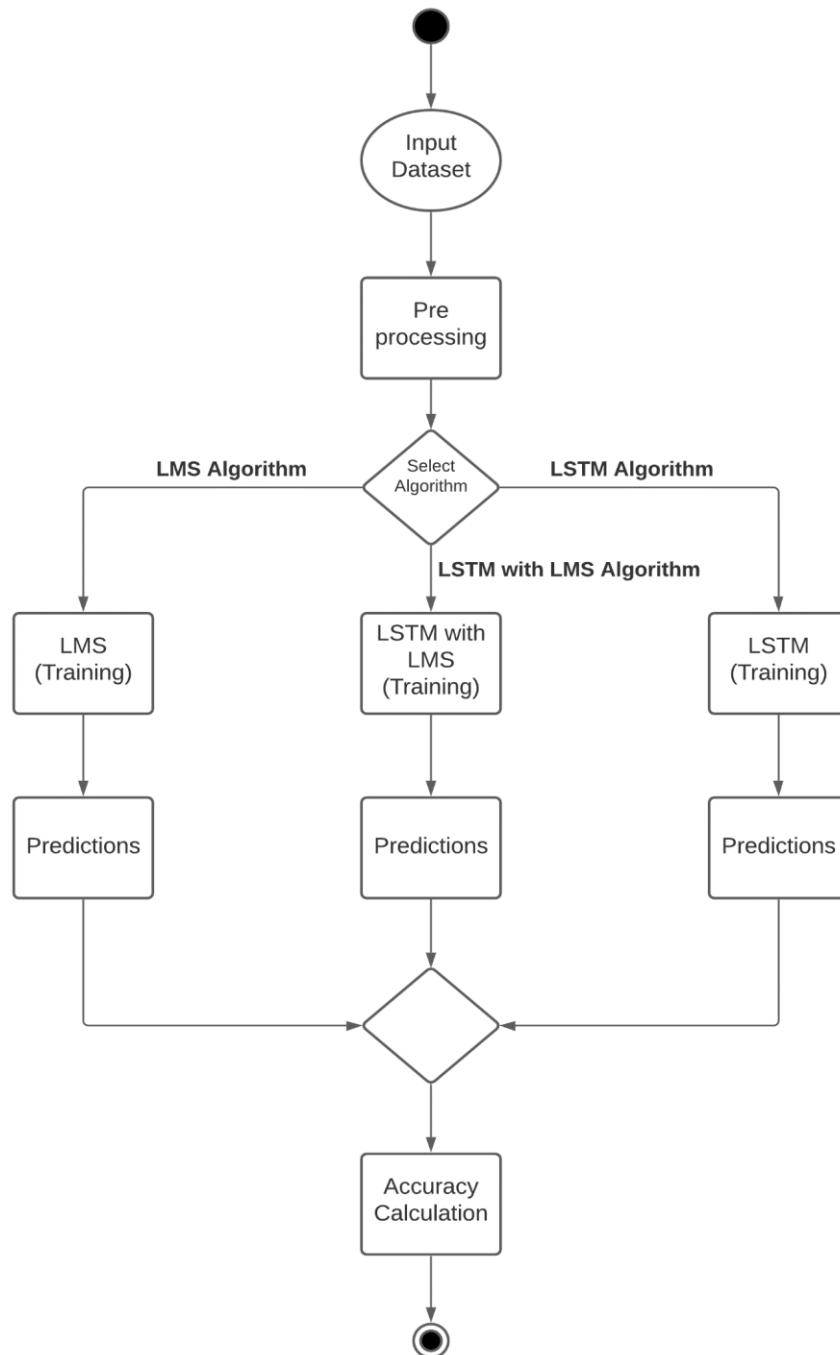


Fig. 10: Execution based on algorithm selection

## 4.2.4 Collaboration Diagram

Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.

The collaborations are used when it is essential to depict the relationship between the object. Both the sequence and collaboration diagrams represent the same information, but the way of portraying it quite different. The collaboration diagrams are best suited for analyzing use cases.

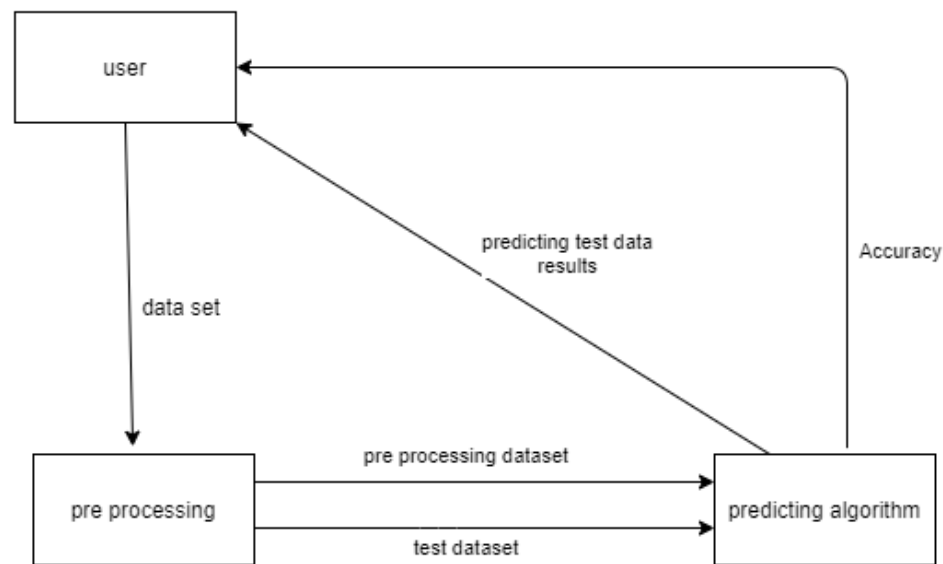


Fig. 11: Data transfer between modules

### 4.2.5 Flow Chart

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

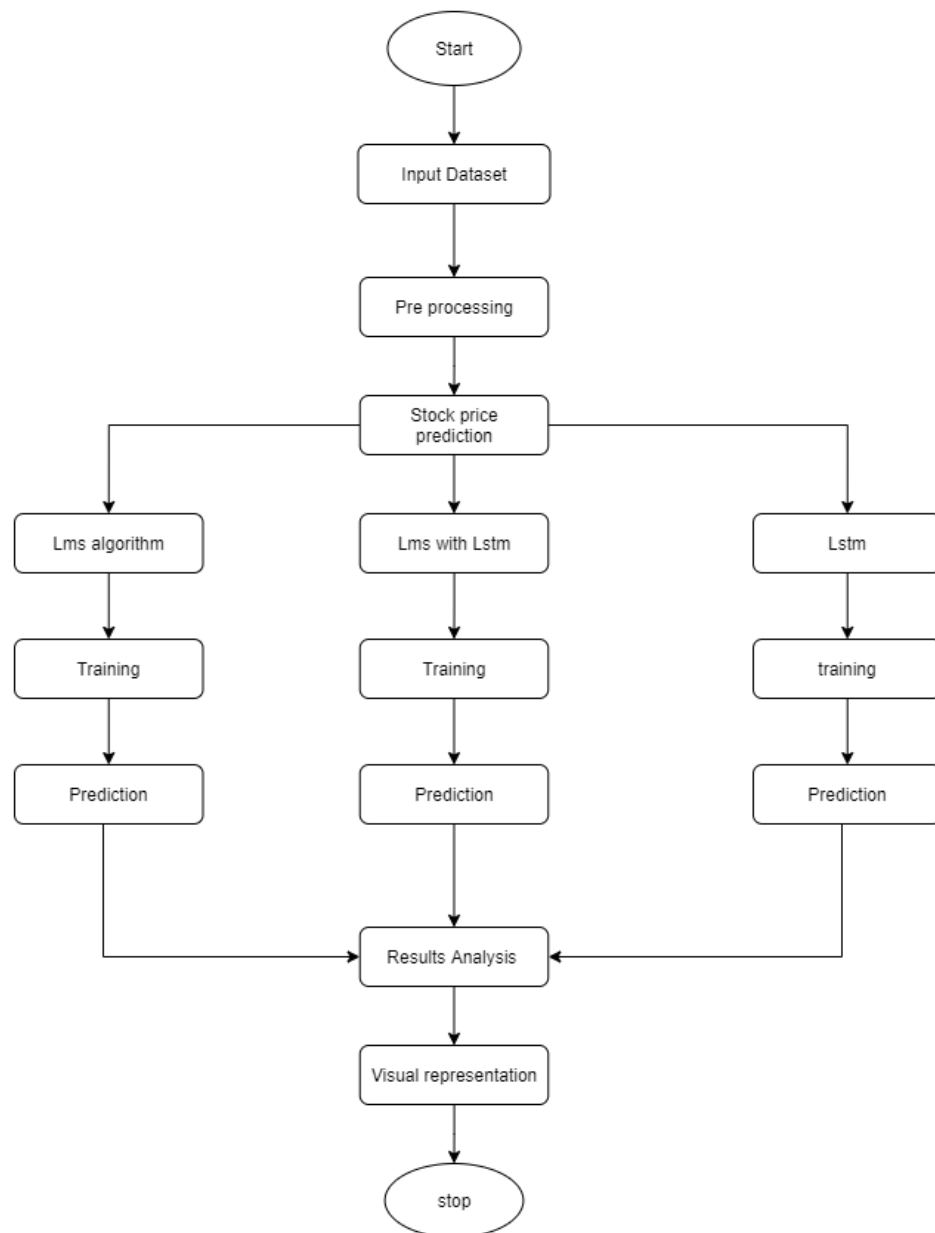


Fig. 12: Flow of execution

## 4.2.6 Component Diagram

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

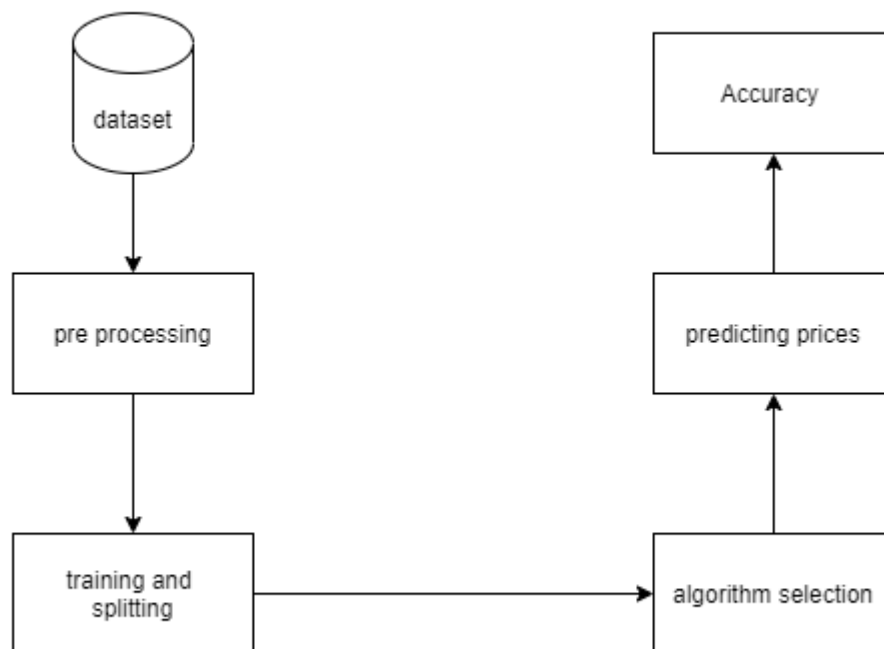


Fig. 13: Components present in the system

## CHAPTER 5

### EXPERIMENT ANALYSIS

#### 5.1 system configuration

This project can run on commodity hardware. We ran entire project on an Intel I5 processor with 8 GB Ram, 2 GB Nvidia Graphic Processor, It also has 2 cores which runs at 1.7 GHz, 2.1 GHz respectively. First part of the is training phase which takes 10-15 mins of time and the second part is testing part which only takes few seconds to make predictions and calculate accuracy.

##### 5.1.1 Hardware Requirements:

- RAM: 4 GB
- Storage: 500 GB
- CPU: 2 GHz or faster
- Architecture: 32-bit or 64-bit

##### 5.1.2 Software requirements

- Python 3.5 in Google Colab is used for data pre-processing, model training and prediction.
- Operating System: windows 7 and above or Linux based OS or MAC OS.

