

STOCK PRICE PREDICTION USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

RAVIRAJ BHOSALE (20BAI10379)

ARYA DHRANGDHRIA (20BAI10016)

RAJ DAMA (20BAI10350)

ANSHUL KUMAR (20BAI10006)

*in partial fulfillment for the award of the degree
of*

BACHELOR OF TECHNOLOGY

In

**Computer Science And Engineering with Specialization in Artificial Intelligence
and Machine Learning**



VIT[®]
BHOPAL
www.vitbhopal.ac.in

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

**KOTRIKALAN, SEHORE
MADHYA PRADESH - 466114**

APRIL 2022

**VIT BHOPAL UNIVERSITY, KOTRIKALAN, SEHORE
MADHYA PRADESH – 466114**

BONAFIDE CERTIFICATE

Certified that this project report titled “**STOCK PRICE PREDICTION USING MACHINE LEARNING**” is the bonafide work of “**RAVIRAJ BHOSALE (20BAI10379), ARYA DHRANGDHRIA (20BAI10016), ANSHUL KUMAR(20BAI10006), RAJ DAMA (20BAI10350)**” who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR

Dr. S Sountharajan (Associate Professor)
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

PROJECT GUIDE

Dr. Rudra Kalyan Nayak (Assistant Professor)
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

The Project Exhibition II Examination is held on 29/04/2022

ACKNOWLEDGEMENT

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr Sountharajan, Head of the Department, School of Aeronautical Science for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide Dr. Rudra Kalyan Nayak for continuously guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computer Science and Engineering, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

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 |

LIST OF FIGURES AND GRAPHS

| FIGURE NO. | TITLE | PAGE NO. |
|-----------------------|--|-----------------|
| 1 | LMS Inputs and Outputs | 18 |
| 2 | LMS updating weights | 18 |
| 3 | Actual Vs Predicted Graph | 20 |
| 4 | LSTM Architecture | 21 |
| 5 | Preprocessing of Data | 24 |
| 6 | Overall Architecture | 25 |
| 7 | Training and prediction | 26 |
| 8 | Using LMS, LSTM and LSTM with LMS in the system | 29 |
| 9 | Execution based on model selection | 29 |
| 10 | Execution based on algorithm selection | 30 |
| 11 | Flow of execution | 31 |
| 12 | Components present in the system | 32 |

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

TABLE OF CONTENTS

| CHAPTER NO. | TITLE | PAGE NO. |
|-------------|---|--------------------------------|
| | List of Abbreviations | 4 |
| | List of Figures and Graphs | 5 |
| | Abstract | 6 |
| 1 | CHAPTER-1 PROJECT DESCRIPTION AND OUTLINE 1.1 Introduction 1.2 Motivation for the work 1.3 Problem Statement | 10 |
| 2 | CHAPTER-2 LITERATURE SURVEY 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 networks on tick data 2.2.4 The Stock Market and Investment 2.2.5 Automated Stock Price Prediction Using Machine Learning | 12 |
| | | |

| | | |
|---|---|----|
| | 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 | |
| 3 | CHAPTER-3 METHODOLOGY 3.1 Proposed Systems 3.1.1 System Architecture | 17 |
| 4 | CHAPTER-4 DESIGN METHODOLOGY 4.1 Structure Chart 4.2 UML Diagrams 4.2.1 Use Case Diagram 4.2.2 Sequence Diagram 4.2.3 Activity Diagram 4.2.5 Flow Chart 4.2.6 Component Diagram | 26 |
| 5 | CHAPTER-5 EXPERIMENT ANALYSIS 5.1 System configuration 5.2 Sample Code 5.3 Working Layout of Forms 5.4 Test and validation 5.5 Performance Analysis(Graphs/Charts) | 33 |

| | | |
|---|--|----|
| | 5.6 Summary | |
| 7 | CHAPTER-6 CONCLUSIONS AND RECOMMENDATION 7.1 Conclusion 7.2 Future Work | 46 |
| | References | 47 |

CHAPTER 1

1.1 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 ATSSs, 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.

1.2 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.

1.3 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 out performed 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 market¹. Markets are mostly a non-parametric, non-linear, noisy and deterministic chaotic system (Ahangar 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, 6

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.

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 95%.

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.

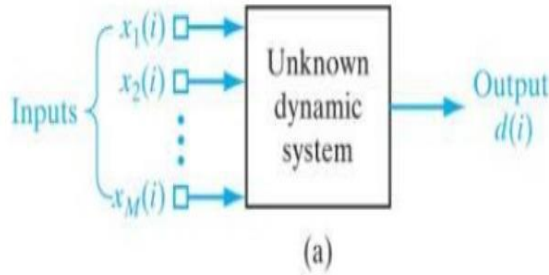


Fig. 1: LMS Inputs and Outputs

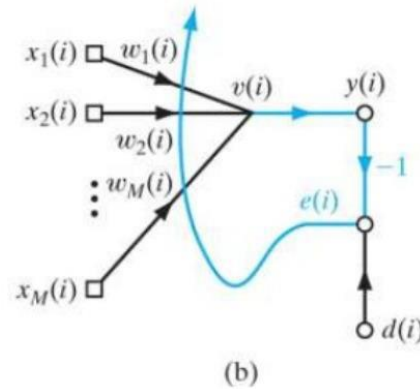


Fig 2: LMS updating weights

Algorithm 1: LMS

Input:

x : input vector
 d : desired vector
 μ : 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**.

