# STOCK MARKET FORECASTING

**SEMINAR-1 REPORT**

***Submitted by***


**NIRANJAN P – RA2011003020078**

**SUBASH ARAVINDH B – RA2011003020095**

**ADITYA J P – RA2011003020090**


Under the guidance of

**Ms. SWATHI R, M.E.,**

**Ms. PABITHA, M.E.,**

**(Assistant Professor, Department of Computer Science and Engineering)**

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

***of***

**BACHELOR OF TECHNOLOGY**

***in***

**COMPUTER SCIENCE AND ENGINEERING**

***of***

**FACULTY OF ENGINEERING AND TECHNOLOGY**


**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**RAMAPURAM CAMPUS, CHENNAI-600089**

**NOV 2022**

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University Under Section 3 of UGC Act, 1956)

# BONAFIDE CERTIFICATE

Certified that the Seminar-I report titled "**STOCK MARKET FORECASTING**" is the bonafide work of "**NIRANJAN.P [RA2011003020078], SUBASH ARAVINDH.B [RA2011003020095], ADITYA J P [RA2011003020090]**" submitted for the course 18CSP103L Seminar – I. This report is a record of successful completion of the specified course evaluated based on literature reviews and the supervisor. No part of the Seminar Report has been submitted for any degree, diploma, title, or recognition before.

**SIGNATURE**

Ms. SWATHI R , M.E.,
Assistant Professor
Computer Science & Engineering
SRM Institute of Science and Technology
Ramapuram, Chennai.

**SIGNATURE**

Dr. K. RAJA, M.E., Ph.D.,
Professor and Head
Computer Science & Engineering
SRM Institute of Science and Technology
Ramapuram, Chennai.

Submitted for the Seminar-1 Viva Voce Examination held on …… at SRM Institute of Science and Technology, Ramapuram Campus, Chennai-600089.

**EXAMINER 1**

**EXAMINER 2**

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, RAMAPURAM, CHENNAI – 89

## DECLARATION

We hereby declare that the entire work contained in this project report titled **STOCK MARKET FORECASTING** has been carried out by **NIRANJAN [RA2011003020078], SUBASH ARAVINDH B [RA2011003020095], ADITYA J P [RA2011003020090]** at SRM Institute of Science and Technology, Ramapuram Campus, Chennai- 600089, under the guidance of Mrs. Swathi, Assistant Professor, Department of Computer Science and Engineering.

Place: Chennai

Date:

| | | |
|---|---|---|
| NIRANJAN P | SUBASH ARAVINDH B | ADITYA J P |
| RA2011003020078 | RA2011003020095 | RA2011003020090 |

# ABSTRACT

In this project we attempt to implement a machine learning approach to predict stock prices. Machine learning is effectively implemented in forecasting stock prices. The objective is to predict the stock prices 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." Intraday 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 can 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.

In the era of big data, deep learning for predicting stock market prices and trends has become even more popular than before. We collected 2 years of data from Chinese stock market and proposed a comprehensive customization of feature engineering and deep learning-based model for predicting price trend of stock markets.

The proposed solution is comprehensive as it includes pre-processing of the stock market dataset, utilization of multiple feature engineering techniques, combined with a customized deep learning-based system for stock market price trend prediction.

We conducted comprehensive evaluations on frequently used machine learning models and conclude that us proposed solution outperforms due to the comprehensive feature engineering that we built. The system achieves overall high accuracy for stock market trend prediction.

With the detailed design and evaluation of prediction term lengths, feature engineering, and data pre-processing methods, this work contributes to the stock analysis research community both in the financial and technical domains.

## TABLE OF CONTENTS

| SR. NO. | CONTENTS | PAGE NO. |
|---|---|---|

# 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 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 smallinitial amounts of money, low risk compared to the risk of opening new business or the need for a high salary career. Stock markets are affected by many factors causing theuncertainty 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 analyzed.