#### 5.2 Sample code

Normalization

```
def minmaxscaler(X, min, max):
    omax, omin = X.max(axis=0), X.min(axis=0)

    X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
    X_scaled = X_std * (max - min) + min

    return X_scaled, omax, omin

def inverse_scalar(X, omax, omin, min, max):
    X = X - min
    X = X / (max - min)

    p1 = X + omin
    p2 = omax - omin
    X = X * (omax - omin)
    X += omin

    return X
```

## Get required Columns

```
def getColumnData(df, cols):  
    print("Retriving", ' '.join(cols), "Columnn(s)")  
    return df[cols]  
  
def getRequiredColumns(df):  
    res = []  
    dateColName = None  
    closeColName = None  
  
    for col in df.columns:  
        if (('date' in col.lower()) or ('time' in col.lower())):  
            dateColName = col  
            break  
  
    for col in df.columns:  
        if ('open' in col.lower()):  
            res.append(col)  
            break  
  
    for col in df.columns:  
        if ('open' in col.lower()):  
            res.append(col)  
            break  
  
    for col in df.columns:  
        if ('low' in col.lower()):  
            res.append(col)  
            break  
  
    for col in df.columns:  
        if ('high' in col.lower()):  
            res.append(col)  
            break
```

```

for col in df.columns:
    if (('close' in col.lower()) and ('adj' not in col.lower()) and ('prev' not in col.lower())):
        res.append(col)
        closeColName = col
        break

for col in df.columns:
    if (('volume' in col.lower()) or ('turnover' in col.lower())):
        res.append(col)
        break

return res, dateColName, closeColName

```

## LMS

```

def LMS(df, pred_col, next_days, epochs, updateEpochs):
    print("LMS Training for", pred_col)

    ndf, omax, omin = minmaxscaler(df[pred_col], 1000, 2000)
    x = ndf.values

    tmp = []
    for i in x: tmp.append(i)

    x = np.array(tmp)

```

```

def lmsPred(x,l,u,N):
    xd = np.block([1, x]).T
    y=np.zeros((len(xd),1))

    xn = np.zeros((N+1,1))
    xn = np.matrix(xn)

    wn=np.random.rand(N+1,1)/10

    M=len(xd)
    for epoch in range(epochs):
        updateEpochs(epoch)
        print("epoch ", epoch+1, "/", epochs, sep='')

        for n in range(0,M):
            xn = np.block([[xd[n]], [xn[0:N]]])
            y[n]= np.matmul(wn.T, xn)

            if(n>M-1-1): e = 0;
            else: e=int(x[n]-y[n])

            wn = wn + 2*u*e*xn

        return y,wn;

x_train = x[:-next_days]
u = 2**(-30);

l=next_days;
N=100;

y,wn = lmsPred(x_train,l,u,N)

x = inverse_scalar(ndf, omax, omin, 1000, 2000)
y = inverse_scalar(y, omax, omin, 1000, 2000)

# plotGraph(cols=[x, y], title=pred_col, colors=['black', 'red'])

json = {
    "inputs": x,
    "outputs": y,
    "actual": x[-1:].values,
    "predicted": y[-1:]
}

return json

```



## LSTM

```
def LSTM_Cell(inputs, init_h, init_c, kernel, recurrent_kernel, bias,
              mask, time_major, go_backwards, sequence_lengths,
              zero_output_for_mask):

    input_length = inputs.shape[1]

    def operations(cell_inputs, cell_states):
        """Step function that will be used by Keras RNN backend."""
        h_tm1 = cell_states[0] # previous memory state
        c_tm1 = cell_states[1] # previous carry state

        z = K.dot(cell_inputs, kernel)
        z += K.dot(h_tm1, recurrent_kernel)
        z = K.bias_add(z, bias)

        z0, z1, z2, z3 = array_ops.split(z, 4, axis=1)

        i = nn.sigmoid(z0)
        f = nn.sigmoid(z1)
        c = f * c_tm1 + i * nn.tanh(z2)
        o = nn.sigmoid(z3)

        h = o * nn.tanh(c)
        return h, [h, c]

    last_output, outputs, new_states = K.rnn(
        operations,
        inputs, [init_h, init_c],
        constants=None,
        unroll=False,
        time_major=time_major,
        mask=mask,
        go_backwards=go_backwards,
        input_length=input_length,
        zero_output_for_mask=zero_output_for_mask
    )

    return (last_output, outputs, new_states[0], new_states[1])
```

```

class LSTM(recurrent.DropoutRNNCellMixin, recurrent.LSTM):
    def __init__(self,
                  units,
                  activation='tanh',
                  recurrent_activation='sigmoid',
                  use_bias=True,
                  bias_initializer='zeros',
                  unit_forget_bias=True,
                  return_sequences=False,
                  **kwargs):

        super(LSTM, self).__init__(
            units,
            return_sequences=return_sequences,
            activation=activation,
            recurrent_activation=recurrent_activation,
            use_bias=use_bias,
            bias_initializer=bias_initializer,
            unit_forget_bias=unit_forget_bias,
            **kwargs)

    def call(self, inputs, mask=None, training=None, initial_state=None):
        row_lengths = inputs.shape[0]
        inputs, initial_state, _ = self._process_inputs(inputs, initial_state, None)

        lstm_kwargs = {
            'inputs': inputs,
            'init_h': initial_state[0],
            'init_c': initial_state[1],
            'kernel': self.cell.kernel.read_value(),
            'recurrent_kernel': self.cell.recurrent_kernel.read_value(),
            'bias': self.cell.bias.read_value(),
            'mask': mask,
            'time_major': self.time_major,
            'go_backwards': self.go_backwards,
            'sequence_lengths': row_lengths,
            'zero_output_for_mask': self.zero_output_for_mask
        }

        (last_output, outputs, new_h, new_c) = LSTM_Cell(**lstm_kwargs)

        output = last_output

        return output

```

## EpochPrintingCallback

```
class EpochPrintingCallback(keras.callbacks.Callback):  
    def __init__(self, updateEpochs):  
        self.updateEpochs = updateEpochs  
  
    def on_epoch_end(self, epoch, logs=None):  
        print(epoch)  
        self.updateEpochs(epoch)
```

## LSTM Algorithm

```
def LSTMAlgorithm(fileName, train_size, epochs, updateEpochs):  
    df = pd.read_csv('./datasets/' + fileName + '.csv')  
    cols, dateColName, trade_close_col = getRequiredColumns(df)  
  
    scaling_data_frame = df.filter(cols)  
  
    scaler = MinMaxScaler(feature_range=(0,1))  
    scaled_Data = scaler.fit_transform(scaling_data_frame)  
    scaled_data_frame = pd.DataFrame(data=scaled_Data, index=[df[trade_close_col]], columns=cols)  
  
    stock_close_data = df.filter([trade_close_col])  
    stock_close_dataset = stock_close_data.values  
  
    trainingDataLength = math.ceil( len(stock_close_dataset) * train_size )  
  
    scaler = MinMaxScaler(feature_range=(0,1))  
    scaledData = scaler.fit_transform(stock_close_dataset)  
  
    StockTrainData = scaledData[0:trainingDataLength , :]  
  
    Xtrain = []  
    Ytrain = []  
  
    for i in range(60, len(StockTrainData)):  
        Xtrain.append(StockTrainData[i-60:i, 0])  
        Ytrain.append(StockTrainData[i, 0])  
  
    Xtrain = np.array(Xtrain)  
    Ytrain = np.array(Ytrain)  
  
    Xtrain = np.reshape(Xtrain, (Xtrain.shape[0], Xtrain.shape[1], 1))
```

```

testingData = scaledData[trainingDataLength - 60: , :]

Xtest = []
Ytest = stock_close_dataset[trainingDataLength:, :]
for i in range(60, len(testingData)):
    Xtest.append(testingData[i-60:i, 0])

Xtest = np.array(Xtest)
Xtest = np.reshape(Xtest, (Xtest.shape[0], Xtest.shape[1], 1))

print("\n\nLSTM Algorithm for "+str(epochs)+" epochs")
model = Sequential()

neurons = 50

model.add(LSTM(neurons, return_sequences=True, input_shape= (Xtrain.shape[1], 1)))
model.add(LSTM(neurons, return_sequences=False))

model.add(Dense(25))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mse')

history_data = model.fit(Xtrain, Ytrain,
                        batch_size=50, epochs=epochs, validation_split=0.2,
                        verbose=0, callbacks=[EpochPrintingCallback(updateEpochs=updateEpochs)])
print("Saving Model----->")

model.save('pretrained/' + fileName + ".h5")

predictions = model.predict(Xtest)
predictions = scaler.inverse_transform(predictions)

training = stock_close_data[:trainingDataLength]
validation = pd.DataFrame(df[trade_close_col][trainingDataLength:], columns=['Close'])

validation['Predictions'] = predictions

real = validation['Close'].values
pred = validation['Predictions'].values
n = len(pred)

```

```

accuracy = 0
for i in range(n):
    accuracy += (abs(real[i] - pred[i])/real[i])*100

print('For', epochs, "epochs")
print("Accuracy:", 100 - accuracy/n, end='\n\n')

return model

```

### getPredictionsFromModel

```

def getPredictionsFromModel(fileName, train_size):
    df = pd.read_csv('./datasets/' + fileName + '.csv')
    cols, dateColName, trade_close_col = getRequiredColumns(df)

    model = tf.keras.models.load_model('./pretrained/' + fileName + '.h5')

    scaling_data_frame = df.filter(cols)

    scaler = MinMaxScaler(feature_range=(0,1))
    scaled_Data = scaler.fit_transform(scaling_data_frame)
    scaled_data_frame = pd.DataFrame(data=scaled_Data, index=[df[trade_close_col]], columns=cols)

    stock_close_data = df.filter([trade_close_col])
    stock_close_dataset = stock_close_data.values

    trainingDataLength = math.ceil( len(stock_close_dataset) * train_size )

    scaler = MinMaxScaler(feature_range=(0,1))
    scaledData = scaler.fit_transform(stock_close_dataset)

    StockTrainData = scaledData[0:trainingDataLength , :]

    Xtrain = []
    Ytrain = []

    for i in range(60, len(StockTrainData)):
        Xtrain.append(StockTrainData[i-60:i, 0])
        Ytrain.append(StockTrainData[i, 0])

    Xtrain = np.array(Xtrain)
    Ytrain = np.array(Ytrain)

    Xtrain = np.reshape(Xtrain, (Xtrain.shape[0], Xtrain.shape[1], 1))

```

```

testingData = scaledData[trainingDataLength - 60: , :]

Xtest = []
Ytest = stock_close_dataset[trainingDataLength:, :]
for i in range(60, len(testingData)):
    Xtest.append(testingData[i-60:i, 0])

Xtest = np.array(Xtest)
Xtest = np.reshape(Xtest, (Xtest.shape[0], Xtest.shape[1], 1))

# predictions

predictions = model.predict(Xtest)
predictions = scaler.inverse_transform(predictions)

training = stock_close_data[:trainingDataLength]
validation = pd.DataFrame(df[trade_close_col][trainingDataLength:], columns=['Close'])

validation['Predictions'] = predictions

real = validation['Close'].values
pred = validation['Predictions'].values
n = len(pred)

accuracy = 0
for i in range(n):
    accuracy += (abs(real[i] - pred[i])/real[i])*100

# print('For', epochs, "epochs")
accuracyPercentage = 100 - accuracy/n
# print("Accuracy:", , end='\n\n')

trainingDates = df[dateColName].iloc[:trainingDataLength]
trainingDates = list(trainingDates.values)
trainingData = list(training[trade_close_col].values)

realData = list(real)

predictionDates = df[dateColName].iloc[trainingDataLength:]
predictionDates = list(predictionDates.values)
predictionData = list(pred)

for i in range(len(trainingData)): trainingData[i] = float(trainingData[i])
for i in range(len(predictionData)): predictionData[i] = float(predictionData[i])

```

```

    json = {
        "training": {
            "dates": trainingDates,
            "data": trainingData
        },
        "predictions": {
            "dates": predictionDates,
            "realData": realData,
            "predictedData": predictionData,
            "accuracy": accuracyPercentage
        }
    }

    return json

```

### Flask Code

```

app = Flask("Stock Price Prediction")
CORS(app)

df = None
cols, dateColName, closeColName = None, None, None
train_size = 0.75
totalEpochs = 2
session = {
    "training": {
        "status": "ready",
        "fileUploaded": False,
        "fileName": None,
        "totalEpochs": totalEpochs
    },
    "prediction": {
        "status": "ready",
        "preTrainedModelNames": None
    }
}

def updateEpochs(epoch):
    global session

    session['training']['epochs'] = epoch + 1

from api import *

```

```

@app.route("/")
def index():
    return "Welcome to Stock Price Prediction API"

@app.route("/upload", methods=['POST', 'GET'])
def upload():
    if (request.method == "POST"):
        global session, df, cols, dateColName, closeColName

        df = pd.read_csv(request.files['file'])
        cols, dateColName, closeColName = getRequiredColumns(df)
        # print(df[[dateColName] + cols].head().values)
        dfColVals = []
        dfDateVals = []
        dfCloseVals = []
        for row in df[[dateColName] + cols].values:
            dfColVals.append(list(row))
            dfCloseVals.append(row[4])
            dfDateVals.append(row[0])

        session['training']['fileUploaded'] = True
        session['training']['fileName'] = request.files['file'].filename[: -4]
        session['training']['cols'] = [dateColName] + cols
        session['training']['dfColVals'] = dfColVals
        session['training']['dfCloseVals'] = dfCloseVals
        session['training']['dfDateVals'] = dfDateVals

        return session['training']
    else:
        return "This API accepts only POST requests"

@app.route("/startTraining", methods=['POST', 'GET'])
def startTraining():
    if (request.method == "POST"):
        global session, df

        fileName = request.form['fileName']

        df.to_csv('datasets/' + fileName + '.csv')

        session['training']['status'] = "training"
        session['training']['epochs'] = 0

```



```

        model = LSTMAlgorithm(fileName, train_size, totalEpochs, updateEpochs=updateEpochs)

        session['training']['status'] = "trainingCompleted"

        return session['training']
    else:
        return "This API accepts only POST requests"

@app.route("/trainingStatus", methods=['POST', 'GET'])
def trainingStatus():
    if (request.method == "POST"):
        return session['training']
    else:
        return "This API accepts only POST requests"

# Prediction Page

@app.route("/getPreTrainedModels", methods=['POST', 'GET'])
def getPreTrainedModels():
    if (request.method == "POST"):
        global session

        files = glob.glob("./pretrained/*.H5")

        for i in range(len(files)):
            files[i] = files[i][13:-3]

        session['prediction']['preTrainedModelNames'] = files

        return session['prediction']
    else:
        return "This API accepts only POST requests"

@app.route("/getPredictions", methods=['POST', 'GET'])
def getPredictions():
    if (request.method == "POST"):
        global session

        modelName = request.form['modelName']
        session['prediction']['modelName'] = modelName

```

```

        modelData = getPredictionsFromModel(modelName, train_size)
        session['prediction']['modelData'] = modelData

        return session['prediction']
    else:
        return "This API accepts only POST requests"

```

```

if __name__ == '__main__':

    debug = False
    port = 7676

    app.run(
        debug=debug,
        port=port
    )

```

## **Frontend Pages**

### Common Header

```

<head>
    <title>Stock Price Prediction</title>

    <meta charset="utf-8">
    <link rel="icon" href="../favicon.ico" type="image/gif" sizes="16x16">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link rel="stylesheet" href="../static/bootstrap.min.css">
    <script src="../static/jquery.min.js"></script>
    <script src="../static/bootstrap.min.js"></script>

    <script src="../static/chart.js"></script>

    <link rel="stylesheet" href="../header.css">