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

μ : 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

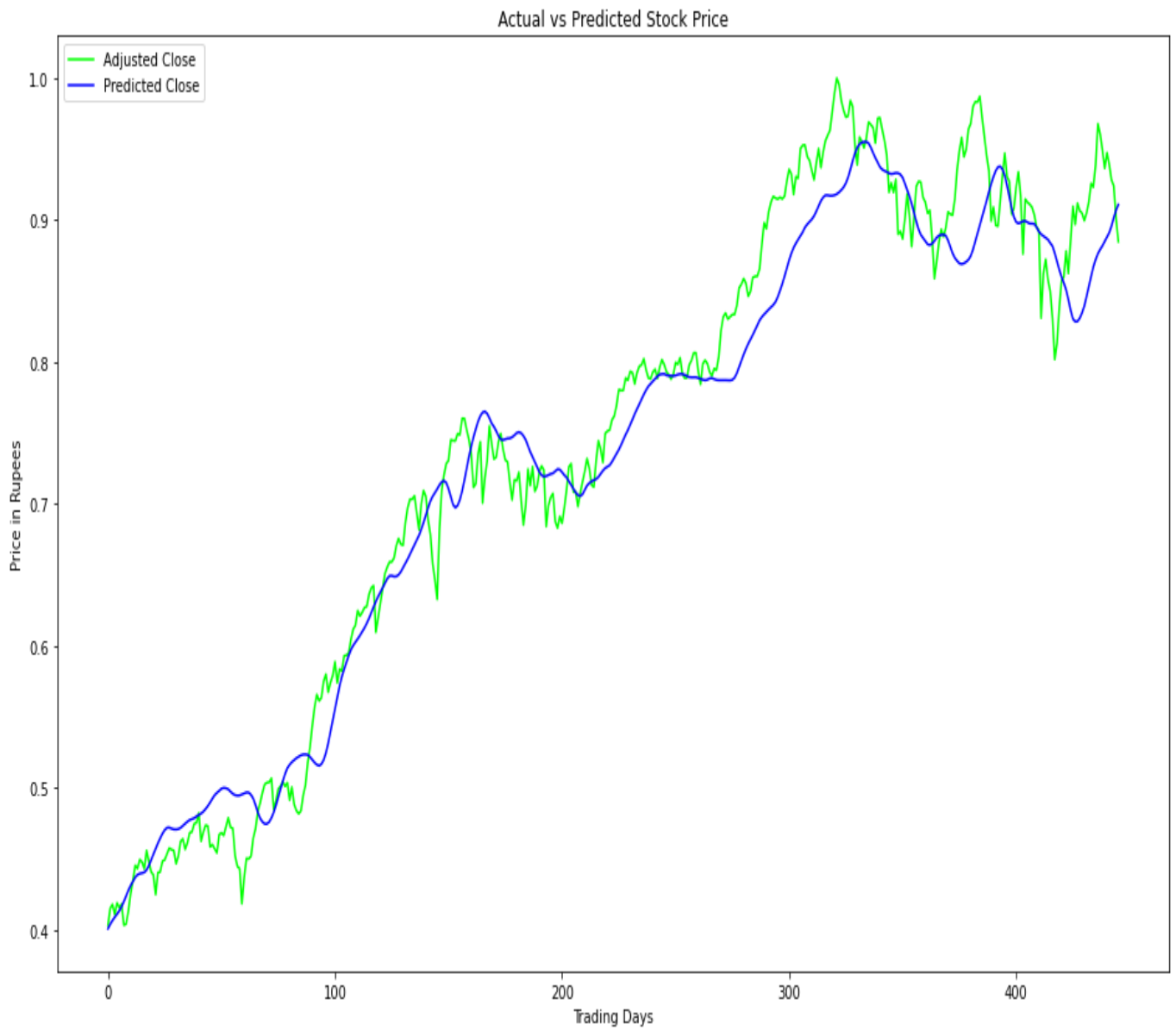


Fig. 5: LSTM Output

One example of stock market prediction result

LSTM Architecture

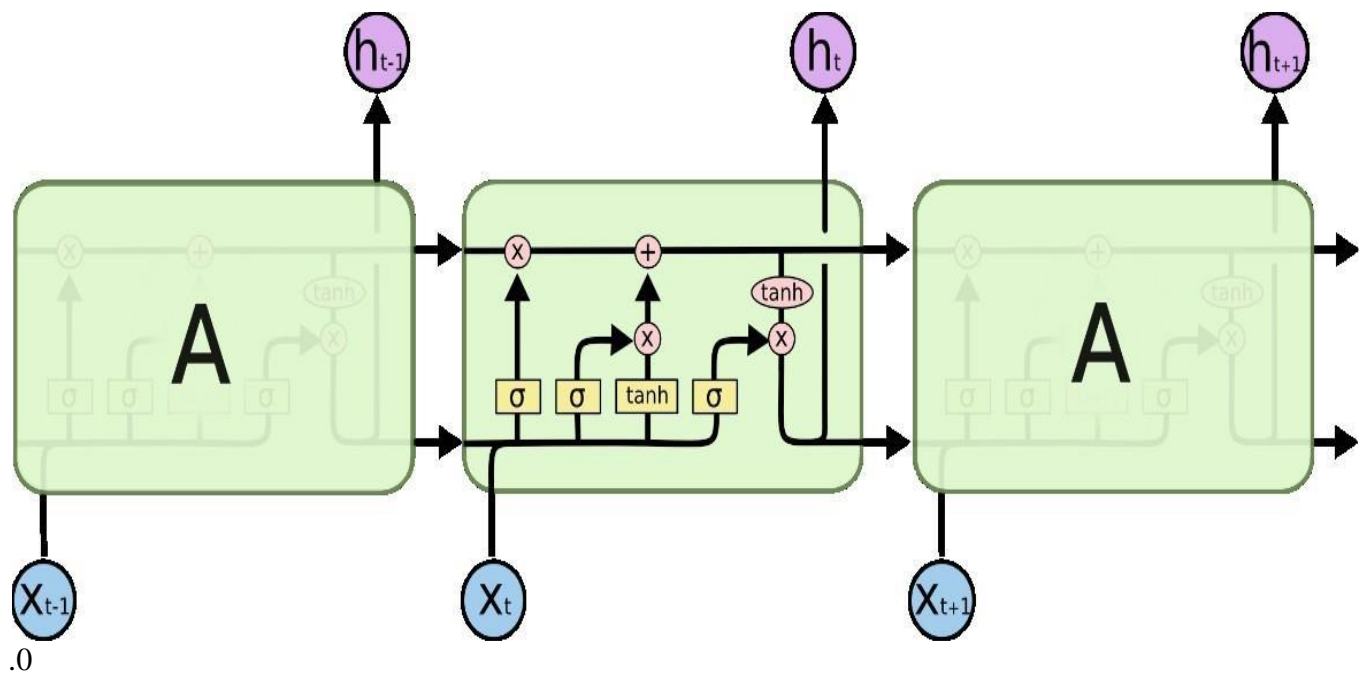


Fig. 4: LSTM Architecture

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:

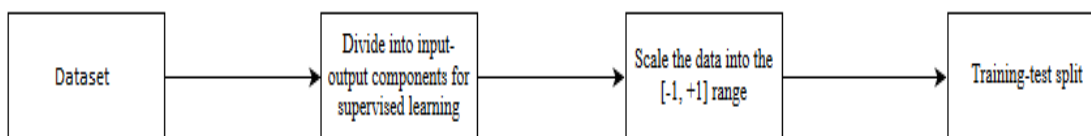
Product properties

- 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.

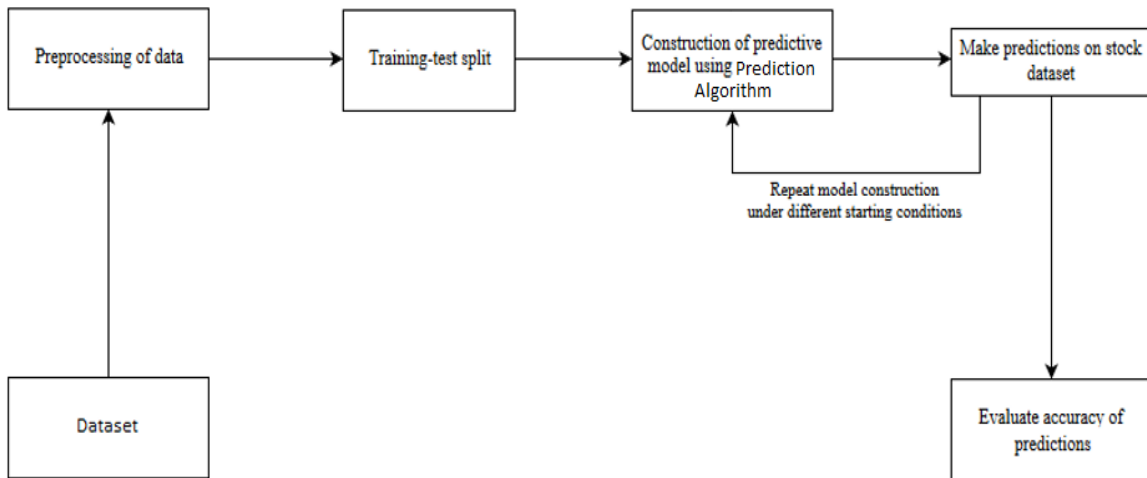
Performance: It is a quality attribute of the stock prediction software that describes the responsiveness to various user interactions with it.

3.1.1 SYSTEM ARCHITECTURE

1) Preprocessing of data



2) 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.