Time-series prediction is a common technique widely used in manyreal-world applications such as weather forecasting and financial market prediction. It usesthe continuous data in a period to predict the result in the next time unit. Many time-series prediction algorithms have shown their effectiveness in practice.

The most commonalgorithms now are based on Recurrent Neural Networks (RNN), as well as its special type Long-short Term Memory (LSTM) and Gated Recurrent Unit (GRU). The stock market is a typical area that presents time-series data and many researchers' studies on it and proposed various models. In this project, LSTM model is used to predict the stock price

## 1.1 PROBLEM STATEMENT

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

The stock market appears in the news every day. You hear about it every time it reaches a new high or a new low. The rate of investment and business opportunities in the Stock market can increase if an efficient algorithm could be devised to predict the short-term price of an individual stock.

Previous methods of stock predictions involve the use of Artificial Neural Networks and Convolution Neural Networks which has an error loss at an average of 20%. In this report, we will see if there is a possibility of devising a model using Recurrent Neural Network which will predict stock price with a less percentage of error.

Let us see the data on which we will be working before we begin implementing the software to anticipate stock market values. In this section, we will examine the stock price of Microsoft Corporation (MSFT) as reported by the National Association of Securities Dealers Automated Quotations (NASDAQ). The stock price data will be supplied as a Comma Separated File (.csv), that may be opened and analyzed in Excel or a Spreadsheet.

MSFT's stocks are listed on NASDAQ and their value is updated every working day of the stock market. It should be noted that the market does not allow trading on Saturdays and Sundays, therefore there is a gap between the two dates. The Opening Value of the stock, the Highest and Lowest values of that stock on the same days, as well as the Closing Value at the end of the day, are all indicated for each date.

The Adjusted Close Value reflects the stock's value after dividends have been declared (too technical!). Furthermore, the total volume of the stocks in the market is provided, with this information, it is up to the job of a Machine Learning/Data Scientist to look at the data and develop different algorithms that may extract patterns from the historical data of the Microsoft Corporation stock.

# 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 forcefor this area of interest. And there are many challenges involved in this process which needsto be walked all over in order to attain proper outcomes out of them. In this survey we analyzed basic methodology that usually happens in this process and measures that are to be taken to overcome the challenges being faced.

In present days so many peoples are interested in investing money in stock market for earning more in short period of time. Here, in stock market consist of many numbers of company shares along with prices in stock market every minute stock price will changes depending on the company environment and country economic structure decisions.

In stock market there are many brokers for handing the stocks buying and selling between the clients and company. In previous year's there is difficult to predict the stock market because lack of technology and knowledge but in present days technology will increases day by day for that we can predict the stock market easily when compared to past here we can predict stock price by analysing the previous data by using machine learning techniques.

From these techniques we can use neural network (it is means that it is interconnected with networking it looks like human neural brain structure) and simple moving average method. Stock market prediction is always a challenging task because it is highly volatile and dynamic. Many methods have been proposed to forecast the future directions of the stock market. Moreover, news is one of the most significant factors impacting people's reaction in the stock market. Recently, the number of online news have rocketed which make it hard for the investors to cover all the latest information.

For years, the stock market prediction has just depended on the historical market data. Researchers applied a variety of algorithms such as: Moving average, Multiple Kernel Learning, Support Vector Machines, and other techniques to analyse the stock market's behaviour. Although, this research had a promising result, they could not predict the stock market accurately because researchers tried to predict the future prices from the historical prices with such a random behaviour of the stock market and there are no justification for it.

## 2.2 EXISTING METHODS

### 2.2.1 Stock Market Prediction Using Machine Learning

In the finance world stocktrading is one of the most important activities. Stock market prediction is an act of trying todetermine 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 stockbrokerswhile making the stock predictions.

The programming language is used to predict the stockmarket 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 andthen uses the acquired knowledge for an accurate prediction.

In this context this study usesa machine learning technique called Support Vector Machine (SVM) to predict stock pricesfor the large and small capitalizations and in the three different markets, employing prices with both daily and up-to-the-minute frequencies.