```

```

<script>

    var pages = {
        "home": "home.html",
        "training": "training.html",
        "predictions": "predictions.html",
        "team": "team.html",
        "contactus": "contactus.html"
    }

    function changePage(e) {
        var page = e.getAttribute("page");
        window.location = pages[page];
    }

</script>

</head>

```

## Common Navbar

```

<div id="header">
    <div class="container-fluid">
        <div class="row">
            <div class="col-sm-4 header-title">Stock Price Prediction</div>
            <div class="col-sm-8">
                <div class="navbar">
                    <div onclick="changePage(this)" page="home" class="nav-item active">Home</div>
                    <div onclick="changePage(this)" page="training" class="nav-item">Training</div>
                    <div onclick="changePage(this)" page="predictions" class="nav-item">Predictions</div>
                    <div onclick="changePage(this)" page="team" class="nav-item">Team</div>
                    <div onclick="changePage(this)" page="contactus" class="nav-item">Contact us</div>
                </div>
            </div>
        </div>
    </div>
</div>

```

## Home page

```
<link rel="stylesheet" href="./home.css">

<div class="welcomeBanner">
  
  <div class="text">
    <div class="title">Stock Price Prediction</div>
    <div class="subtitle">A place for better predictions</div>
  </div>
</div>

<div class="container-fluid someStocks">
  <div class="title">Some Stocks</div>
  <div class="row">
    <div class="col-sm-3"></div>
    <div class="col-sm-3"></div>
    <div class="col-sm-3"></div>
    <div class="col-sm-3"></div>
  </div>
</div>
```

## Training

```
<link rel="stylesheet" href="./training.css">

<div class="upload">
  <button onclick="$('.selectFile').click()">New Training Data</button>
  <input class="selectFile" type="file" />
</div>
<center>
  <div class="container-fluid datasetProperties">
    <div class="placeholder">
      <div class="layer">
        <div class="layerName">Dataset</div>
        <div class="layerDesc">Click on "New Training Data" button to load the dataset</div>
      </div>
    </div>
    <div class="filename">Filename: <input class="fileNameInput" type="text" /></div>
    <div class="row">
      <div class="col-sm-7">
        <div class="dfHead"></div>
      </div>
      <div class="col-sm-5 closePriceGraphHolder">
        <canvas id="closePriceGraph"></canvas>
      </div>
    </div>
    <button class="startTraining">Start Training</button>
  </div>
</center>
```

```

<center>
  <div class="container-fluid trainingProgress">
    <div class="placeholder">
      <div class="layer">
        <div class="layerName">Training</div>
        <div class="layerDesc">Click on "Start Training" button to start the training</div>
      </div>
    </div>
    <div class="inProgress">
      <div class="layer">
        <div class="layerName">Training</div>
        
        <div class="layerDesc">Training</div>
      </div>
    </div>
    <div class="trainingCompleted">
      <div class="layer">
        <div class="layerName">Training Completed</div>
        <div class="layerDesc">Now you can see the predictions in "Predictions" tab</div>
      </div>
    </div>
  </div>
</center>

```

```

<script>

  var checkTrainingStatusVar = null;

  function loaddfHead(cols, tableData) {
    var head = `<thead><tr>`;
    for (var col in cols) {
      head += `<th scope="col">` + cols[col] + `</th>`;
    }
    head += `</tr></thead>`;

    var body = `<tbody>`;
    for (var row in tableData) {
      var rowHTML = `<tr>`;
      for (col in tableData[row]) {
        rowHTML += `<td>` + tableData[row][col] + `</td>`;
      }
      rowHTML += `</tr>`;
      body += rowHTML;
    }
    body += `</tbody>`;

    var table = `<table style="height: 100px!important" class="table table-hover">` + head + body + `
    $('<table>').html(table);
  }

```

```

function plotGraph(chartId, dfDateVals, rawData, showAnimation) {
  var data = [];
  var animation = false;

  for (let i = 0; i < rawData.length; i++) {
    data.push({x: i, y: rawData[i]});
  }

  if (showAnimation) {
    const totalDuration = 3000;
    const delayBetweenPoints = totalDuration / data.length;
    const previousY = (ctx) => ctx.index === 0 ? ctx.chart.scales.y.getPixelForValue(100) : ctx.
    animation = {
      x: {
        type: 'number',
        easing: 'linear',
        duration: delayBetweenPoints,
        from: NaN, // the point is initially skipped
        delay(ctx) {
          if (ctx.type !== 'data' || ctx.xStarted) {
            return 0;
          }
          ctx.xStarted = true;
          return ctx.index * delayBetweenPoints;
        }
      },
      y: {
        type: 'number',
        easing: 'linear',
        duration: delayBetweenPoints,
        from: previousY,
        delay(ctx) {
          if (ctx.type !== 'data' || ctx.yStarted) {
            return 0;
          }
          ctx.yStarted = true;
          return ctx.index * delayBetweenPoints;
        }
      }
    };
  }
}

```

```

const config = {
  type: 'line',
  data: {
    datasets: [{
      borderColor: "#3aa4eb",
      borderWidth: 1,
      radius: 0,
      data: data,
    }]
  },
  options: {
    animation,
    interaction: {
      intersect: false
    },
    plugins: {
      legend: false
    },
    scales: {
      x: {
        type: 'category',
        labels: dfDateVals
      }
    }
  }
};

```

```

var myChart = new Chart(
  document.getElementById(chartId),
  config
);

return myChart;

```

```

}

```

```

function loadDataset(res) {
  $('.fileNameInput').val(res.fileName);
  loaddfHead(res.cols, res.dfColVals);

  $('.closePriceGraphHolder').html(`<canvas id="closePriceGraph"></canvas>`);
  plotGraph('closePriceGraph', res.dfDateVals, res.dfCloseVals, false);

  $('.datasetProperties .placeholder').hide();
}

```

```

function checkTrainingStatus() {
    checkTrainingStatusVar = setInterval(() => {
        $.ajax({
            url: 'http://localhost:7676/trainingStatus',
            method: 'post',
            success: (res) => {
                console.log(res);
                $('#trainingProgress .inProgress .layerDesc').html("Training " + Math.round((res.epo
                if (res.status !== "training") {
                    stopTrainingStatusCheck();
                }
            }
        });
    }, 1000);
}

```

```

function stopTrainingStatusCheck() {
    clearInterval(checkTrainingStatusVar);

    $('#trainingProgress .inProgress').hide();
    $('#trainingProgress .trainingCompleted').show();
}

```

```

$('#selectFile').change(() => {
    var file = $('#selectFile')[0].files[0];

    var formData = new FormData();
    formData.append('file', file);

    $.ajax({
        url: 'http://localhost:7676/upload',
        type: "POST",
        processData: false,
        contentType: false,
        data: formData,
        success: (res) => {
            console.log(res);
            loadDataset(res);
        }
    });
});

```



```

$($('.startTraining').click(() => {
    $($('.trainingProgress .placeholder').hide());
    $($('.trainingProgress .trainingCompleted').hide());

    $.ajax({
        url: 'http://localhost:7676/startTraining',
        method: "POST",
        data: {"fileName": $($('.fileNameInput').val())},
        success: (res) => {

        }
    });

    checkTrainingStatus();
});

$(document).ready(() => {
    $.ajax({
        url: 'http://localhost:7676/trainingStatus',
        method: 'post',
        success: (res) => {
            if (res.status == "training") {
                loadDataset(res);

                $($('.trainingProgress .placeholder').hide());
                $($('.trainingProgress .trainingCompleted').hide());

                checkTrainingStatus();
            }
        }
    });
});

```

</script>

## Predictions

<link rel="stylesheet" href="./predictions.css">

<div class="predictionsPage">

<center>

<div class="btn-group selectModel">

<button type="button" data-toggle="dropdown" >Select Model</button>

<div class="dropdown-menu dropdown-menu-center"></div>

</div>

</center>

```

<center>
  <div class="container-fluid predictionResults">
    <div class="placeholder">
      <div class="layer">
        <div class="layerName">Model</div>
        <div class="layerDesc">Select the model by clicking "Select Model" button to sh
      </div>
    </div>
    <div class="inProgress">
      <div class="layer">
        <div class="layerName">Model</div>
        
        <div class="layerDesc">Loading model from server</div>
      </div>
    </div>
    <div class="row">
      <div class="modelName"></div>
      <div class="col-sm-8 predictionGraphHolder"><canvas id="predictionGraph"></canvas></div>
      <div class="col-sm-4">
        <div class="accuracyPercentage"></div>
        <div class="realPredTable"></div>
      </div>
    </div>
  </div>
</center>
</div>

```

```
<script>
```

```

function selectModel(modelName) {
  $('.placeholder').hide();
  $('.inProgress').show();

  $('.predictionGraphHolder').html(`<canvas id="predictionGraph"></canvas>`);
  $('.realPredTable').html('');

  $.ajax({
    url: 'http://localhost:7676/getPredictions',
    method: 'post',
    data: {"modelName": modelName},
    success: (res) => {
      console.log(res);

      $('.modelName').html(res.modelName + " Dataset");

      $('.inProgress').hide();
    }
  });
}

```

```

plotGraph(
    'predictionGraph',
    res.modelData.training.dates,
    res.modelData.training.data,
    res.modelData.predictions.dates,
    res.modelData.predictions.predictedData,
    true
);

var dates = res.modelData.predictions.dates;
var realData = res.modelData.predictions.realData;
var predictedData = res.modelData.predictions.predictedData;

var head =
<thead>
    <tr>
        <th scope="col">Date</th>
        <th scope="col">Actual</th>
        <th scope="col">Predicted</th>
    </tr>
</thead>
`;

var len = realData.length;
var body = `<tbody>`;

for (var row=0; row<len; row++) {
    body += `
    <tr>
        <td>`+dates[row]+`</td>
        <td>`+realData[row].toFixed(2)+`</td>
        <td>`+predictedData[row].toFixed(2)+`</td>
    </tr>
    `;
}
body += `</tbody>`;

var table = `<table style="height: 100px!important" class="table ta
$(`.realPredTable`).html(table);

$(`.accuracyPercentage`).html("Accuracy: "+res.modelData.prediction
}
});
}

```

```

$(document).ready(() => {
    $('.inProgress').hide();

    $.ajax({
        url: 'http://localhost:7676/getPreTrainedModels',
        method: 'post',
        success: (res) => {
            console.log(res);

            var preTrainedModelNames = res.preTrainedModelNames;
            var options = ``;

            for (var i in preTrainedModelNames) {
                options += `<a class="dropdown-item" onclick="selectModel('${preTrainedModelNames[i]}')"`;
            }

            $('.selectModel .dropdown-menu').html(options);
        }
    });
});
</script>

```

### 5.3 Sample Input and Output:

Google

Attribute Name	Min	Max
Open	87.74	1005.49
Low	86.37	996.62
High	89.29	1008.61
Close	87.58	1004.28

Table 1: Min and Max of columns in Google Dataset

## Nifty50

Attribute Name	Min	Max
Open	7735.15	12932.5
Low	7511.1	12819.35
High	8036.95	12948.85
Close	7610.25	12938.25

Table 2: Min and Max of columns in Nifty50 Dataset

## Reliance

Attribute Name	Min	Max
Open	205.5	3298.0
Low	197.15	3141.3
High	219.5	3298.0
Close	203.2	3220.85

Table 3: Min and Max of columns in Reliance Dataset

## Sample Input:

	Trade High	Trade Low	Trade Open	Trade Volume	Trade Count
0	214.23	214.14	214.15	1022241	2274
1	214.38	214.14	214.15	582984	1902
2	214.37	214.18	214.37	705964	1943
3	214.30	214.16	214.29	430066	1321
4	214.20	214.09	214.18	444761	1599

Table 4: Sample input

Sample Output:

Trade Close = 214.07

## 5.4 Website Pages

### Home Page

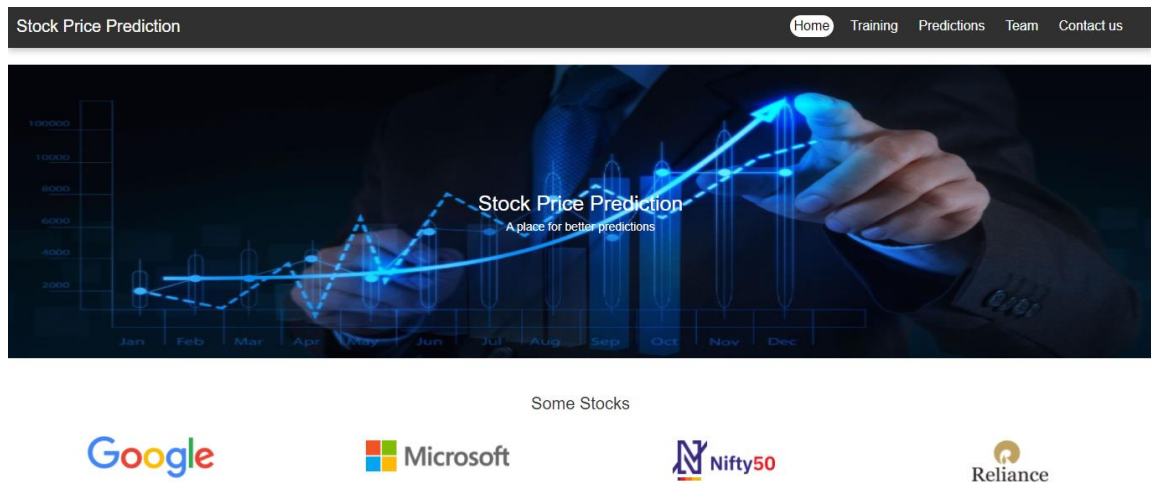


Fig. 14: Home page

### Training Page

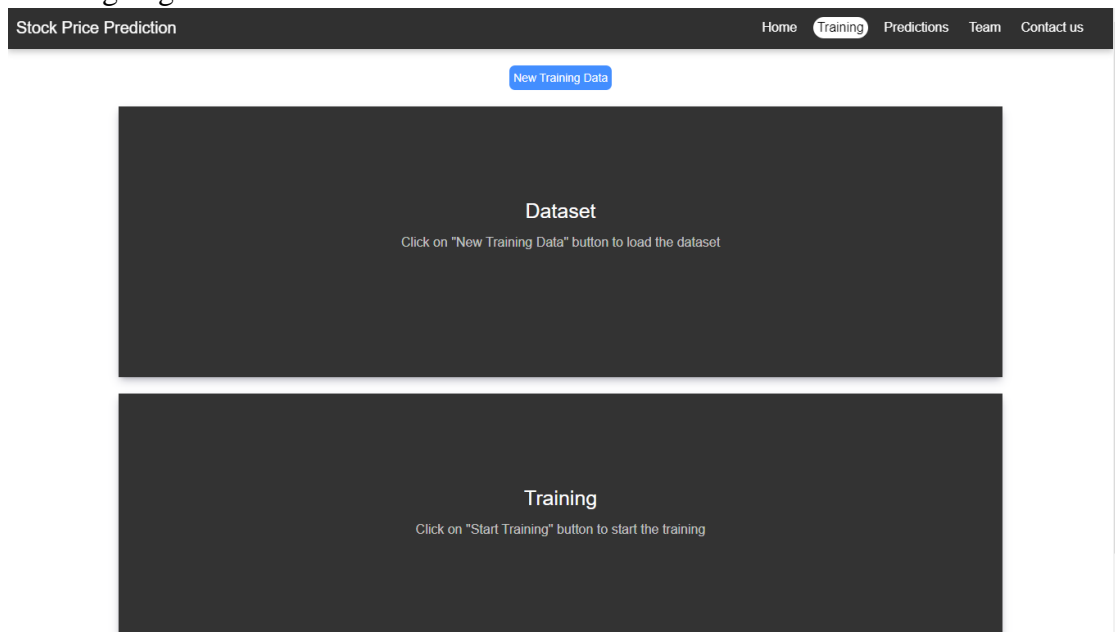


Fig. 15: Training page

# Training Page: After Selecting the dataset

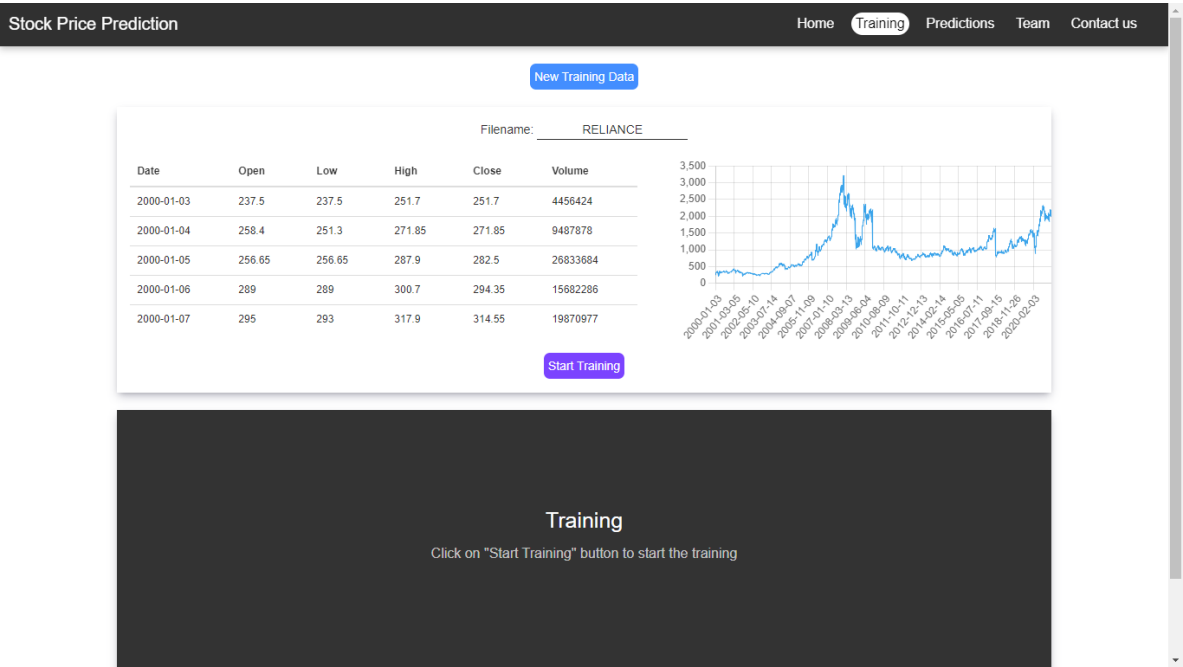


Fig.16: Training Page: After Selecting the dataset

# Training Page: While Training

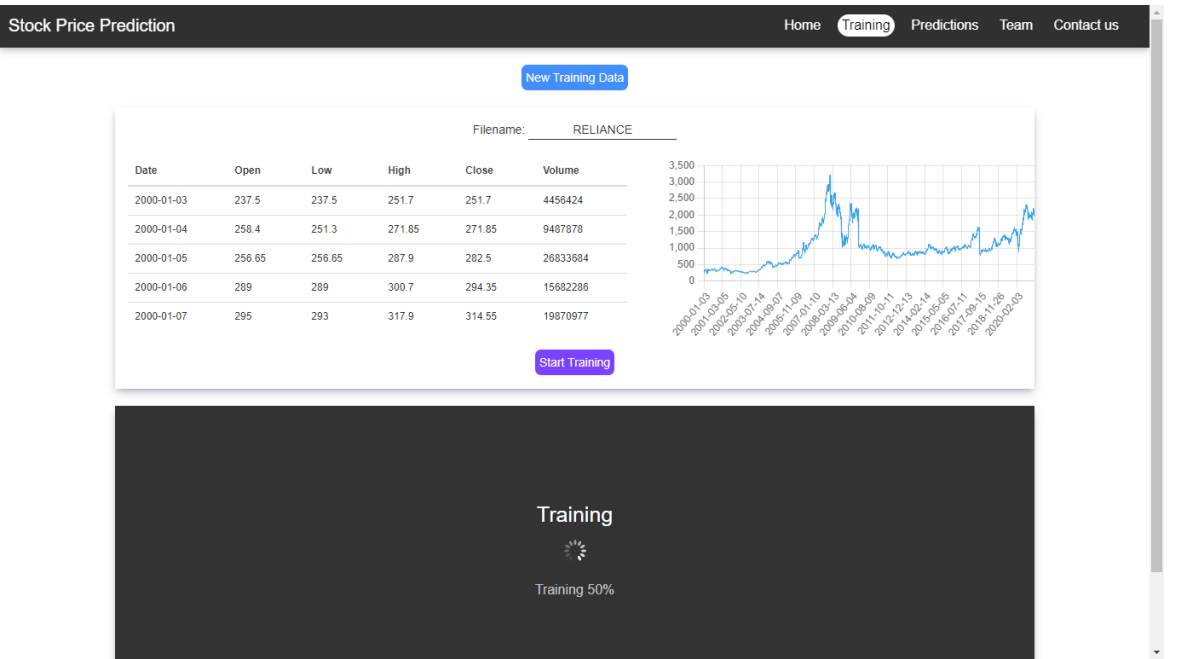


Fig.17: Training Page: While Training

# Training Page: Training Completed

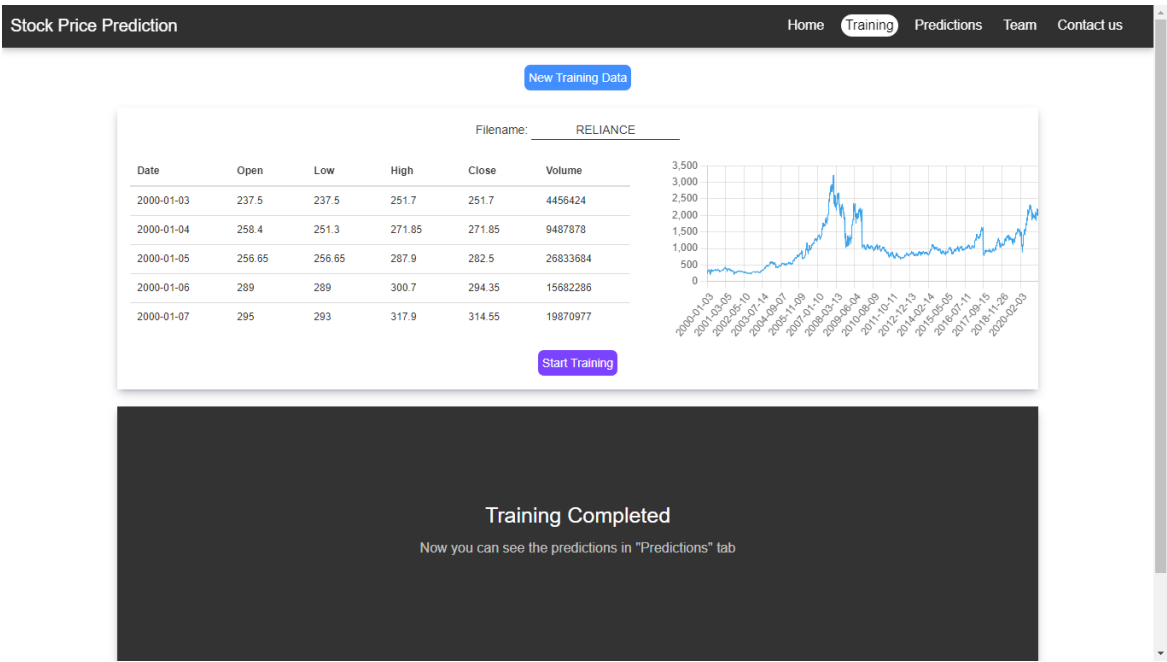


Fig. 18: Training Page: Training Completed

# Predictions Page

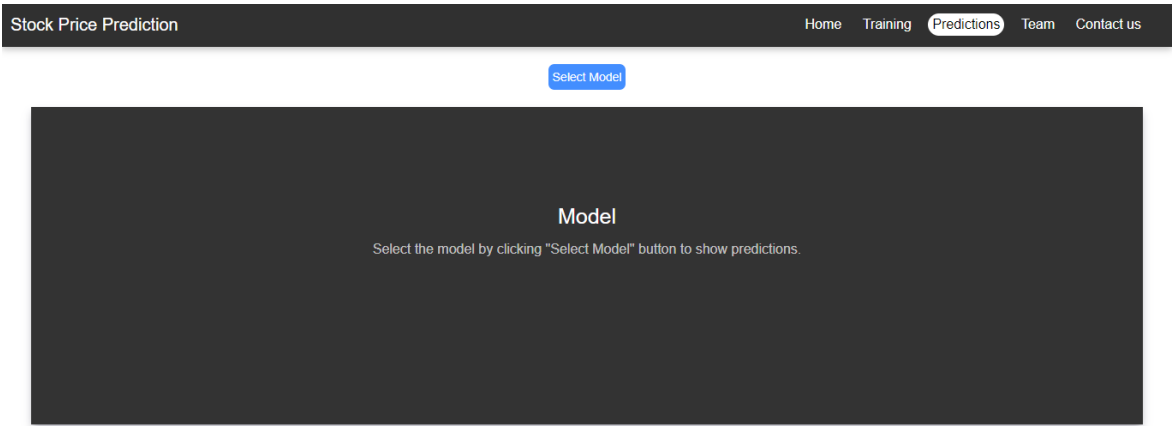


Fig. 19: Predictions Page



Predictions Page: After selecting the model



Fig. 20: Predictions Page: After selecting the model

Team Page

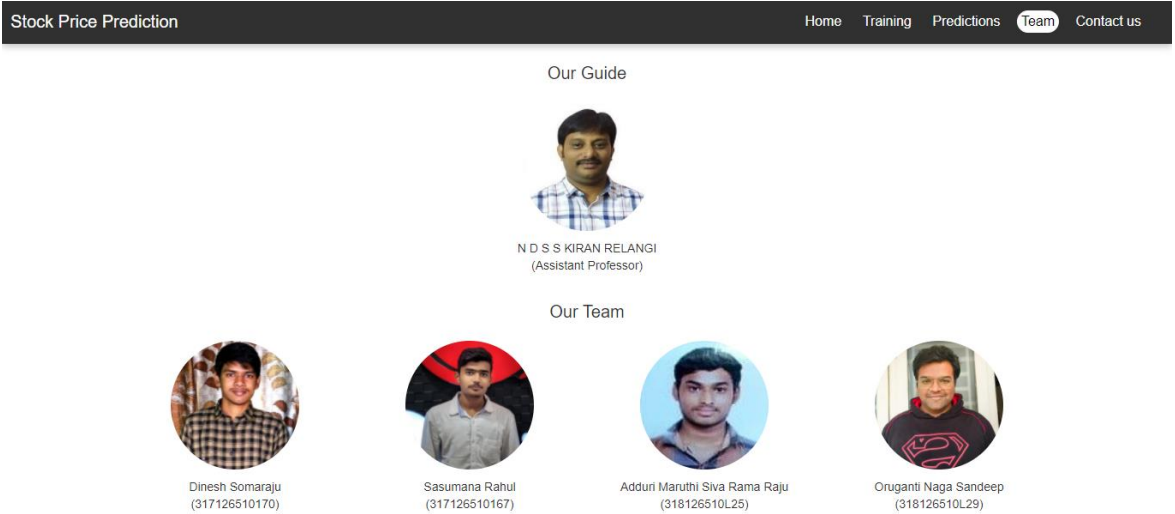


Fig. 21: Team Page

## 5.5 Performance Measure

### 5.5.1 LSTM

#### 5.5.1.1 Google

epochs	Accuracy	MSE	RMSE
10	93.00717	207.6578	14.41034
20	94.01166	156.3873	12.50549
30	95.64188	105.3248	10.26279
40	95.59026	99.17409	9.958619
50	96.99466	62.24641	7.88964

epochs	Accuracy
100	98.28213337528945
200	97.63336589796519
300	96.94409289369247
400	97.35454469043535

Table 5: Epochs for Google Dataset using LSTM

#### 5.5.1.2 Nifty50

epochs	Accuracy	MSE	RMSE
10	97.154	201835	449.26
20	95.2489	438515	662.204
30	97.4571	159737	399.671
40	97.8633	95549	309.11
50	97.9285	106734	326.702

epochs	Accuracy
100	97.97517139027555
200	98.30422315197787
300	92.95956921171869
400	98.67452775485742