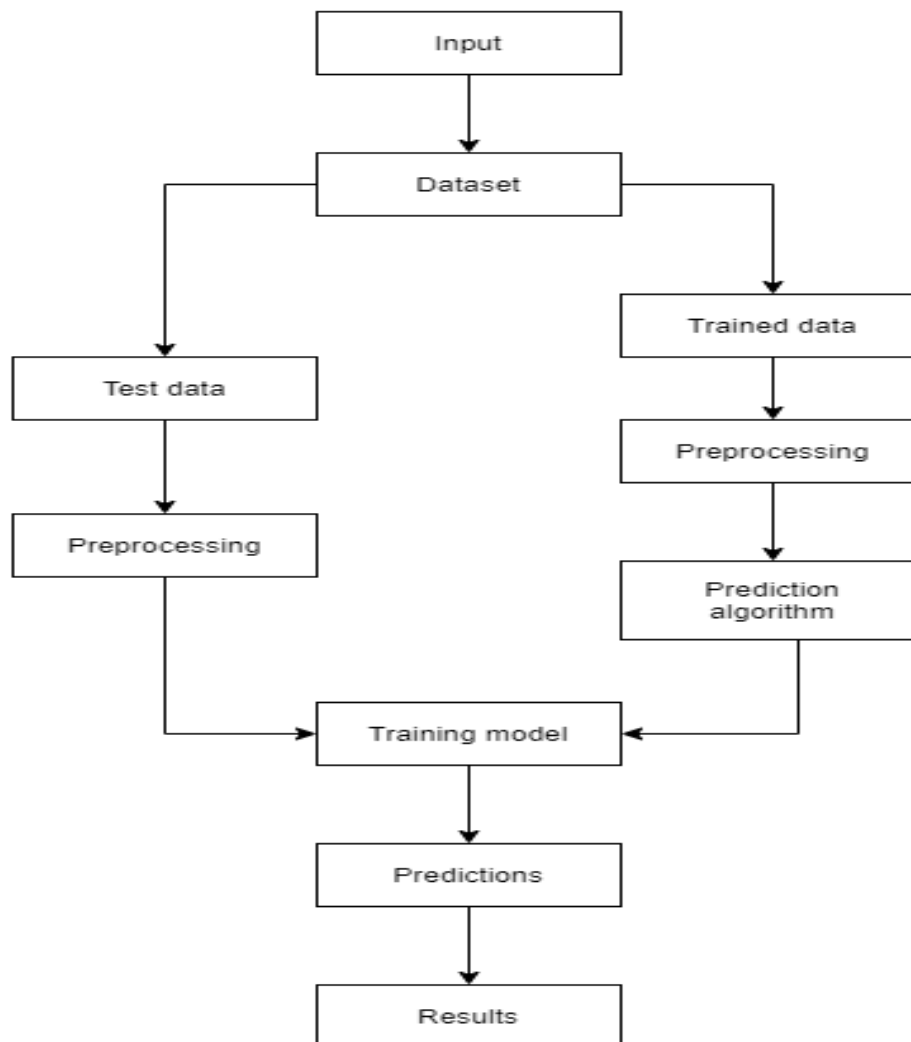


Fig. 7: Training and prediction

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.

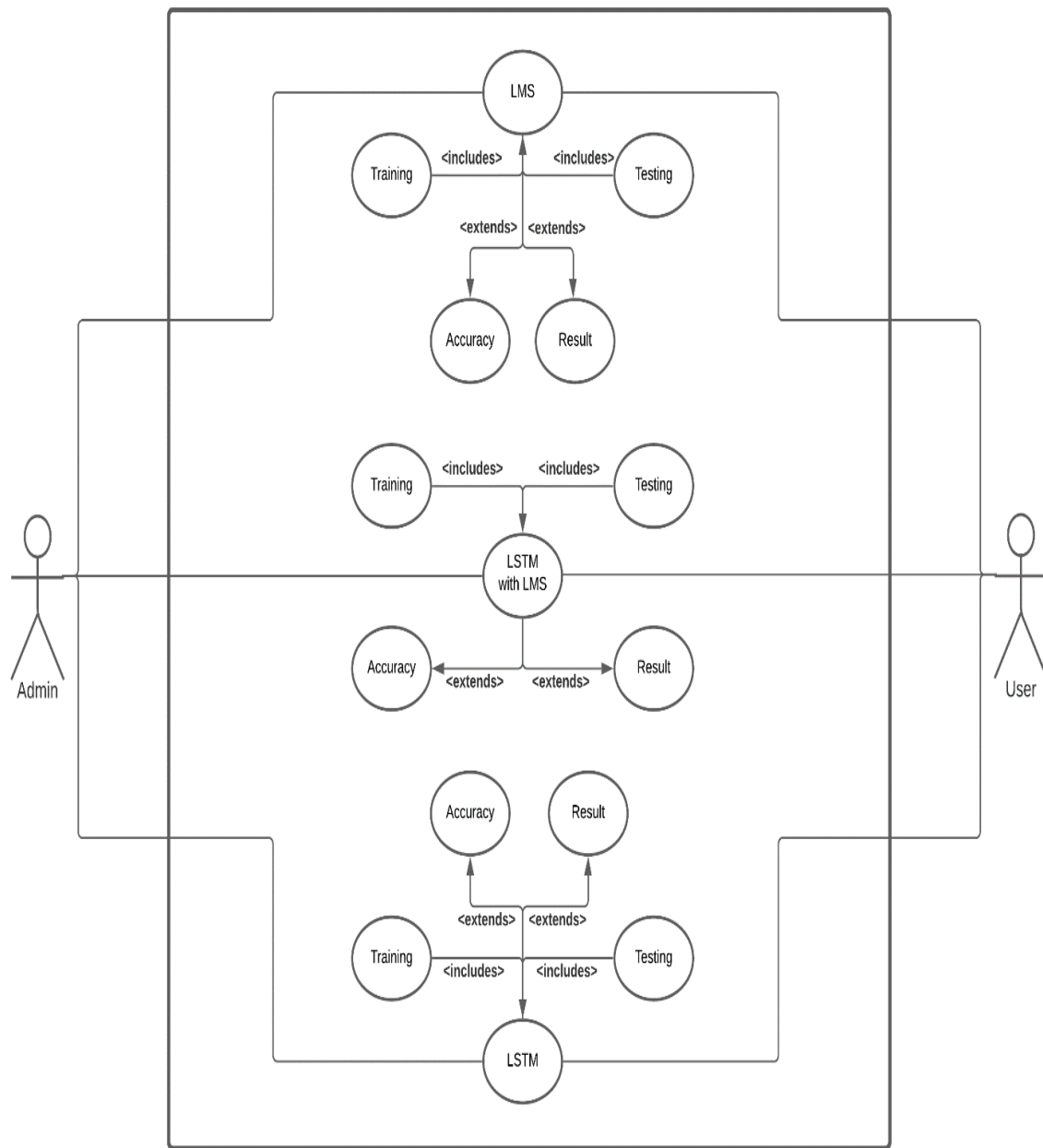


Fig. 8: Using LMS, LSTM and LSTM with LMS in the system

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.