The efficient market hypothesis posits that stock prices are a function of information and expectations, and that newly revealed information about a company's prospects is immediately reflected in the current stock price.

This would imply that all publicly known information about a company, which obviously includes its price history, would already be reflected in current price of the stock. Accordingly, changes in the stock price reflect release of new information, changes in the market generally, or random movements around the value that reflects the existing information.

### 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 ofScience 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 priceof a stock market index. Three artificial intelligence techniques, namely, neural networks (NN), support vector machines and neuro-fuzzy systems are implemented in forecasting thefuture price of a stock market index based on its historical price information. Artificial intelligence techniques can 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 randomwalk (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-Index.

The results showed that the three techniques could 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 toshow that the three techniques can disprove the weak form of market efficiency.

The resultsshow 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 OnTick Data

The research work done by Dharmaraja Selvamuthu, Vineet Kumarand 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 Prior work has proposed effective methods to learn event representations thatcan 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 andemotions 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, scriptevent 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. 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 themto 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 pricerises. The efficient market hypothesis states that it is not possible to predict stock prices andthat 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 fortraders 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 canalso 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 positivecorrelation between changes equity prices and investment is, in general, significant. Hence, monetary authorities should monitor reactions of share prices to monetary policy and theireffects 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 analyze the stockprices 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 industryfocused 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 systemthat integrates mathematical functions, machine learning, and other external factors such asnews' sentiments for the purpose of achieving better stock prediction accuracy and issuingprofitable trades.

Particularly, we aim to determine the price or the trend of a certain stockfor the coming end-of-day considering the first several trading hours of the day. To achievethis goal, we trained traditional machine learning algorithms and created/trained multiple deep learning models taking into consideration the importance of the relevant news. Variousexperiments 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. ofBusiness Administration Korea University Seoul, Korea. Predicting the price correlation oftwo 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 twoindividual stocks. RNN's are competent in understanding temporal dependencies. The useof 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-indexmodel 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 methodsto learn event representations that can capture syntactic and semantic information over textcorpus, 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 fordistinguishing event pairs when there are only subtle differences in their surface realizations. To address this issue, this paper proposes to leverage external common-senseknowledge about the intent and sentiment of the event.

Experiments on three event-relatedtasks, 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.

### 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, JajatiKeshari Sahoo Department of Computer Science & Information Systems, BITS Pilani K.K.Birla Goa campus, India Division of Mathematical Sciences, Nanyang TechnologicalUniversity, Singapore c Department 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 analyze their effectiveness in forecasting out-of-sample directional movements of constituent stocks of the S&P 500 from January 1993till December 2018 for intraday trading.

We introduce a multi-feature setting consisting notonly 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 marketin 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 Liu1 Hongyang Yang,Qian Chen4,Runjia ZhangLiuqing Yang Bowen Xiao Christina Dan Wang Electrical Engineering, 2Department of Statistics, 3Computer Science, Columbia University, 3AI4Finance LLC., USA, Ion Media Networks, USA, Department of Computing, ImperialCollege, 6New 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 thatdecides 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 ownstock 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 testingis analyzed via trading performance. Moreover, it incorporates important trading constraintssuch as transaction cost, market liquidity and the investor's degree of risk-aversion. FinRLis 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.

(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 includesecurities 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 nearpast, we faced huge financial problems due to huge drop in price of shares in stock marketsworldwide. 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.This phenomenon can be brought under control especially by strict monitoring and instancestock market analysis. If we can analyze 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 aboutinvestment, which cannot be done without thorough analysis of the market. If we can predictthe stock market by analyzing 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 Longshort-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:

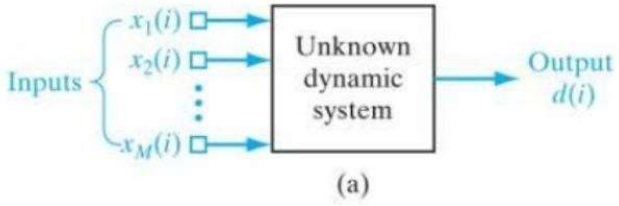
LSTM is a special network structure with three “gate” structures. Three gates are placed in an LSTM unit, called input gate, forgetting gate and output gate.While information enters the LSTM’s network, it can be selected by rules. Only the information conforms to the algorithm will be left, and the information 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 tofind an optimization algorithm that requires less resources and has faster convergence speed.