Table 6: Epochs for Nifty50 Dataset using LSTM

#### 5.5.1.3 Reliance

epochs	Accuracy	MSE	RMSE
10	96.25328	4839.56908	69.56701
20	97.63885	2653.12783	51.50852
30	98.19937	1650.33374	40.6243
40	98.13572	1616.9295	40.21106
50	98.37254	1361.80983	36.90271

epochs	Accuracy
100	96.49200483894309
200	98.56496342868206
300	98.57297238982146
400	97.90036066587572

Table 7: Epochs for Reliance Dataset using LSTM

## 5.5.2 LSTM with LMS

### 5.5.2.1 Google

epochs	Accuracy	MSE	RMSE	epochs	Accuracy
10	92.5615	339.549	18.4269	100	95.50810593088478
20	94.2892	219.856	14.8276	200	93.24950439506634
30	94.6971	169.259	13.01	300	94.68220175615014
40	95.1746	141.106	11.8788	400	96.29794053164949
50	94.747	161.208	12.6968	500	95.589282359809

Table 8: Epochs for Google Dataset using LSTM with LMS

### 5.5.2.2 Nifty50

epochs	Accuracy	MSE	RMSE	epochs	Accuracy
10	90.14171	1.50E+06	1225.58	100	97.67512047219317
20	94.41587	5.52E+05	742.9043	200	91.36568721761229
30	94.54524	5.47E+05	739.3578	300	92.21746083762834
40	96.65602	2.68E+05	517.9008	400	88.3763795882347
50	96.79688	2.50E+05	500.3757	500	91.26283879244312

Table 9: Epochs for Nifty50 Dataset using LSTM with LMS

### 5.5.2.3 Reliance

epochs	Accuracy	MSE	RMSE	epochs	Accuracy
10	95.283656	8079.78008	89.887597	100	97.41168889605405
20	95.055813	7370.14221	85.849532	200	97.44153787870181
30	96.530505	4622.22982	67.986983	300	97.72196960171793
40	95.594117	5847.60569	76.469639	400	97.75577851463954
50	96.681513	3858.11477	62.113724	500	97.52291451371063

Table 10: Epochs for Reliance Dataset using LSTM with LMS

## **CHAPTER 6**

### **CONCLUSION AND FUTURE WORK**

#### **6.1 Conclusion**

In this project, we are predicting closing stock price of any given organization, we developed a web application for predicting close stock price using LMS and LSTM algorithms for prediction. We have applied datasets belonging to Google, Nifty50, TCS, Infosys and Reliance Stocks and achieved above 95% accuracy for these datasets.

#### **6.2 Future work**

- We want to extend this application for predicting cryptocurrency trading.
- We want to add sentiment analysis for better analysis.

## REFERENCES

Datasets: [Nifty 50 data](#), [tw\\_spydata\\_raw\(trading data\)](#)



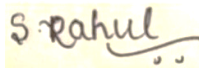

- [1] Stock Price Prediction Using LSTM on Indian Share Market by Achyut Ghosh, Soumik Bose<sup>1</sup>, Giridhar Maji, Narayan C. Debnath, Soumya Sen
- [2] S. Selvin, R. Vinayakumar, E. A. Gopalkrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in International Conference on Advances in Computing, Communications and Informatics, 2017.
- [3] Murtaza Roondiwala, Harshal Patel, Shraddha Varma, "Predicting Stock Prices Using LSTM" in Undergraduate Engineering Students, Department of Information Technology, Mumbai University, 2015.
- [4] Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin, "An innovative neural network approach for stock market prediction", 2018
- [5] Ishita Parmar, Navanshu Agarwal, Sheirsh Saxena, Ridam Arora, Shikhin Gupta, Himanshu Dhiman, Lokesh Chouhan Department of Computer Science and Engineering National Institute of Technology, Hamirpur – 177005, INDIA - Stock Market Prediction Using Machine Learning.
- [6] Pranav Bhat Electronics and Telecommunication Department, Maharashtra Institute of Technology, Pune. Savitribai Phule Pune University - A Machine Learning Model for Stock Market Prediction.
- [7] Anurag Sinha Department of computer science, Student, Amity University Jharkhand Ranchi, Jharkhand (India), 834001 - Stock Market Prediction Using Machine Learning.
- [8] V Kranthi Sai Reddy Student, ECM, Sreenidhi Institute of Science and Technology, Hyderabad, India - Stock Market Prediction Using Machine Learning.
- [9] Asset Durmagambetov currently works at the mathematics, CNTFI. Asset does research in Theory of Computation and Computing in Mathematics, Natural Science, Engineering and Medicine. Their current project is 'The Riemann Hypothesis-Millennium Prize Problems' - stock market predictions.

- [10] Mariam Moukalled Wassim El-Hajj Mohamad Jaber Computer Science Department American University of Beirut - Automated Stock Price Prediction Using Machine Learning.
- [11] Manh Ha Duong Boriss Siliverstovs June 2006 - The Stock Market and Investment.
- [12] Dharmaraja Selvamuthu , Vineet Kumar and Abhishek Mishra Department of Mathematics, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India - Indian stock market prediction using artificial neural networks on tick data.
- [13] Lufuno Ronald Marwala A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering - Forecasting the Stock Market Index Using Artificial Intelligence Techniques.
- [14] Xiao-Yang Liu<sup>1</sup> Hongyang Yang, Qian Chen<sup>4</sup>, Runjia Zhang Liuqing Yang Bowen Xiao Christina Dan Wang Electrical Engineering, <sup>2</sup>Department of Statistics, <sup>3</sup>Computer Science, Columbia University, <sup>3</sup>AI4Finance LLC., USA, Ion Media Networks, USA, Department of Computing, Imperial College, <sup>6</sup>New York University (Shanghai) - A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance.
- [15] Pushpendu Ghosh, Ariel Neufeld, Jajati Keshari Sahoo Department of Computer Science & Information Systems, BITS Pilani K.K.Birla Goa campus, India  
bDivision of Mathematical Sciences, Nanyang Technological University, Singapore  
cDepartment of Mathematics, BITS Pilani K.K.Birla Goa campus, India - Forecasting directional movements of stock prices for intraday trading using LSTM and random forests.
- [16] Xiao Ding, Kuo Liao, Ting Liu, Zhongyang Li, Junwen Duan Research Center for Social Computing and Information Retrieval Harbin Institute of Technology, China - Event Representation Learning Enhanced with External Commonsense Knowledge.
- [17] Huicheng Liu Department of Electrical and Computer Engineering Queen's University, Canada - Leveraging Financial News for Stock Trend Prediction with Attention-Based Recurrent Neural Network.
- [18] Hyeon Kyu Choi, B.A Student Dept. of Business Administration Korea University Seoul, Korea = Stock Price Correlation Coefficient Prediction with ARIMA-LSTM Hybrid Model.

- [19] M. Nabipour Faculty of Mechanical Engineering, Tarbiat Modares University, 14115-143 Tehran, Iran; Mojtaba.nabipour@modares.ac.ir - Deep Learning for Stock Market prediction.
- [20] Lavanya Ra SRM Institute of Science and Technology | SRM · Department of Computer Science - Stock Market Prediction.
- [21] M. Mekayel Anik · M. Shamsul Arefin (B) Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong, Bangladesh - An Intelligent Technique for Stock Market Prediction.

**Batch no: 9C**

**Team members:**

			
Somaraju Dinesh	Adduri Maruthi Siva Rama Raju	Sasumana Rahul	Oruganti Naga Sandeep
(317126510170)	(318126510L25)	(317126510167)	(318126510L29)



**(N D S S KIRAN RELANGI)**



# Stock Price Prediction Using LSTM on Indian Share Market

Achyut Ghosh<sup>1</sup>, Soumik Bose<sup>1</sup>, Giridhar Maji<sup>2</sup>, Narayan C. Debnath<sup>3</sup>,  
Soumya Sen<sup>1</sup>

<sup>1</sup>A.K. Chowdhury School of I.T., University of Calcutta

<sup>2</sup>Asansol Polytechnic, Asansol, India

<sup>3</sup>Department of Software Engineering, Eastern International University, Vietnam  
achyutghosh06@gmail.com, 1994bolusoumik@gmail.com,  
giridhar.maji@gmail.com, narayan.debnath@eiu.edu.vn,  
iamsoumyasen@gmail.com

## Abstract

Predicting stock market is one of the most difficult tasks in the field of computation. There are many factors involved in the prediction – physical factors vs. physiological, rational and irrational behavior, investor sentiment, market rumors, etc. All these aspects combine to make stock prices volatile and very difficult to predict with a high degree of accuracy. We investigate data analysis as a game changer in this domain. As per efficient market theory when all information related to a company and stock market events are instantly available to all stakeholders/market investors, then the effects of those events already embed themselves in the stock price. So, it is said that only the historical spot price carries the impact of all other market events and can be employed to predict its future movement. Hence, considering the past stock price as the final manifestation of all impacting factors we employ Machine Learning (ML) techniques on historical stock price data to infer future trend. ML techniques have the potential to unearth patterns and insights we didn't see before, and these can be used to make unerringly accurate predictions. We propose a framework using LSTM (Long Short-Term Memory) model and companies' net growth calculation algorithm to analyze as well as prediction of future growth of a company.

## 1 Introduction

Data analysis have been used in all business for data-driven decision making. In share market, there are many factors that drive the share price, and the pattern of the change of price is not regular. This is why it is tough to take a robust decision on future price. Artificial Neural Network (ANN) has the capability to learn from the past data and make the decision over future. Deep learning networks



such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) etc. works great with multivariate time series data. We train our model from the past stock data and calculate the future price of that stock. This future price use to calculate the future growth of a company. Moreover, we found a future growth curve from different companies. Thus we can analyze and investigate the similarity of one company's future curve over another. Stock price of a listed company in a stock exchange varies every time an order is placed for sell or buy and a transaction completes. An exchange collects all sell bids with expected price per stock (normally it is more than the price paid while bought by the investor) and all buy bids with or without a price limit (normally an investor expects the future price of the stock will be more than the current price he is paying now) and a buy-sell transaction is committed when both bids have a match i.e. selling bid price is same with buying bid price of some buy-bid Fama in 1970 [1] proposed efficient market hypothesis which says that in an efficient market (where all events are known to all stakeholders as an when it happens) the effect all market events are already incorporated in stock prices hence it is not possible to predict using past events or prices. The stock price of a company depends on many intrinsic as well as extrinsic attributes. Macro-economic conditions too play an important role in growth or decline of a sector as a whole. Some of the intrinsic factors could be company's net profit, liabilities, demand stability, competition in market, technically advanced assembly line, surplus cash for adverse situations, stakes in raw material supplier and finished product distributors etc. Those factors that are beyond the control of the company such as crude oil price, dollar exchange rate, political stability, government policy decision etc. come under extrinsic attribute. Many researchers have tried using the historical stock prices as the basis for time series analysis to forecast future stock prices. Many different statistical models were applied since long like moving average (MA), autoregression (AR), weighted moving average, ARIMA, CARIMA etc. Later some non linear models were also tried like GARCH. Recently different neural network models, evolutionary algorithms were being applied for stock prediction with success. Deep neural networks like CNN, RNN are also used with different parameter settings and features. In this paper we shall explore a special type of RNN known as LSTM to predict future company growth based on past stock prices.

## 2 Indian Stock Market Overview

Almost every country has one or more stock exchanges, where the shares of listed companies can be sold or bought. It is a secondary market place. When a company first lists itself in any stock exchange to become a public company, the promoter group sells substantial amount of shares to public as per government norms. During incorporation of a company shares are bought by promoter groups or institutional investors in a primary market. Once promoter offloads major portion of the shares to public retail investors, then those could be traded in secondary market i.e. in stock exchanges. In India the BSE (Bombay Stock Exchange) and the NSE (National Stock Exchange) are the two most active stock exchange. The BSE has around 5000 listed companies where as NSE had around 1600. Both the exchange has similar trading mechanism and market open time, closing time and settlement process. Stock exchanges helps individual investors to take part in the share market and allows to buy even a single share of some listed company with the help of a trading account and demat account. These online markets have revolutionized the Indian investment arena along with government initiative like tax benefit on equity investment, National Pension Scheme (NPS) investing in share market etc. Due to continuous reduction in bank interest rates and increasing inflation middle class investors are moving towards equity market from the safe haven of fixed deposits. All these have helped to grow the capitalization of both the exchanges.

### 3 Related Studies

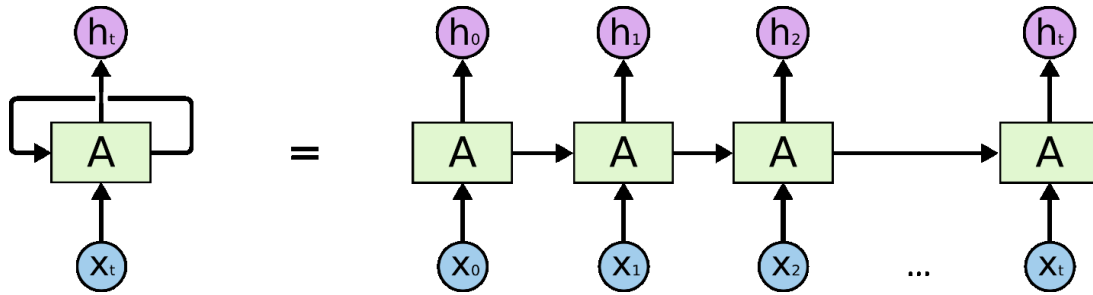
There are lots of research work in stock market prediction as well as in LSTM. Almost every data mining and prediction techniques were applied for prediction of stock prices. Many different features and attributes were used for the same purpose. There are three main categories of stock market analysis and prediction such as (a) Fundamental analysis, (b) Technical analysis and (c) Time series analysis. Most of the stock forecasting techniques with time series data normally use either a linear such as AR, MA, ARIMA, ARMA, CARIMA, etc. [1],[2] or non-linear models (ARCH, GARCH, ANN, RNN, LSTM, etc.). Authors in [3] have analyzed many different macro-economic factors by designing a data warehouse that affects share price movement such as crude oil price, exchange rate, gold price, bank interest rate, political stability, etc. Researchers in [4] employed frequent itemset mining technique to find a lagged correlation between price movement between different sectorial index in Indian share market. Roondiwala et al. in [5] has used RNN-LSTM model on NIFTY-50 stocks with 4 features (high/close/open/low price of each day). They have used 21 days window to predict the next day price movement. A total of 5 years data has been used for prediction and RMSE as error metric to minimize with backpropagation.