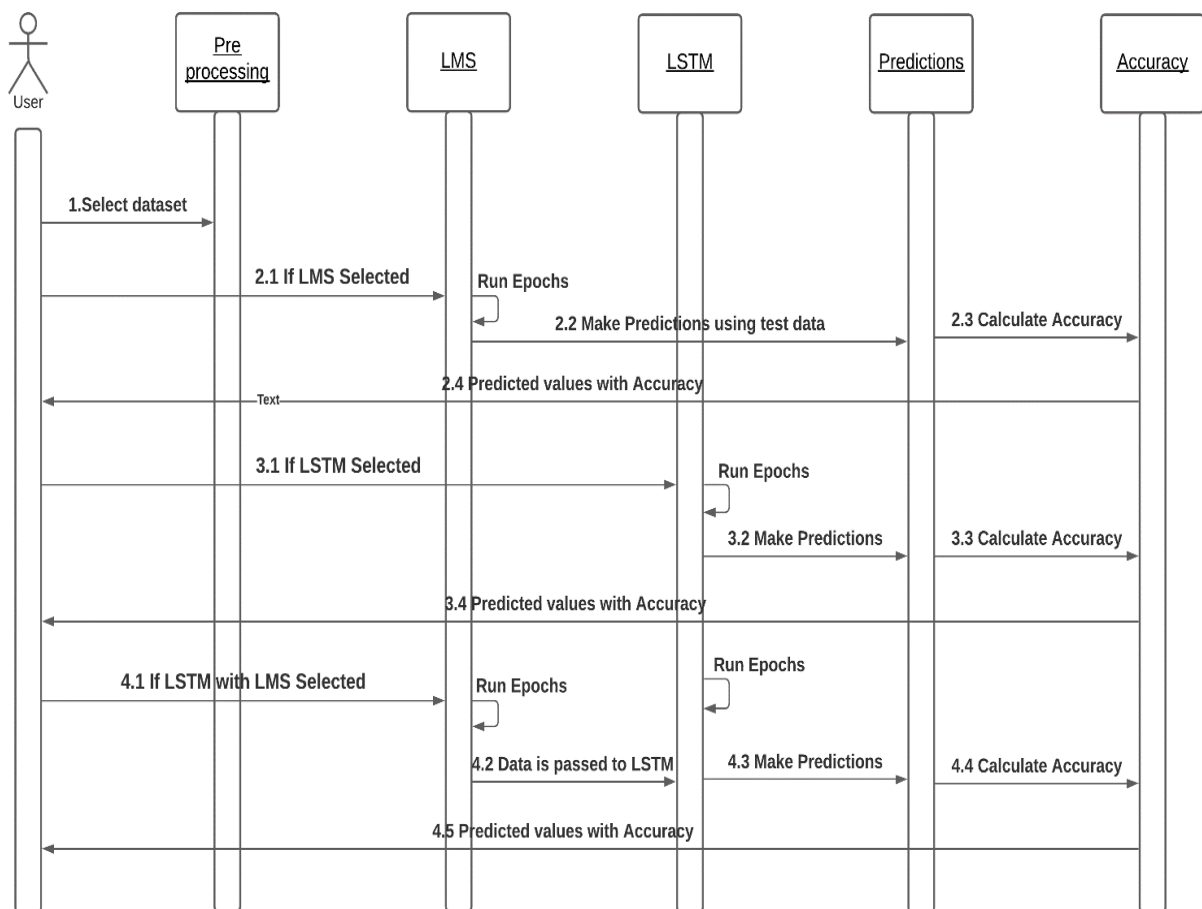


Fig. 9: Execution based on model selection

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.

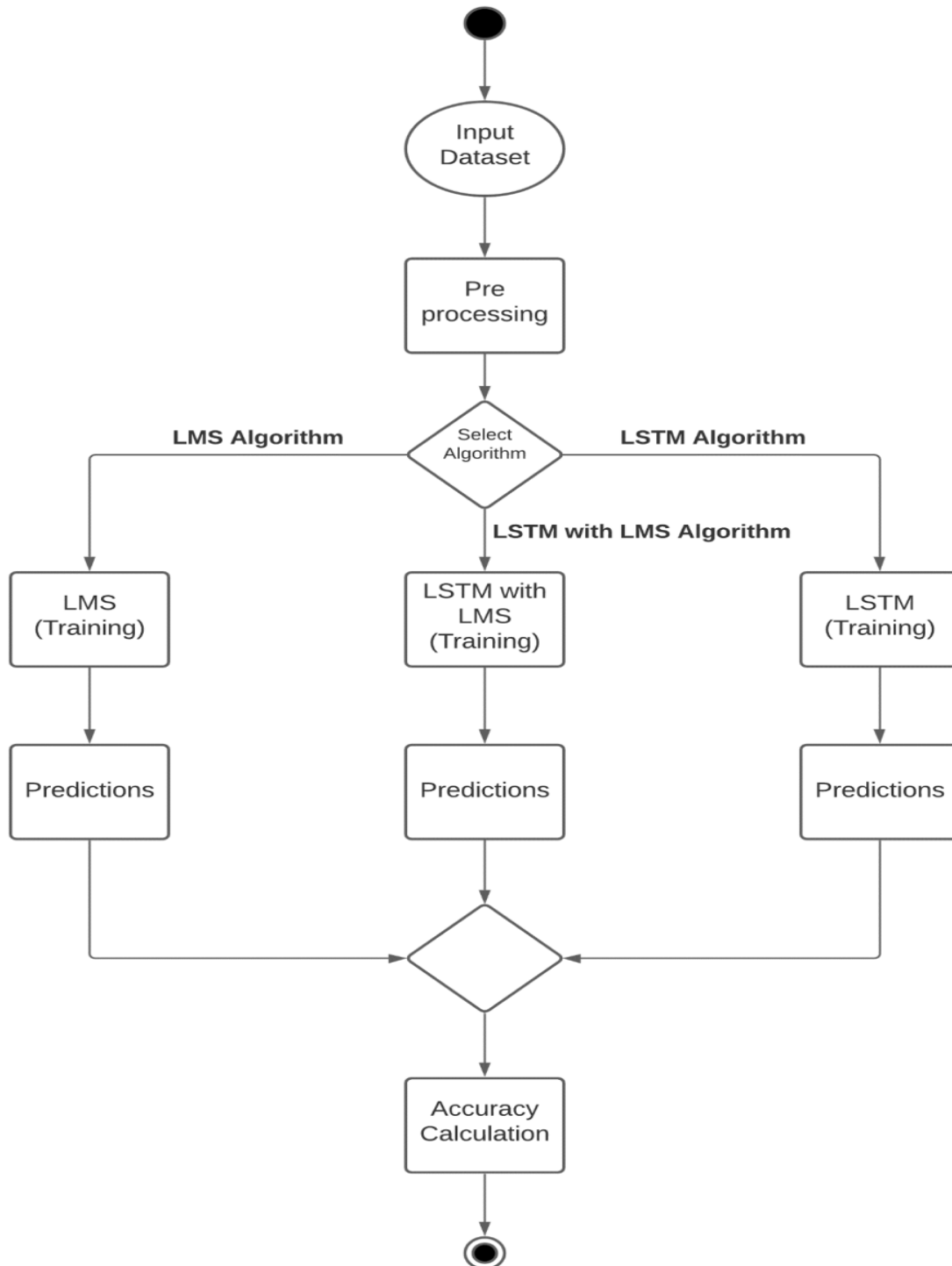


Fig. 10: Execution based on algorithm selection

4.2.5 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.