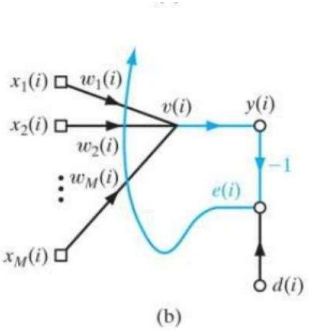
- Used Long Short-term Memory (LSTM) with embedded layer and the LSTM neural network with automatic encoder.
- LSTM is used instead of RNN to avoid exploding and vanishing gradients.
- In this project python is used to train the model, MATLAB is used to reduce dimensions of the input. MySQL is used as 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.




LMS Inputs and Outputs



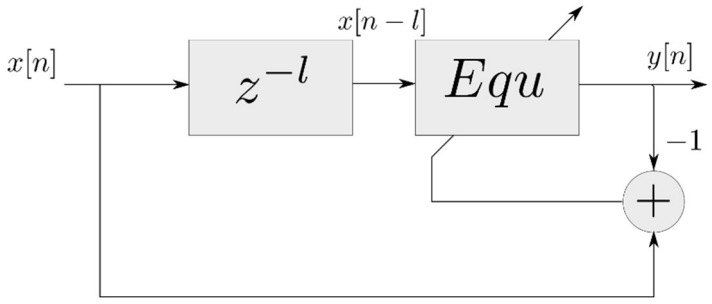

LMS updating weights

In general, we don't know exactly if the problem can be solved verywell with linear approach, so we usually test a linear and a non-linear algorithm. Since theinternet 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 mimetites 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, ourfilter 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.

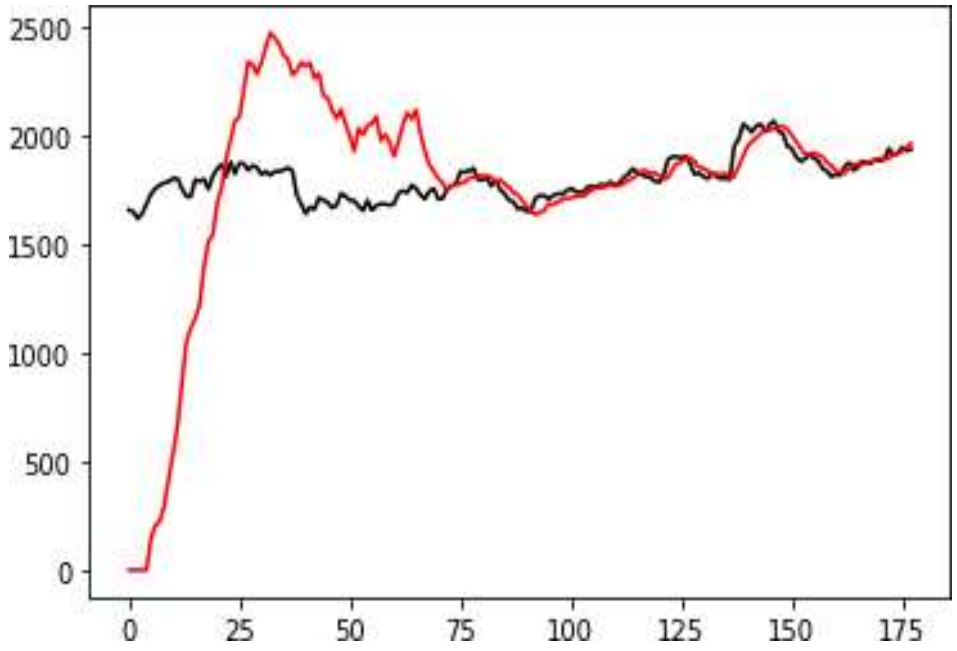
# The system:

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

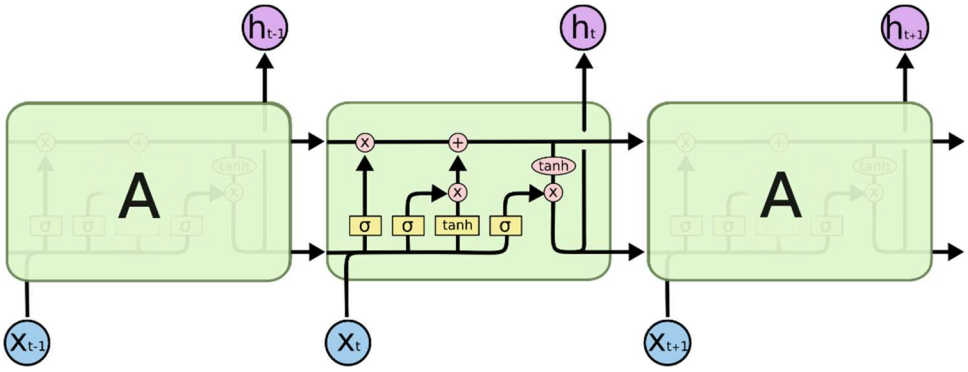



LMS updating weights

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



## 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 theinformation 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 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 hi-1 and x_t.
2. Creating a vector containing all possible values that can be added (as perceived fromh_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:

1. Creating a vector after applying tanh function to the cell state, thereby scaling the values to the range -1 to +1.
2. Making a filter using the values of h_t-1 and x_t, such that it can regulate the valuesthat need to be output from the vector created above. This filter again employs a sigmoid function.
3. 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.

**ALGORITHM:**

```
#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 LSTM Algorithm (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 getPredictonsFromModel (model, X)
  predictions = model.predict(X)
  return predictions
  epochs = 100
```

**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 thesoftware.
- 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. Theyrepresent some of the core functions.

**Non-Functional requirements**

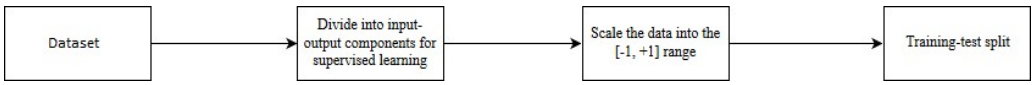
Product properties

- 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 stocktrader and other stakeholders in stock market.
- Efficiency: maintaining the possible highest accuracy in the closing stock prices inshortest 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.