Kim et al. in [6] proposed a model, ‘the feature fusion long short-term memory-convolutional neural network (LSTM-CNN) model’. They have used CNN to learn the features from stock chart images. They found that the candlestick charts are the best candidate for predicting future stock price movement. Next they employed LSTM and fed with historical price data. They have tested on minute-wise stock price and used 30 minute sliding window to forecast 35<sup>th</sup> minute price. They have tested on S&P 500 ETF data with stock price and trade volume using CNN. They use the CNN and LSTM individually on different representation of the same data and then used the combined feature fusion model for the same purpose. It is observed that the combined model outperforms individual models with less prediction error. Thus this work establishes the fact that different representation of the same data (raw stock price and trade volume and stock chart image) with combined models where each individual model is optimized for separate data format can learn more intrinsic data dynamics and features which is analogous to looking on the same object from different perspective angles that gives new insight. Hiransha et al. in their paper [7], employed three different deep learning network architectures such as RNN, CNN and LSTM to forecast stock price using day wise past closing prices. They have considered two company from IT sector (TCS and Infosys) and one from Pharma sector (Cipla) for experiment. The uniqueness of the study is that they have trained the models using data from a single company and used those models to predict future prices of five different stocks from NSE and NYSE (New York Stock Exchange). They argued that linear models try to fit the data to the model but in deep networks underlying dynamic of the stock prices are unearthed. As per their results CNN outperformed all other models as well as classical linear models. The DNN could forecast NYSE listed companies even though the model has learned from NSE dataset. The reason could be the similar inner dynamics of both the stock exchanges. Gers & Schmidhuber proposed a variation of LSTM by introducing “peephole connections” [18]. In this model the gate layers can look into the cell state. In another case the model coupled forget and input gates. In this case, decision to add new information or to forget it is taken together. It forgets only when it needs to input something in its place. This architecture inputs new values to the cell state when it forgets anything older. Cho, et al. [19] proposed another popular LSTM variation known as the Gated Recurrent Unit (GRU). It aggregates both the forget and input gates into an “update gate.” The cell state and hidden state are merged along with a few other minor modifications to make the final model more simple than the original LSTM. Due to the above reason this model is becoming popular day by day. These are by no means an exhaustive list of modified-LSTMs. There are many other variants such as Depth Gated LSTMs by Yao, et al. [20]. Koutnik, et al. [21] proposed ‘Clockwork RNNs’ to tackle long-term dependencies in a completely different manner.

## 4 LSTM Architecture

### 4.1 An overview of Recurrent Neural Network (RNN)

In a classical neural network, final outputs seldom act as an input for the next step but if we pay attention to a real-world phenomenon, we observe that in many situations our final output depends not only on the external inputs but also on earlier output. For example, when humans read a book, understanding of each sentence depends not only on the current list of words but also on the understanding of the previous sentence or on the context that is created using past sentences. Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. This concept of 'context' or 'persistence' is not available with classical neural networks. Inability to use context-based reasoning becomes a major limitation of traditional neural network. Recurrent neural networks (RNN) are conceptualized to alleviate this limitation. RNN are networked with feedback loops within to allow persistence of information. The Figure 1 **Error! Reference source not found.** shows a simple RNN with a feedback loop and its unrolled equivalent version side by side.



**Figure 1:** An unrolled recurrent neural network

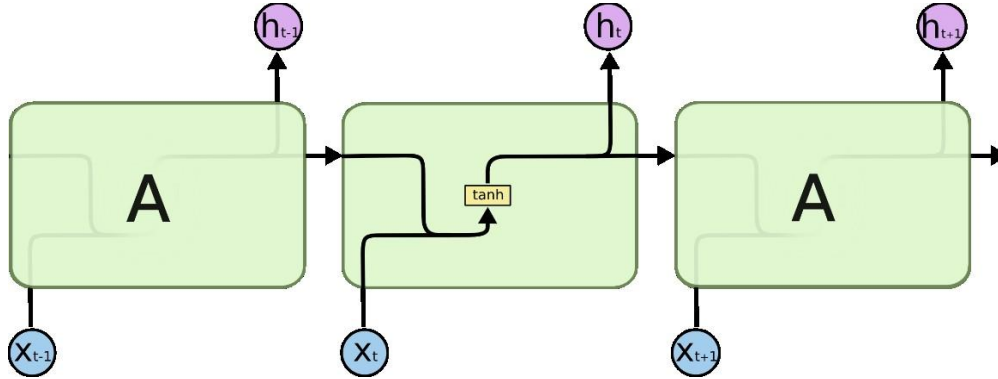
Initially (at time step  $t$ ) for some input  $x_t$  the RNN generates an output of  $h_t$ . In the next time step ( $t+1$ ) the RNN takes two input  $x_{t+1}$  and  $h_t$  to generate the output  $h_{t+1}$ . A loop allows information to be passed from one step of the network to the next.

RNNs are not free from limitations though. When the 'context' is from near past it works great towards the correct output. But when an RNN has to depend on a distant 'context' (i.e. something learned long past) to produce correct output, it fails miserably. This limitation of the RNNs was discussed in great detail by Hochreiter [8] and Bengio, et al. [9]. They also traced back to the fundamental aspects to understand why RNNs may not work in long-term scenarios. The good news is that the LSTMs are designed to overcome the above problem.

### 4.2 LSTM Networks

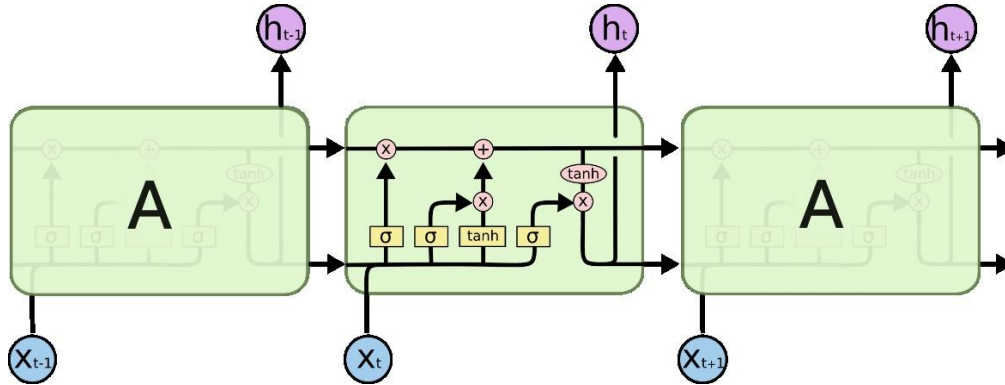
Hochreiter & Schmidhuber [10] introduced a special type of RNN which is capable of learning long term dependencies. Later on many other researchers improved upon this pioneering work in [11] [12] [13] [14]. LSTMs are perfected over the time to mitigate the long-term dependency issue. The evolution and development of LSTM from RNNs are explained in [15] [16].

Recurrent neural networks are in the form of a chain of repeating modules of the neural network. In standard RNNs, this repeating module has a simple structure like a single *tanh* layer as shown in Figure 2.



**Figure 2:** The repeating module in a standard RNN contains a single layer

LSTMs follow this chain-like structure, however the repeating module has a different structure. Instead of having a single neural network layer, there are four layers, interacting in a very special way as shown in Figure 3.



**Figure 3:** The repeating module in an LSTM contains four interacting layers

In Figure 3, every line represents an entire feature vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denotes its content being copied and the copies going to different locations.

### 4.3 The Working of LSTM

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. The cell state is like a conveyor belt. This runs straight down the entire chain, having some minor linear interactions. LSTM has the ability to add or remove information to the cell state, controlled by structures called gates. Gates are used to optionally let information through. Gates are composed of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between 0 and 1, describing how much of each component should be let through. A value of 0 means “let nothing through,” while a value of 1 means “let everything through!” An LSTM has three of these gates, to protect and control the cell state.

The first step of LSTM is to decide what information are to be thrown out from the cell state. It is made by a sigmoid layer called the “forget gate layer.” It looks at  $h_{t-1}$  and  $x_t$ , and outputs a

number between 00 and 11 for each number in the cell state  $C_{t-1}$ . A 11 represents “completely keep this” while a 00 represents “completely remove this.”

In the next step it is decided what new information are going to be stored in the cell state. It has two parts. First, a sigmoid layer called the “input gate layer” decides which values are to be updated. Thereafter, a *tanh* layer creates a vector of new candidate values,  $\tilde{C}_t$ , that could be added to the state. In the next step, these two are combined to create an update to the state. It is now time to update the old cell state,  $C_{t-1}$ , into the new cell state  $C_t$ .

We multiply the old state by  $f_t$ . Then we add  $f_t * \tilde{C}_t$ . This is the new candidate values, scaled by how much we decide to update each state value.

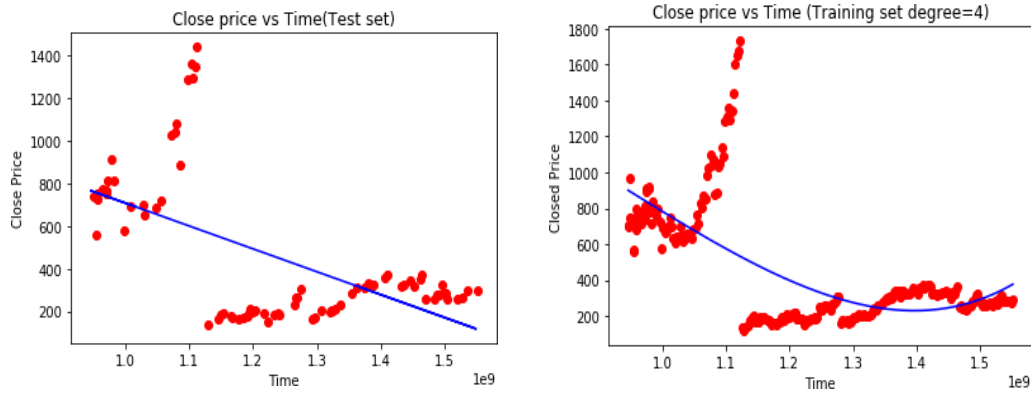
Finally, we need to decide on the output. The output will be a filtered version of the cell state. First, we run a sigmoid layer which decides what parts of the cell state we’re going to output. Then, we put the cell state through *tanh* (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

## 5 Proposed Framework to Forecast Share Price & Company Growth in Different Time Span

In this section, we shall first analyze some existing techniques and their merits to finally arrive at our methodology. Next, we shall discuss the algorithmic and implementation steps in detail. It is implemented in Python.

### 5.1 Analyzing Different Methods

Regression is one of the popular way to do the prediction of share prices. In figure 4 two figures on TCS share price using linear regression & polynomial regression of degree four are shown.



**Figure 4:** stock market closing prices of TCS over a time period and polynomial(degree 4) regression line

Regression is found not to be very much useful here to compute the error values. Also, we found a problem with curve fitting. The above graphs are showing a poor result in terms of curve fitting. This has a clear justification. For time series data, such as text, signals, stock prices, etc. LSTM is better suited to learn temporal patterns in deep neural networks. An LSTM solves the ‘vanishing gradient’ problem that exists in a RNN while learning long-term dependencies with time series dataset with the use of memory cell (states) and (input and forget) gates. So, LSTM may be a better option for future prediction of the company’s share price as well as growth.

## 5.2 Methodology

The purpose of our framework is to analyze which is the best time span to predict the future share price of a company from a particular sector. Our objective is to predict the future price and calculate the future growth of the company in the different time span. Then we analyze the prediction error for each company of different sector. Based on that we conclude which time span is best for future prediction of that particular sector.

We first predict the future closing price of 5 different companies from some pre-decided sectors with the help of LSTM. This prediction will be done on historical data & the future prediction will be done for 3-month, 6-month, 1 year & 3 years. In these four different time spans (3 & 6 months, 1 & 3 years), we calculate the growth of those companies. Then by analyzing the deviations of closing price for each time span, we took the resultant time span which has maximum growth, i.e. less error for the particular sector, e.g. companies A, B, C, D & E from a sector S1 has more growth in 3-months' time span of prediction then we draw an conclusion that for sector S1, our framework gives the best prediction for next 3-months for that particular sector. In our analysis, let's consider we are using the data for Months. Then the weight of a company is defined as:

$$\text{weight} = 1 / (P * (P+1) / 2)$$

In our case, month-wise weight ( $Y_i$ ) will be calculated using the following algorithm:

```

N:= M
weight := 1/(M*(M+1)/2) FOR i = 1 to M
Begin
Yi := weight * N ; /* Yi is the weight of previous ith month*/Q = Q - 1;
i := i + 1End
End FOR

```

Suppose the growth rate between different time periods is  $Gr_i$  where  $i=1$  to  $M$ , considering current year as 0<sup>th</sup> year. Therefore,  $Gr_i$  is the growth rate of  $(i-1)^{th}$  time period w.r.t its immediate earlier year i.e.  $i^{th}$  year. To maximize the impact of current growth over the growth of older year, we would develop a mathematical formula stated below. Suppose the growth rates of a company are  $Gr_1; Gr_2 \dots Gr_m$  respectively from present to  $M$  years earlier.

Then the Company Net Growth Rate (CNGR) by the following formula.

$$CNGR_j = Y_1 * Gr_1 + Y_2 * Gr_2 + \dots + Y_i * Gr_i + \dots + Y_p * Gr_m$$

Where  $CNGR_j$  is the Company Net Growth Rate of the  $j^{th}$  company (where  $j=1$  to  $m$ )

## 5.3 Implementation Steps

**Step1: Raw Stock Price Dataset:** Day-wise past stock prices of selected companies are collected from the BSE (Bombay Stock Exchange) official website.