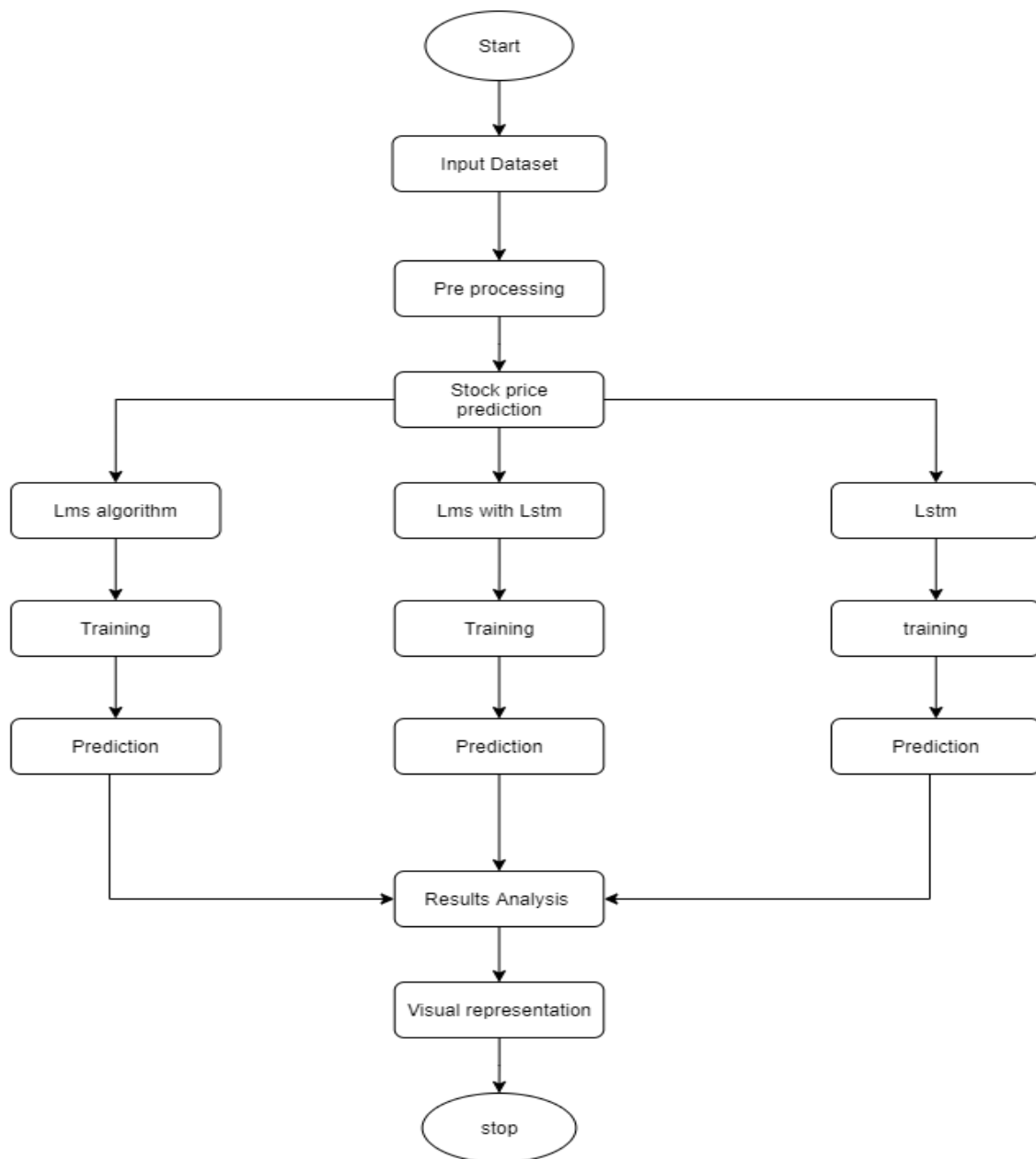


Fig. 11: Flow of execution

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.

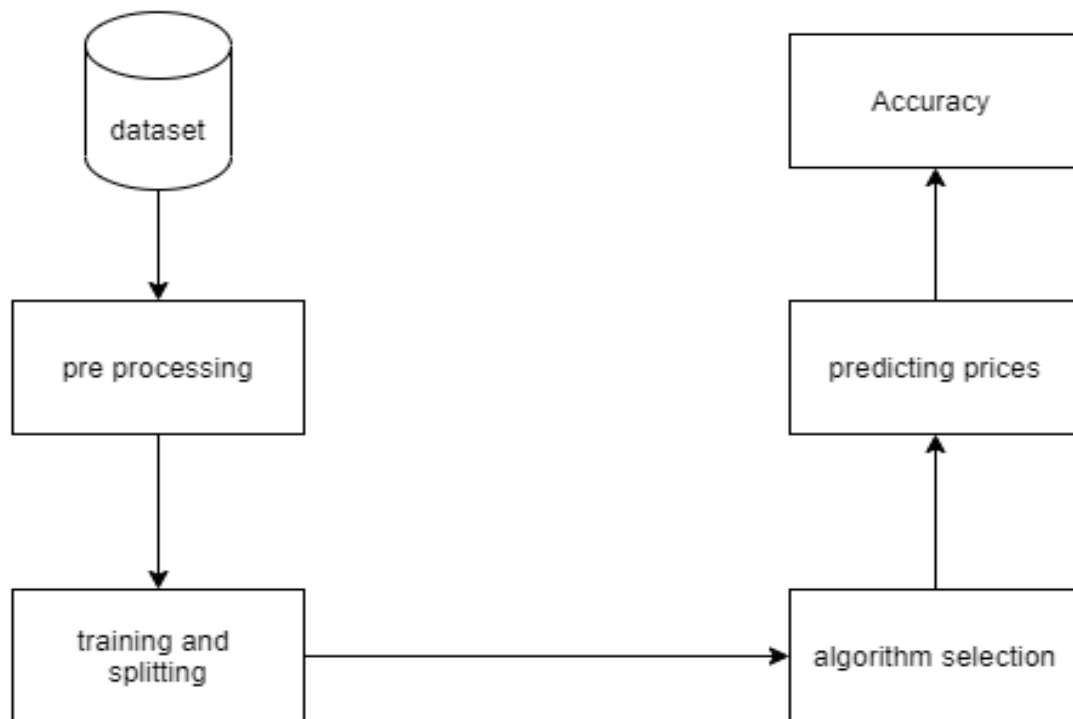


Fig. 12: Components present in the system

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

5.2 Sample code

▼ Importing Libraries

```
[ ] import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
```

```
[ ] df = pd.read_csv("/content/drive/MyDrive/CSV/^NSEI (2).csv")
```

```
[ ] df.head()
```

| | Date | Open | High | Low | Close | Adj Close | Volume |
|---|------------|-------------|-------------|-------------|-------------|-------------|--------|
| 0 | 2007-09-17 | 4518.450195 | 4549.049805 | 4482.850098 | 4494.649902 | 4494.649902 | 0.0 |
| 1 | 2007-09-18 | 4494.100098 | 4551.799805 | 4481.549805 | 4546.200195 | 4546.200195 | 0.0 |
| 2 | 2007-09-19 | 4550.250000 | 4739.000000 | 4550.250000 | 4732.350098 | 4732.350098 | 0.0 |
| 3 | 2007-09-20 | 4734.850098 | 4760.850098 | 4721.149902 | 4747.549805 | 4747.549805 | 0.0 |
| 4 | 2007-09-21 | 4752.950195 | 4855.700195 | 4733.700195 | 4837.549805 | 4837.549805 | 0.0 |

df

▼ Data Preprocessing

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3601 entries, 0 to 3600
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Date        3601 non-null   object
1   Open        3571 non-null   float64
2   High        3571 non-null   float64
3   Low         3571 non-null   float64
4   Close       3571 non-null   float64
5   Adj Close   3571 non-null   float64
6   Volume      3571 non-null   float64
dtypes: float64(6), object(1)
memory usage: 197.1+ KB
```