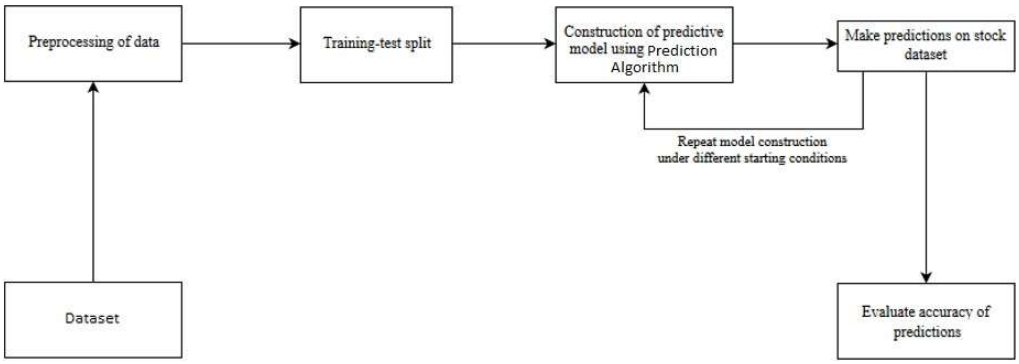
## 3.1.1 SYSTEM ARCHITECTURE

1) Preprocessing of data




Pre-processing of data

2) Overall Architecture

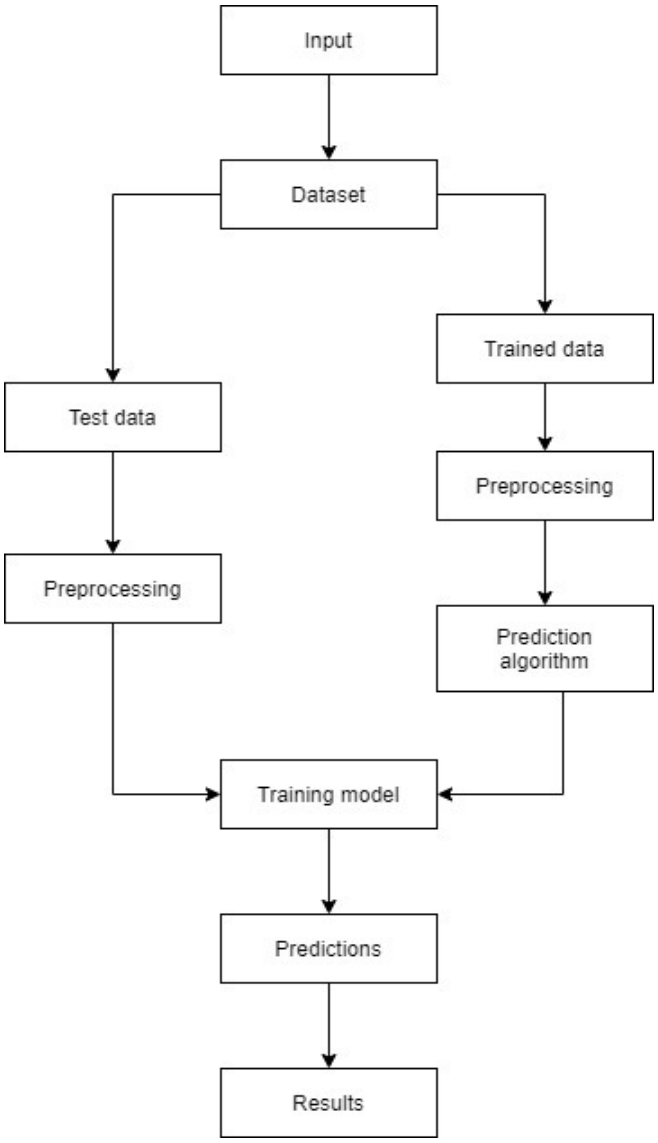



Overall Architecture

# CHAPTER 4
# MODULE
# DESCRIPTION

## 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.




Training and prediction

## 4.2 UML Diagrams

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

The kind of the diagram is defined by the primary graphical symbolsshown on the diagram. For example, a diagram where the primary symbols in the contentsarea 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 nestedinside a use case. Consequently, the boundaries between the various kinds of diagrams arenot 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.