**Step2: Pre-processing:** This step incorporates the following:

- Data discretization: Part of data reduction but with particular importance, especially for numerical data
- Data transformation: Normalization.

c) Data cleaning: Fill in missing values.

d) Data integration: Integration of data files. After the dataset is transformed into a clean dataset, the dataset is divided into training and testing sets so as to evaluate. Creating a data structure with 60 timesteps and 1 output.

**Step3: Feature Selection:** In this step, data attributes are chosen that are going to be fed to the neural network. In this study Date & Close Price are chosen as selected features.

**Step 4: Train the NN model:** The NN model is trained by feeding the training dataset. The model is initiated using random weights and biases. Proposed LSTM model consists of a sequential input layer followed by 3 LSTM layers and then a dense layer with activation. The output layer again consists of a dense layer with a linear activation function.

**Step5: Output Generation:** The RNN generated output is compared with the target values and error difference is calculated. The Backpropagation algorithm is used to minimize the error difference by adjusting the biases and weights of the neural network.

**Step 6: Test Dataset Update:** Step 2 is repeated for the test data set.

**Step 7: Error and companies' net growth calculation:** By calculating deviation we check the percentage of error of our prediction with respect to actual price.

**Step 8: Visualization:** Using Keras[21] and their function APIs the prediction is visualized.

**Step 9:** Investigate different time interval: We repeated this process to predict the price at different time intervals. For our case, we took 2-month dataset as training to predict 3-month, 6-month, 1 year & 3 years of close price of the share. In this different time span, we calculate the percentage of error in the future prediction. This would be different for different sectors. So, this will help to find a frame for the particular sector to predict future companies' net growth.

## 6 Results

The proposed LSTM based model is implemented using Python. In Table 1 the Error value for different companies belong to Banking Sector based on the historical data of 1 month, 3 month, 6 month, 1 Year, 3 Year span is shown.

**Table 1:** Error Value for Different Banks

Bank Names	1 month	3 month	6 month	1 year	3 year
SBI	93.30438	9.371283	19.5584	5.148866	0.830179
HDFC	532.8527	523.4962	162.8642	24.40721	0.987856
ICICI	71.80286	9.881709	10.76914	4.575525	0.863681
Avg Error	232.6533	180.9164	64.39726	11.3772	0.893905

**Table 2:** Error Value for Different Sectors

Sector	1 month	3 month	6 month	1 year	3 year
<b>IT</b>	39.56394	8.049353	1.48794	1.840666	0.782617
<b>Pharma</b>	250.7862	94.87654	29.48869	7.358529	0.903381
<b>FMCG</b>	426.7132	134.2102	60.45957	11.9643	0.874805
<b>Aviation</b>	291.025	35.08927	36.90103	30.97042	0.944595
<b>Bank</b>	232.6533	180.9164	64.39726	11.3772	0.893905

In the same way calculation is done for other sectors also based on the top level companies belong to that sector. The error values for the sector is shown in Table 2.

It has been observed from the result that for almost all the sectors the error level comes down drastically with the test data for longer periods. So we suggest to apply this LSTM based model to predict the share price on long time historical data.

## 7 Conclusions

In this paper, we analyze the growth of the companies from different sector and try to find out which is the best time span for predicting the future price of the share. So, this draws an important conclusion that companies from a certain sector have the same dependencies as well as the same growth rate. The prediction can be more accurate if the model will train with a greater number of data set.

Moreover, in the case of prediction of various shares, there may be some scope of specific business analysis. We can study the different pattern of the share price of different sectors and can analyze a graph with more different time span to fine tune the accuracy. This framework broadly helps in market analysis and prediction of growth of different companies in different time spans. Incorporating other parameters (e.g. investor sentiment, election outcome, geopolitical stability) that are not directly correlated with the closing price may improve the prediction accuracy.

## References

- [1]F. a. o. Eugene, "Efficient capital markets: a review of theory and empirical work," Journal of finance, vol. 25, no. 2, pp. 383-417, 1970.
- [2]Z. A. Farhath, B. Arputhamary and L. Arockiam, "A Survey on ARIMA Forecasting Using Time Series Model," Int. J. Comput. Sci. Mobile Comput, vol. 5, pp. 104-109, 2016.
- [3]S. Wichaidit and S. Kittitornkun, "Predicting SET50 stock prices using CARIMA (cross correlation ARIMA)," in 2015 International Computer Science and Engineering Conference (ICSEC), IEEE, 2015, pp. 1-4.
- [4]D. Mondal, G. Maji, T. Goto, N. C. Debnath and S. Sen, "A Data Warehouse Based Modelling Technique for Stock Market Analysis," International Journal of Engineering & Technology, vol. 3, no. 13, pp. 165-170, 2018.
- [5]G. Maji, S. Sen and A. Sarkar, "Share Market Sectoral Indices Movement Forecast with Lagged Correlation and Association Rule Mining," in International Conference on Computer Information



- Systems and Industrial Management, Bialystok, Poland, Sprigner, 2017, pp. 327-340.
- [6] M. Roondiwala, H. Patel and S. Varma, "Predicting stock prices using LSTM," *International Journal of Science and Research (IJSR)*, vol. 6, no. 4, pp. 1754-1756, 2017.
  - [7] T. Kim and H. Y. Kim, "Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data," *PloS one*, vol. 14, no. 2, p. e0212320, April 2019.
  - [8] S. Selvin, R. Vinayakumar, E. A. Gopalkrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in *International Conference on Advances in Computing, Communications and Informatics*, 2017.
  - [9] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen," Diploma, Technische Universität München, vol. 91, no. 1, 1991.
  - [10] Y. Bengio, P. Simard, P. Frasconi and others, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157-166, 1994.
  - [11] S. Hochreiter and J. Schmidhuber, "LSTM can solve hard long time lag problems," in *Advances in neural information processing systems*, NIPS, 1997, pp. 473-479.
  - [12] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 2, pp. 107-116, 1998.
  - [13] J. Schmidhuber, D. Wierstra, M. Gagliolo and F. Gomez, "Training recurrent networks by evolution," *Neural computation*, vol. 19, no. 3, pp. 757-779, 2007.
  - [14] L. Pasa and A. Sperduti, "Pre-training of recurrent neural networks via linear autoencoders," in *Advances in Neural Information Processing Systems*, NIPS, 2014, pp. 3572-3580.
  - [15] J. Chen and N. S. Chaudhari, "Segmented-memory recurrent neural networks," *IEEE transactions on neural networks*, vol. 20, no. 8, pp. 1267-1280, 2009.
  - [16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735-1780, 1997.
  - [17] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, MIT Press, 2018.
  - [18] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, IEEE, 2000, pp. 189-194.
  - [19] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
  - [20] K. Yao, T. Cohn, K. Vylomova, K. Duh and C. Dyer, "Depth-gated LSTM," *arXiv preprint arXiv:1508.03790*, 2015.
  - [21] J. Koutnik, K. Greff, F. Gomez and J. Schmidhuber, "A clockwork rnn," *arXiv preprint arXiv:1402.3511*, 2014.
  - [22] R. Kotikalapudi, "Keras Visualization Toolkit," [Online]. Available: <https://raghakot.github.io/keras-vis>. [Accessed 31 May 2019].

# Mukth Shabd Journal

ISSN NO: 2347-3150  
Scientific Journal Impact Factor – 4.6



## ACCEPTANCE LETTER TO AUTHOR

Dear Author,

With reference to your paper submitted “**STOCK PRICE PREDICTION USING LSTM**” we are pleased to accept the same for publication in **MSJ Volume X, Issue VI, JUNE 2021**.

**Manuscript ID: MSJ3831**

Please send the scanned Copyright form and Registration form along with bank receipt of an online maintenance/processing fee of **2000 INR** Per paper. Please note that the amount we are charging is very nominal & only an online maintenance and processing fee.

**The Fee includes:**

Online maintenance and processing charge.

No limitation of number of pages.

Editorial fee.

Taxes.

**Note:**

**Paper will be published online with in 24 hours after receiving the fee.**

\*Fee Paid for the Publication of the paper does not refund under any circumstances

\*In case of any query please do not hesitate to contact us at [submitmsj@gmail.com](mailto:submitmsj@gmail.com) Early reply is appreciated.

**DATE**

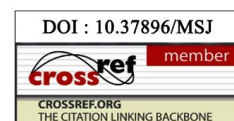
**17-JUNE-2021**

Sincerely,  
Best regards,

**M. ASHITOSH MEHATA**

<http://shabdbooks.com/>

**Sumit Ganguly**  
Editor-In-Chief  
MSJ  
[www.shabdbooks.com](http://www.shabdbooks.com)



## STOCK PRICE PREDICTION USING LSTM

S. Dinesh<sup>1</sup>, A.M.S. Rama Raju<sup>1</sup>, S. Rahul<sup>1</sup>, O. Naga Sandeep<sup>1</sup>, Mr. N D S S Kiran Relangi<sup>2</sup>

<sup>1</sup>Final year students of Department of CSE, Anil Neerukonda Institute of Technology and Sciences (A), Visakhapatnam-531162, India

<sup>2</sup>Assistant Professor at Department of CSE, Anil Neerukonda Institute of Technology and Sciences (A), Visakhapatnam-531162, India

---

### Abstract

Stock price prediction is the most significantly used in the financial sector. Stock market is volatile in nature, so it is difficult to predict stock prices. This is a time series problem. Stock price prediction is a difficult task where there are no rules to predict the price of the stock in the stock market. There are so many existing methods for predicting stock prices. The prediction methods are Logistic Regression Model, SVM, ARCH model, RNN, CNN, Backpropagation, Naïve Bayes, ARIMA model, etc. In these models, Long Short-Term Memory (LSTM) is the most suitable algorithm for time series problems. The main objective is to forecast the current market trends and could predict the stock prices accurately. We use LSTM recurrent neural networks to predict the stock prices accurately. The results show that prediction accuracy is over 93%.

**Keywords:** LSTM, CNN, ML, DL, Trade Open, Trade Close, Trade Low, Trade High.

### 1. INTRODUCTION

Stock market prediction means forecasting the current trends of a company and predict the value of stocks whether it's going up or down. Stock market is the place where a company's shares are traded. A stock is an investment in an institution where it represents ownership in a company. Stock market is a place where those stocks are purchased. Purchasing a stock of a company is owning a small share of an institution.

We are predicting the stock prices using the machine learning algorithm to develop a model which forecasts the stock price effectively based on the current market trends. We have used LSTM recurrent neural networks to predict the stock prices accurately. You would find two types of stocks, one of them was Intraday trading, which is known to us by the term day trading. Intraday trading is that which means all positions are squared-off before the market closes then and there and there would be no possibility of changing the ownership after the day end. LSTM's are very important, as they are very powerful in sequence prediction problems because they could store previous or past information. This is very important in stock prediction as we need to store and read the previous stock information as well to forecast the stock prices accurately in the future.

The rest of the paper is organized as follows. Section 2 introduces the research status of stock price prediction. Section 3 introduces the methodologies. Section 4 consists of the experimental results and the analysis of the results. Section 5 concludes the paper.

### 2. LITERATURE SURVEY

Stock price prediction can be predicted using AI and machine learning models in machine learning fields. Using the SVM model for stock price prediction. SVM is one of the machine learning algorithms which works on classification algorithms. It is used to get a new text as an output. Applying Multiple Linear Regression with Interactions to

predict the trend in stock prices (Osman Hegazy et al. 2013 [20]; V Kranthi Sai Reddy, 2018 [8]; a Banerjee et al. 2020 [21]; Lufuno Ronald Marwala [13]). Random Walk Hypothesis which is proposed by Horne, j. C et al 1997 [27] which is used to predict stockprices, Horne j.c [27] said that the stock values are changes random and the past price values are not dependent on current values. EMH is different from the Random walk hypothesis but the EMH works mainly on Short term patterns for predicting stock prices.

Manh Ha Duong Boris's Siliverstovs, 2006 [11] search the abstraction between equity prices and combined finances in Key Eu nations like UK and Germany. Acceleration in Eu nations investments is apt to results successful even Stronger correlation between the different Eu nations and equity prices. This operation may also lead to a merge in financial development between EU nations, if advancements in stock markets affect real financial instruments, such as investing and Consuming. Fahad Almudhaf et al, 2012 [22], tests the weak-form market efficiency of CIVETS over the period 2002–2012. The random walk hypothesis process is used in CIVETS. In an efficient stock market, the equity values must follow a random walk hypothesis, when it comes to the future price, the values are changing randomly and unpredictable. Everyday returns for rising and improved markets have been tested for random walks.

LSTM algorithm consists of a Recurrent Neural network to encode data. The algorithm inputs are economic news headings infusion From Bloomberg and Reuters. Long Short- term Memory with embedded layer and the LSTM with the automatic encoder in the stock market for predicting stock values. The Xiongwen Pang et al [4]. Used an automatic encoder and embedded layer to vectorizing the values by using LSTM layers. Correlation coefficients in stocks are selected randomly and predicted using ARIMA and the neural network approach. In this RNN and LSTM algorithms are implemented. M. Nabipour et al [17]. Used different machine learning and deep learning algorithms for predicting stock values such as random forest, decision tree and neural networks. LSTM gives the most accurate results and it has the best ability to fit. LSTM gives the best results while predicting stock prices with the least error rate (Hyeong Kya Choi, 2018 [16]; Huicheng Liu, 2018 [15]; M. Nabipour et al, 2020 [17]; Xiongwen Pang et al, 2020 [4]).

Recently, Pranab Bhat, 2020 used convolution neural networks for predicting stock values, in this model learning is finished by computing the mean square blunder for each consequent perception and a model is picked that has the least mistake and high prescient power. In this paper, they are utilizing CNN for anticipating stocks and incentives for the following day. Mohammad Mekayel Anik et al, 2020 [23], implemented a linear regression algorithm for future stock price prediction. In this they achieved their goals in predicting accuracy of the model is very good and it might be used for predicting stock values. Xiao Ding et al. 2020 [14] used an easy and effective interface to add common sense knowledge to the process while learning of events.