```
[ ] df.describe()
```

| | Open | High | Low | Close | Adj Close | Volume |
|-------|-------------|-------------|-------------|-------------|-------------|--------------|
| count | 3571.000000 | 3571.000000 | 3571.000000 | 3571.000000 | 3571.000000 | 3.571000e+03 |
| mean | 8242.757085 | 8291.736266 | 8179.481555 | 8236.812672 | 8236.812672 | 1.908289e+05 |
| std | 3564.148151 | 3571.392797 | 3546.859430 | 3560.113978 | 3560.113978 | 2.205500e+05 |
| min | 2553.600098 | 2585.300049 | 2252.750000 | 2524.199951 | 2524.199951 | 0.000000e+00 |

```
df = df.drop(['Date', 'High', 'Low', 'Adj Close'], axis = 'columns')
```

```
[ ] df
```

| | Open | Close | Volume |
|------|--------------|--------------|----------|
| 0 | 4518.450195 | 4494.649902 | 0.0 |
| 1 | 4494.100098 | 4546.200195 | 0.0 |
| 2 | 4550.250000 | 4732.350098 | 0.0 |
| 3 | 4734.850098 | 4747.549805 | 0.0 |
| 4 | 4752.950195 | 4837.549805 | 0.0 |
| ... | ... | ... | ... |
| 3596 | 17740.900391 | 17674.949219 | 251700.0 |
| 3597 | 17584.849609 | 17530.300781 | 266000.0 |
| 3598 | 17599.900391 | 17475.650391 | 245100.0 |
| 3599 | 17183.449219 | 17173.650391 | 376100.0 |
| 3600 | 17258.949219 | 16958.650391 | 401400.0 |

3601 rows × 3 columns

```
[ ] df = df.drop(df.index[0:1321], axis = 0)
df.head()
```

| | Open | Close | Volume |
|------|-------------|-------------|----------|
| 1321 | 6085.750000 | 6082.299805 | 130900.0 |
| 1322 | 6080.149902 | 6048.500000 | 129000.0 |
| 1323 | 6052.850098 | 6054.299805 | 137000.0 |
| 1324 | 6046.200195 | 6019.350098 | 185200.0 |
| 1325 | 6024.500000 | 6074.649902 | 147600.0 |

```
[ ] df.describe()
```

| | Open | Close | Volume |
|-------|--------------|--------------|--------------|
| count | 2266.000000 | 2266.000000 | 2.266000e+03 |
| mean | 10125.504878 | 10117.103158 | 3.007281e+05 |
| std | 3147.340953 | 3144.290119 | 2.088148e+05 |
| min | 5233.450195 | 5285.000000 | 0.000000e+00 |
| 25% | 7967.412475 | 7961.074829 | 1.593250e+05 |
| 50% | 9654.949707 | 9655.524903 | 2.195000e+05 |
| 75% | 11513.887695 | 11498.650391 | 3.785000e+05 |
| max | 18602.349609 | 18477.050781 | 1.811000e+06 |

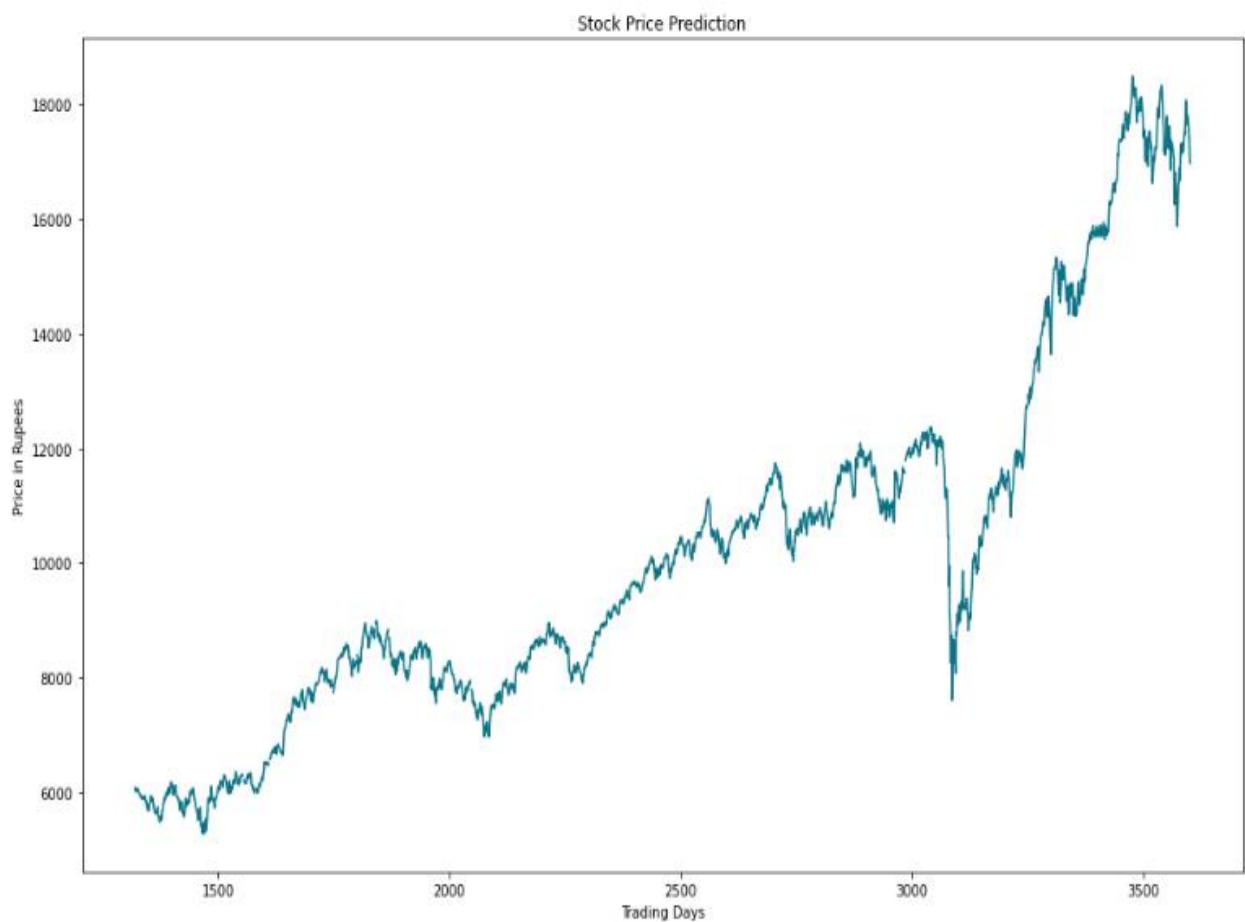
```
[ ] plt.rcParams['figure.figsize'] = (16, 10)

fig, ax = plt.subplots()
ax.plot(df['Close'], '#0A7388')

#ax.format_ydata = price
ax.set_title("Stock Price Prediction")

plt.ylabel("Price in Rupees")
plt.xlabel("Trading Days")

plt.show()
```



▼ Normalizing the Data

```
[ ] def normalize_data(data):  
  
    scaler = MinMaxScaler()  
    numerical = ['Open', 'Close', 'Volume']  
    data[numerical] = scaler.fit_transform(data[numerical])  
  
    return data  
  
df = normalize_data(df)  
  
df
```

▼ Basic LSTM Model

```
[ ] import math  
import pandas as pd  
import numpy as np  
from IPython.display import display  
  
from keras.layers.core import Dense, Activation, Dropout  
from keras.layers.recurrent import LSTM  
from keras.models import Sequential  
from keras.metrics import mean_squared_error  
from sklearn.model_selection import StratifiedKFold  
from sklearn.model_selection import train_test_split  
  
import time  
  
df
```

```
[ ] df.isnull().sum()
```

```
Open      14  
Close     14  
Volume    14  
dtype: int64
```

```
[ ] df = df.dropna(how='any',axis=0)
```

```
[ ] df.isnull().sum()
```

```
Open      0  
Close     0  
Volume    0  
dtype: int64
```

```
[ ] def train_test_split_lstm(stocks, prediction_time=1, test_data_size=450, unroll_length=50):  
    test_data_cut = test_data_size + unroll_length + 1  
  
    x_train = stocks[0:-prediction_time - test_data_cut].values  
    y_train = stocks[prediction_time:-test_data_cut]['Close'].values  
  
    x_test = stocks[0 - test_data_cut:-prediction_time].values  
    y_test = stocks[prediction_time - test_data_cut:]['Close'].values  
  
    return x_train, x_test, y_train, y_test
```

```
[ ] X_train, X_test, y_train, y_test = train_test_split_lstm(df, 5)
```

```
[ ] def unroll(data, sequence_length=24):  
    result = []  
    for index in range(len(data) - sequence_length):  
        result.append(data[index: index + sequence_length])  
    return np.asarray(result)
```

```
[ ] unroll_length = 50  
X_train = unroll(X_train, unroll_length)  
X_test = unroll(X_test, unroll_length)  
y_train = y_train[-X_train.shape[0]:]  
y_test = y_test[-X_test.shape[0]:]  
  
print("x_train", X_train.shape)  
print("y_train", y_train.shape)  
print("x_test", X_test.shape)  
print("y_test", y_test.shape)  
  
x_train (1710, 50, 3)  
y_train (1710,)  
x_test (446, 50, 3)  
y_test (446,)
```

▼ Build a basic Long-Short Term Memory model

```
[ ] def build_basic_model(input_dim, output_dim, return_sequences):
    model = Sequential()
    model.add(LSTM(
        input_shape=(None, input_dim),
        units=output_dim,
        return_sequences=return_sequences))

    model.add(LSTM(
        100,
        return_sequences=False))


    model.add(Dense(
        units=1))
    model.add(Activation('linear'))

    return model
```

```
[ ] model = build_basic_model(input_dim = X_train.shape[-1], output_dim = unroll_length, return_sequences=True)
```

```
[ ] start = time.time()
    model.compile(loss='mean_squared_error', optimizer='adam')
    print('compilation time : ', time.time() - start)
```


compilation time : 0.013763427734375

```
 def standardize(train, test):

    mean = np.mean(train, axis=0)
    std = np.std(train, axis=0)+0.000001

    X_train = (train - mean) / std
    X_test = (test - mean) / std
    return X_train, X_test

standardize(X_train, X_test)
```

```
 (array([[[-1.46385032, -1.46432493, -0.70401641],
          [-1.46828257, -1.48383625, -0.71610383],
          [-1.48428254, -1.48225533, -0.66030869],
          ...,
          [-1.78927629, -1.7705893 , -0.22397562],
          [-1.76503103, -1.76116147, -0.85480443],
          [-1.76515894, -1.74002637, -0.8069976 ]],

          [[-1.46683847, -1.48238946, -0.71705968],
          [-1.48284483, -1.48073746, -0.66121789],
          [-1.48782859, -1.50092397, -0.32981793],
          ...,
          [-1.76256617, -1.75879528, -0.85446491],
          [-1.76268633, -1.73755271, -0.80640094],
          [-1.7443889 , -1.78068925, -0.6142873 ]],

          [[-1.48140531, -1.47928971, -0.66214063],
          [-1.48639202, -1.49941082, -0.33053011],
          [-1.49940018, -1.47138518, -0.58762815],
          ...,
          [-1.76022361, -1.73520724, -0.80600908],
          [-1.74193488, -1.77817915, -0.61334982],
          [-1.79809445, -1.8337617 , -0.64855939]]],
```

```
[ ] model.fit(X_train, y_train, epochs=10, validation_split=0.05)
```

```
Epoch 1/10
51/51 [=====] - 13s 152ms/step - loss: 0.0064 - val_loss: 0.0074
Epoch 2/10
51/51 [=====] - 7s 139ms/step - loss: 3.6317e-04 - val_loss: 0.0072
Epoch 3/10
51/51 [=====] - 7s 140ms/step - loss: 3.4663e-04 - val_loss: 0.0079
Epoch 4/10
51/51 [=====] - 7s 140ms/step - loss: 3.4668e-04 - val_loss: 0.0070
Epoch 5/10
51/51 [=====] - 7s 140ms/step - loss: 3.4478e-04 - val_loss: 0.0075
Epoch 6/10
51/51 [=====] - 7s 140ms/step - loss: 3.3276e-04 - val_loss: 0.0074
Epoch 7/10
51/51 [=====] - 8s 165ms/step - loss: 3.4434e-04 - val_loss: 0.0061
Epoch 8/10
51/51 [=====] - 7s 141ms/step - loss: 3.2004e-04 - val_loss: 0.0055
Epoch 9/10
51/51 [=====] - 7s 141ms/step - loss: 3.2191e-04 - val_loss: 0.0056
Epoch 10/10
51/51 [=====] - 7s 142ms/step - loss: 3.2009e-04 - val_loss: 0.0077
<keras.callbacks.History at 0x7f72228e7e50>
```

Making Predictions

```
[ ] predictions = model.predict(X_test)
```

```
predictions
```

```
array([[0.4009685 ],
       [0.4040743 ],
       [0.40670893],
       [0.4090837 ]])
```

Plotting the Results



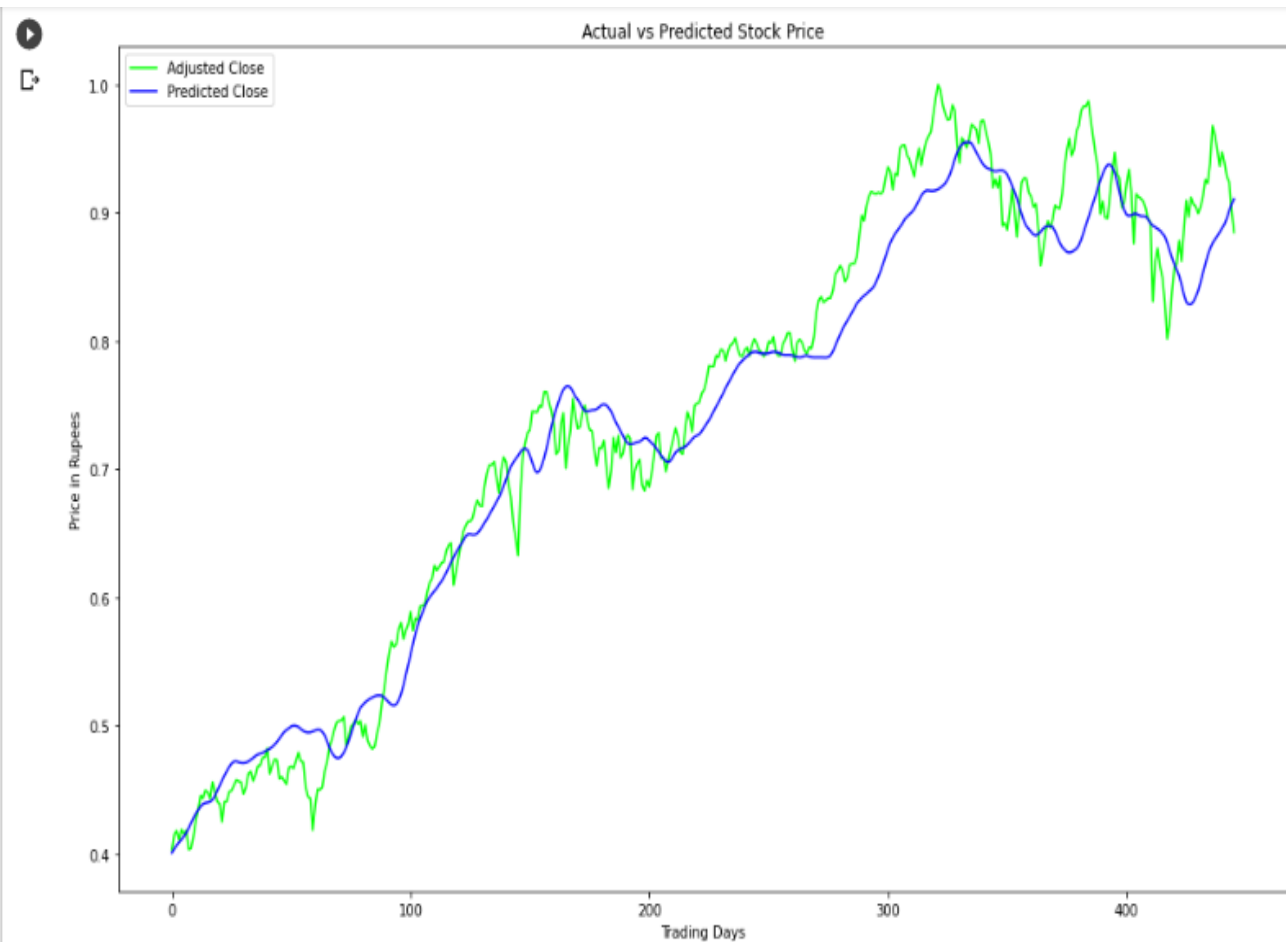
```
plt.rcParams['figure.figsize'] = (16, 10)
fig = plt.figure()
ax = fig.add_subplot(111)