The LMS filter is a type of adaptive filter which is used for solving linear problems. The idea of the filter is to find the filter coefficients and to minimize a system by reducing the least mean square of the error value (Asep Juarna, 2017 [24]; Eleftherios Giovanis, 2018 [25]). They used a hybrid model for predicting the stock values by using deep learning and ML methodologies and they built a model using deep regression based on CNN. Here they used CNN for parameters, thereby increase the no of loops will stabilize the validation loss. They also tested using DL and a hybrid ML algorithm for stock price prediction. Vivek Rajput and Sarika Bobde [26] used sentiment analysis from online posts or multimedia and data mining is used. In sentiment analysis, they are trying to get emotion either positive or negative based on the textual information available on social networks. sentiment analysis for predicting the stock market to get more accurate and efficient results.

### 3. DATA COLLECTION

For the experimental study, we downloaded live datasets namely google, nifty, reliance, etc. from the Yahoo Finance website (<https://finance.yahoo.com/>).

**Table 3.1 Google**

Attribute Name	Min	Max
Open	87.74	1005.49
Low	86.37	996.62
High	89.29	1008.61
Close	87.58	1004.28

**Table 3.2 Nifty50**

Attribute Name	Min	Max
Open	87.74	1005.49
Low	86.37	996.62
High	89.29	1008.61
Close	87.58	1004.28

**Table 3.3 Reliance**

Attribute Name	Min	Max
Open	205.5	3298
Low	197.15	3141.3
High	219.5	3298
Close	203.2	3220.85

### Sample Input

**Table 3.1 Sample Input**

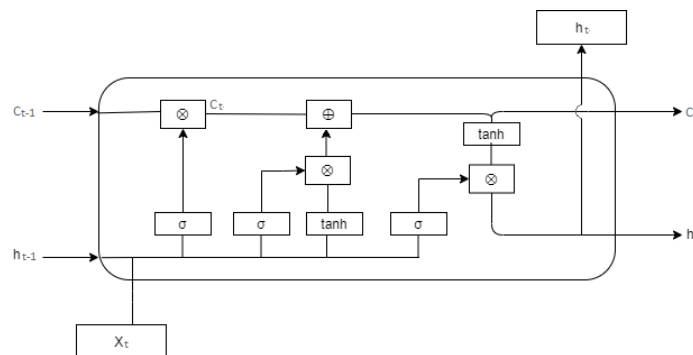
Date	Trade Open	Trade Low	Trade High	Trade Close
11-Jun-2021	2,524.92	2,498.29	2,526.99	2,513.93
10-Jun-2021	2,494.01	2,494.00	2,523.26	2,521.60
09-Jun-2021	2,499.50	2,487.33	2,505.00	2,491.40
08-Jun-2021	2,479.90	2,468.24	2,494.50	2,482.85
07-Jun-2021	2,451.32	2,441.07	2,468.00	2,466.09
04-Jun-2021	2,422.52	2,417.77	2,453.86	2,451.76
03-Jun-2021	2,395.02	2,382.83	2,409.75	2,404.61
02-Jun-2021	2,435.31	2,404.20	2,442.00	2,421.28

## 4. METHODOLOGIES

### 4.1 LSTM Algorithm

LSTM uses the RNN approach which has the ability to memorize. Each LSTM cell has three gates i.e. input, forget and output gates. While the data that enters the LSTM's network, the data that is required is kept and the unnecessary data will be forgotten by the forget gate.

LSTM can be used in many applications such as for weather forecasting, NLP, speech recognition, handwriting recognition, time-series prediction, etc.



### Fig 4.1.1: LSTM Architecture

As shown in Fig. 4.1.1, the inputs to the current cell state ( $C_t$ ) is the previous hidden state ( $h_{t-1}$ ), previous cell state ( $C_{t-1}$ ) and present input ( $X_t$ ). The cell consists of three gates i.e. forget gate, input gate and output gate.

#### Forget Gate:

A forget gate will remove unnecessary data from the cell state.

- The information that is less important or not required for the LSTM to understand things is removed by performing multiplication of hidden state by a sigmoid function.
- This step is necessary to optimize the performance of the model.
- It takes two inputs i.e.,  $h_{(t-1)}$  and  $x_t$ , where  $h_{(t-1)}$  is the previous cell hidden state output and  $x_t$  is the current cell input.

$$F_t = \sigma(W_{fx} * X_t + W_{fh} * h_{t-1} + b_f)$$

#### Input Gate:

1. This cell is responsible for regulating the data that is added to the cell from the input. Forget gate is used to filter some input.
2. A vector is created by adding all the possible values from the previous cell hiddenstate  $h_{(t-1)}$  and current cell input  $X_t$  by using the tanh function. The output of the tanh function in the ranges of  $[-1, 1]$ .
3. Finally, the outputs of sigmoid and tanh functions are multiplied and the output is added to the cell state.

$$I_t = \sigma(W_{ix} * X_t + W_{ih} * h_{t-1} + b_i) + \tanh(W_{cx} * X_t + W_{ch} * h_{t-1} + b_i)$$

#### Output Gate:

- Tanh function is applied to the cell state to create a vector with all possible values.
- Sigmoid function is applied to previous cell hidden state  $h_{(t-1)}$  and current cell input  $x_t$  to filter necessary data from the previous cell.
- Now, the outputs of sigmoid and tanh functions are multiplied and this output is sent as a hidden state of the next cell.

$$O_t = \sigma(W_{ox} * X_t + W_{oh} * h_{t-1} + W_{oc} * C_{t-1} + b_o)$$

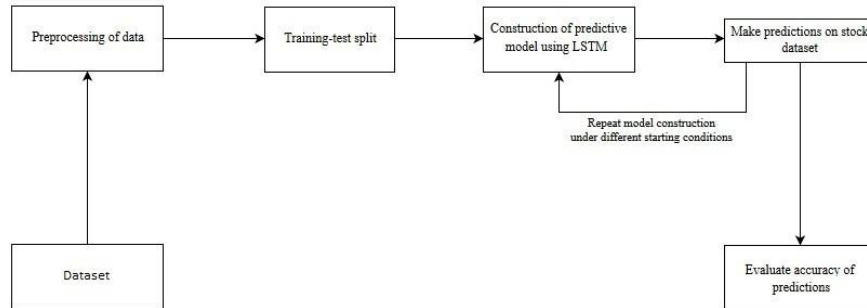
Intermediate cell state ( $C_t$ ) is obtained by the multiplication of Forget gate ( $F_t$ ) with previous cell state ( $C_{t-1}$ ). Then this intermediate state is added to the output of the input gate.

$$C_t = F_t * C_{t-1} + I_t$$

Current hidden/output state is obtained by multiplying output gate and tanh of cell state.

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

## 4.2 SYSTEM ARCHITECTURE



**Fig 4.2.1: Overall Architecture**

**Data Selection:** The first step is to select data for an organization and split the data into training and testing. we have used 75% for training and 25% for testing purposes.

**Pre-processing of data:** In pre-processing, we are selecting attributes required for the algorithm and the remaining attributes are neglected. The selected attributes are Trade Open, Trade High, Trade Low, Trade Close, Trade Volume. In pre-processing, we are using normalization to get values in a particular range.

**Prediction using LSTM:** In this system, we are using the LSTM algorithm for predicting stock values. Initially, the training data is passed through the system and train the model. Then in the testing phase, the predicted values are compared with the actual values.

**Evaluation:** In the evaluation phase we are calculating the Accuracy, Mean Square Error(MSE) and Root Mean Square Error (RMSE) values for comparison.

## 5. EXPERIMENTAL RESULTS

### 5.1 Google



**Fig 5.1.1 Google Graph**

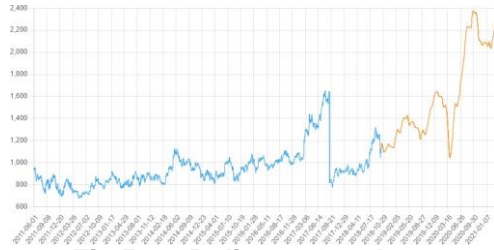
**Table 5.1.2 Google Epochs**

epochs	Accuracy	MSE	RMSE
10	93.00717	207.6578	14.41034
20	94.01166	156.3873	12.50549
30	95.64188	105.3248	10.26279
40	95.59026	99.17409	9.958619
50	96.99466	62.24641	7.88964

In the results, as we have shown in Fig 5.1.1, the graph shows Trade Close value for the google dataset. In this graph blue line indicates the training data and the yellow color

shown is the predicted values from the test data. Table 5.1.2 shows the accuracy, MSE and RMSE values for no of iterations (epochs).

## 5.2 Reliance



**Fig 5.2.1 Reliance Graph**

**Table 5.2.2 Reliance Epochs**

epochs	Accuracy	MSE	RMSE
10	96.25328	4839.5690	69.56701
20	97.63884	2653.1278	51.50852
30	98.19937	1650.3337	40.62430
40	98.13571	1616.9295	40.21106
50	98.37254	1361.8098	36.90270

Above graph 5.2.1 shows Trade Close value for the Reliance dataset and table 5.2.2 shows the MSE, RMSE and accuracy values for the Reliance dataset.

## CONCLUSION

we are predicting the closing stock price of any given organization, we have developed an application for predicting close stock price using LSTM algorithm. We have used datasets belonging to Google, Nifty50, TCS, Infosys and Reliance Stocks and achieved above 93% accuracy for these datasets. In the future, we can extend this application for predicting cryptocurrency trading and also, we can add sentiment analysis for better predictions.

## REFERENCES

- [1] Stock Price Prediction Using LSTM on Indian Share Market by Achyut Ghosh, Soumik Bose<sup>1</sup>, GiridharMaji, Narayan C. Debnath, Soumya Sen
- [2] S. Selvin, R. Vinayakumar, E. A. Gopalkrishnan, V. K. Menon and K. P. Soman - Stock price prediction using LSTM, RNN and CNN-sliding window model - 2017.
- [3] Murtaza Roondiwala, Harshal Patel, Shraddha Varma, "Predicting Stock Prices Using LSTM" in Undergraduate Engineering Students, Department of Information Technology, Mumbai University, 2015.
- [4] Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin, "An innovative neural network approach for stock market prediction", 2018
- [5] Ishita Parmar, Navanshu Agarwal, Sheersh Saxena, Ridam Arora, Shikhin Gupta, Himanshu Dhiman, Lokesh Chouhan Department of Computer Science and Engineering National Institute of Technology, Hamirpur – 177005, INDIA - Stock Market Prediction Using Machine Learning.
- [6] Pranav Bhat Electronics and Telecommunication Department, Maharashtra Institute of Technology, Pune. Savitribai Phule Pune University - A Machine Learning Model for Stock Market Prediction.
- [7] Anurag Sinha Department of computer science, Student, Amity University Jharkhand Ranchi, Jharkhand (India), 834001 - Stock Market Prediction Using Machine Learning.
- [8] V Kranthi Sai Reddy Student, ECM, Sreenidhi Institute of Science and Technology, Hyderabad, India - Stock Market Prediction Using Machine Learning.



- [9] Asset Durmagambetov currently works at the mathematics, CNTFI- 'The Riemann Hypothesis-Millennium Prize Problems' - stock market predictions.
- [10] Mariam Moukalled Wassim El-Hajj Mohamad Jaber Computer Science Department American University of Beirut - Automated Stock Price Prediction Using Machine Learning.
- [11] Manh Ha Duong Boriss Siliverstovs June 2006 - The Stock Market and Investment.
- [12] Dharmaraja Selvamuthu, Vineet Kumar and Abhishek Mishra Department of Mathematics, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India - Indian stock market prediction using artificial neural networks on tick data.
- [13] Lufuno Ronald Marwala A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering - Forecasting the Stock Market Index Using Artificial Intelligence Techniques.
- [14] Xiao Ding, Kuo Liao, Ting Liu, Zhongyang Li, Junwen Duan Research Centre for Social Computing and Information Retrieval Harbin Institute of Technology, China - Event Representation Learning Enhanced with External Common-sense Knowledge.
- [15] Huicheng Liu Department of Electrical and Computer Engineering Queen's University, Canada - Leveraging Financial News for Stock Trend Prediction with Attention-Based Recurrent Neural Network.
- [16] Hyeong Kyu Choi, B.A Student Dept. of Business Administration Korea University Seoul, Korea = Stock Price Correlation Coefficient Prediction with ARIMA-LSTM Hybrid Model.
- [17] M. Nabipour Faculty of Mechanical Engineering, Tarbiat Modares University, 14115-143 Tehran, Iran; Mojtaba.nabipour@modares.ac.ir - Deep Learning for Stock Market prediction.
- [18] Lavanya Ra SRM Institute of Science and Technology | SRM · Department of Computer Science - Stock Market Prediction.
- [19] M. Mekayel Anik · M. Shamsul Arefin (B) Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong, Bangladesh - An Intelligent Technique for Stock Market Prediction.
- [20] Osman Hegazy, Omar S. Soliman and Mustafa Abdul Salam Faculty of Computers and Informatics, Cairo University, Egypt - A Machine Learning Model for Stock Market Prediction
- [21] Sharanya Banerjee, Neha Dabeeru, R. Lavanya SRM Institute of Science and Technology, Chennai in Computer Science and Engineering - Stock Market Prediction
- [22] Fahad Almodhafi, Yaser A. Alkulaib from Kuwait University - Are Civets Stock Markets Predictable?
- [23] Mohammad Mekayel Anik, Mohammad Shamsul Arefin and M. Ali Akber Dewan, Department of Computer Science and Engineering - An Intelligent Technique for Stock Market Prediction
- [24] Asep Juarna, Departemen of Informatics, Gunadarma University, Jakarta-Indonesia - ONE YEAR STOCK PRICE PREDICTION AND ITS VALIDITY USING LEAST SQUARE METHOD IN MATLAB
- [25] Giovanis, Eleftherios - Applications of Least Mean Square (LMS) Algorithm Regression in Time-Series Analysis
- [26] Vivek Rajput and Sarika Bobde - Stock market prediction using hybrid approach
- [27] James C. Van Horne and George G. C. Parker – The Random-walk Theory: An Empirical Test