plt.ylabel("Price in Rupees")
plt.xlabel("Trading Days")

plt.plot(y_test, '#00FF00', label='Adjusted Close')
plt.plot(predictions, '#0000FF', label='Predicted Close')

ax.set_title("Actual vs Predicted Stock Price")
ax.legend(loc='upper left')

plt.show()
```

Test Score

```
[ ] trainScore = model.evaluate(X_train, y_train, verbose=0)
print('Train Score: %.8f MSE (%.8f RMSE)' % (trainScore, math.sqrt(trainScore)))

testScore = model.evaluate(X_test, y_test, verbose=0)
print('Test Score: %.8f MSE (%.8f RMSE)' % (testScore, math.sqrt(testScore)))

Train Score: 0.00077880 MSE (0.02790701 RMSE)
Test Score: 0.00126558 MSE (0.03557493 RMSE)
```

```
[ ] from sklearn.metrics import explained_variance_score

explained_variance_score(y_test, predictions)

0.9628173926799743
```

```
[ ] from sklearn.metrics import max_error

max_error(y_test, predictions)

0.10064564713393487
```

```
[ ] from sklearn.metrics import r2_score

r2_score(y_test, predictions)

0.9588270759941651
```

GUI



```
!pip install gradio
```

```
[ ] import gradio as gr
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler

import math
import pandas as pd
import numpy as np
from IPython.display import display

from keras.layers.core import Dense, Activation, Dropout
from keras.layers.recurrent import LSTM
from keras.models import Sequential
from keras.metrics import mean_squared_error
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split

import time

from sklearn.metrics import explained_variance_score
from sklearn.metrics import max_error
from sklearn.metrics import r2_score

[ ] def greet(file_obj):
    df = pd.read_csv(file_obj.name)
    df = df.drop(['Date', 'High', 'Low', 'Adj Close'], axis = 'columns')
    df = df.drop(df.index[0:1321], axis = 0)
```



```
unroll_length = 50
X_train = unroll(X_train, unroll_length)
X_test = unroll(X_test, unroll_length)
y_train = y_train[-X_train.shape[0]:]
y_test = y_test[-X_test.shape[0]:]

def build_basic_model(input_dim, output_dim, return_sequences):
    model = Sequential()
    model.add(LSTM(
        input_shape=(None, input_dim),
        units=output_dim,
        return_sequences=return_sequences))

    model.add(LSTM(
        100,
        return_sequences=False))

    model.add(Dense(
        units=1))
    model.add(Activation('linear'))

    return model

model = build_basic_model(input_dim = X_train.shape[-1], output_dim = unroll_length, return_sequences=True)

start = time.time()
model.compile(loss='mean_squared_error', optimizer='adam')

def standardize(train, test):

    mean = np.mean(train, axis=0)
    std = np.std(train, axis=0)+0.000001

    X_train = (train - mean) / std
```

```

X_train = (train - mean) / std
X_test = (test - mean) / std
return X_train, X_test

standardize(X_train, X_test)

model.fit(X_train, y_train, epochs=1, validation_split=0.05)

predictions = model.predict(X_test)

# plt.rcParams['figure.figsize'] = (16, 10)

# ax = fig.add_subplot(111)

# fig = plt.figure()

plt.plot(y_test, '#00FF00', label='Adjusted Close')
plt.plot(predictions, '#0000FF', label='Predicted Close')

plt.ylabel("Price in Rupees")
plt.xlabel("Trading Days")

plt.legend()

# ax.set_title("Actual vs Predicted Stock Price")
# ax.legend(loc='upper left')

trainScore = model.evaluate(X_train, y_train, verbose=0)
trainScorePrint = 'Train Score: %.8f MSE (%.8f RMSE)' % (trainScore, math.sqrt(trainScore))

testScore = model.evaluate(X_test, y_test, verbose=0)
testScorePrint = 'Test Score: %.8f MSE (%.8f RMSE)' % (testScore, math.sqrt(testScore))

explained_variance_score = explained_variance_score(y_test, predictions)

```

```

trainScore = model.evaluate(X_train, y_train, verbose=0)
trainScorePrint = 'Train Score: %.8f MSE (%.8f RMSE)' % (trainScore, math.sqrt(trainScore))

testScore = model.evaluate(X_test, y_test, verbose=0)
testScorePrint = 'Test Score: %.8f MSE (%.8f RMSE)' % (testScore, math.sqrt(testScore))

explained_variance_score = explained_variance_score(y_test, predictions)
explained_variance_scorePrint = 'Explained Variance Score: %.8f EVS' % (explained_variance_score)

# max_error = max_error(y_test, predictions)

# r2_score = r2_score(y_test, predictions)

return fig, trainScorePrint, testScorePrint, explained_variance_score

gr.Interface(greet, "file", [gr.outputs.Plot(type="auto"), "text", "text", "text"]).launch()

```

FILE OBJ



^NSEI (2).csv

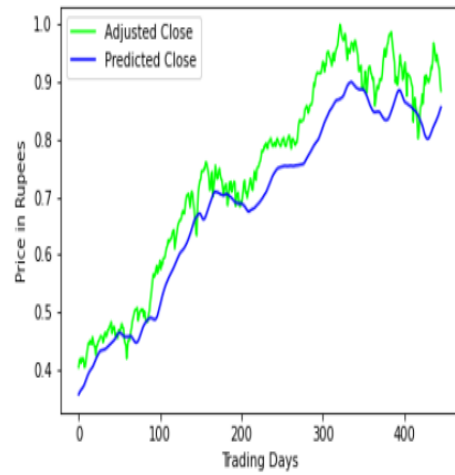
272.0 KB

Clear

Submit

OUTPUT 1

21.0s



OUTPUT 2

Train Score: 0.00070117 MSE (0.02647958 RMSE)

OUTPUT 3

Test Score: 0.00397076 MSE (0.06301393 RMSE)

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 Bose¹, 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.