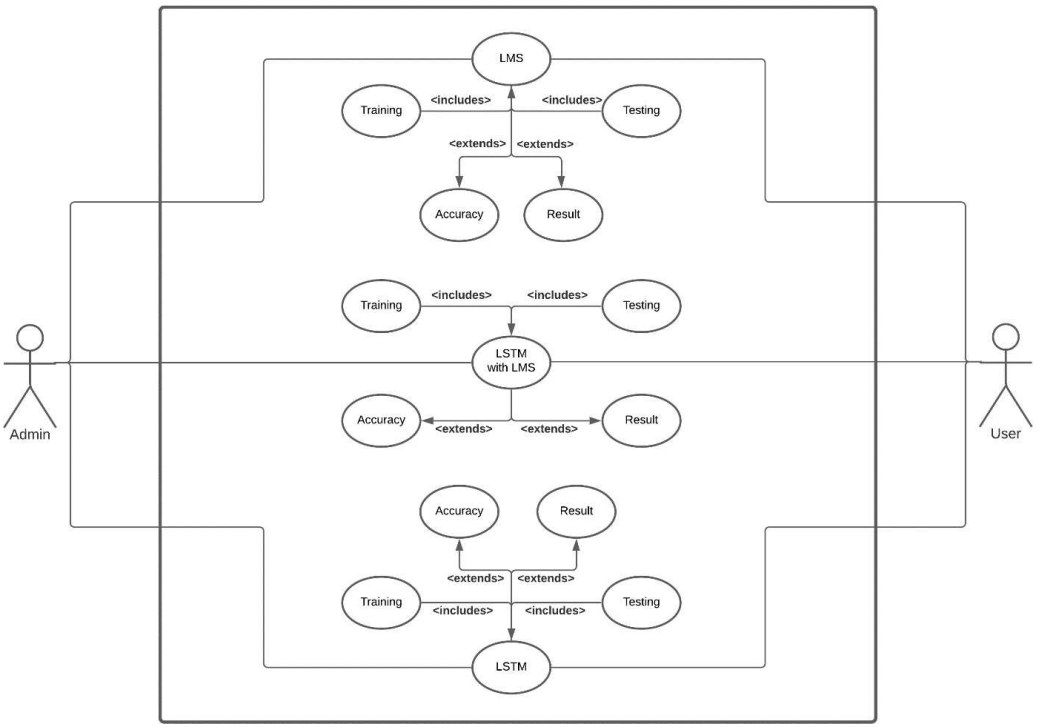
Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation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



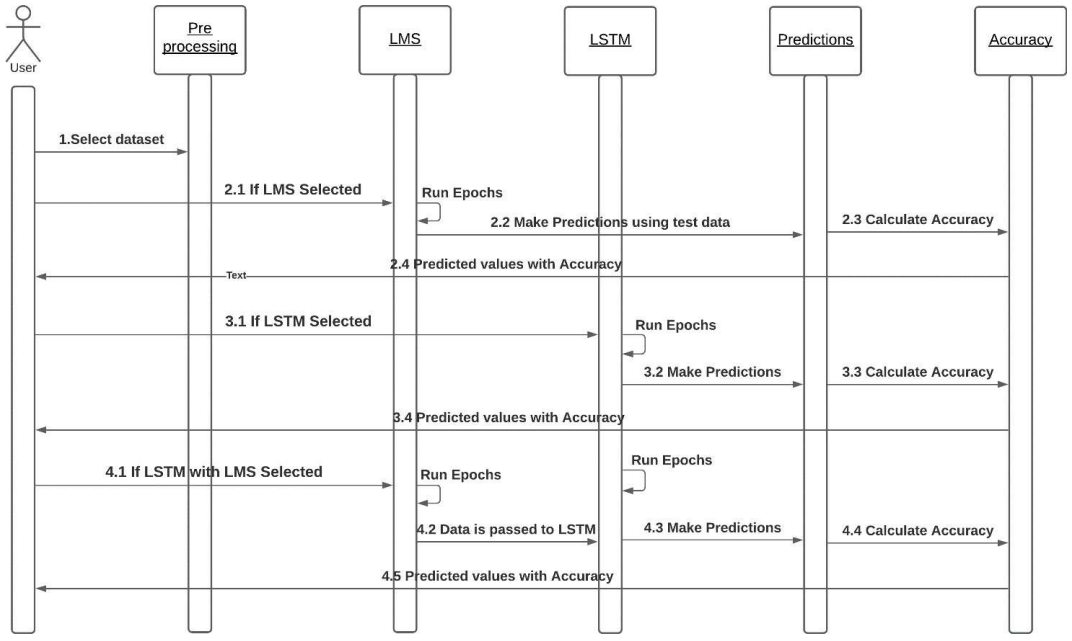

Using LMS, LSTM and LSTM with LMS in the system

### 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 knownas event diagrams or event scenarios.

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

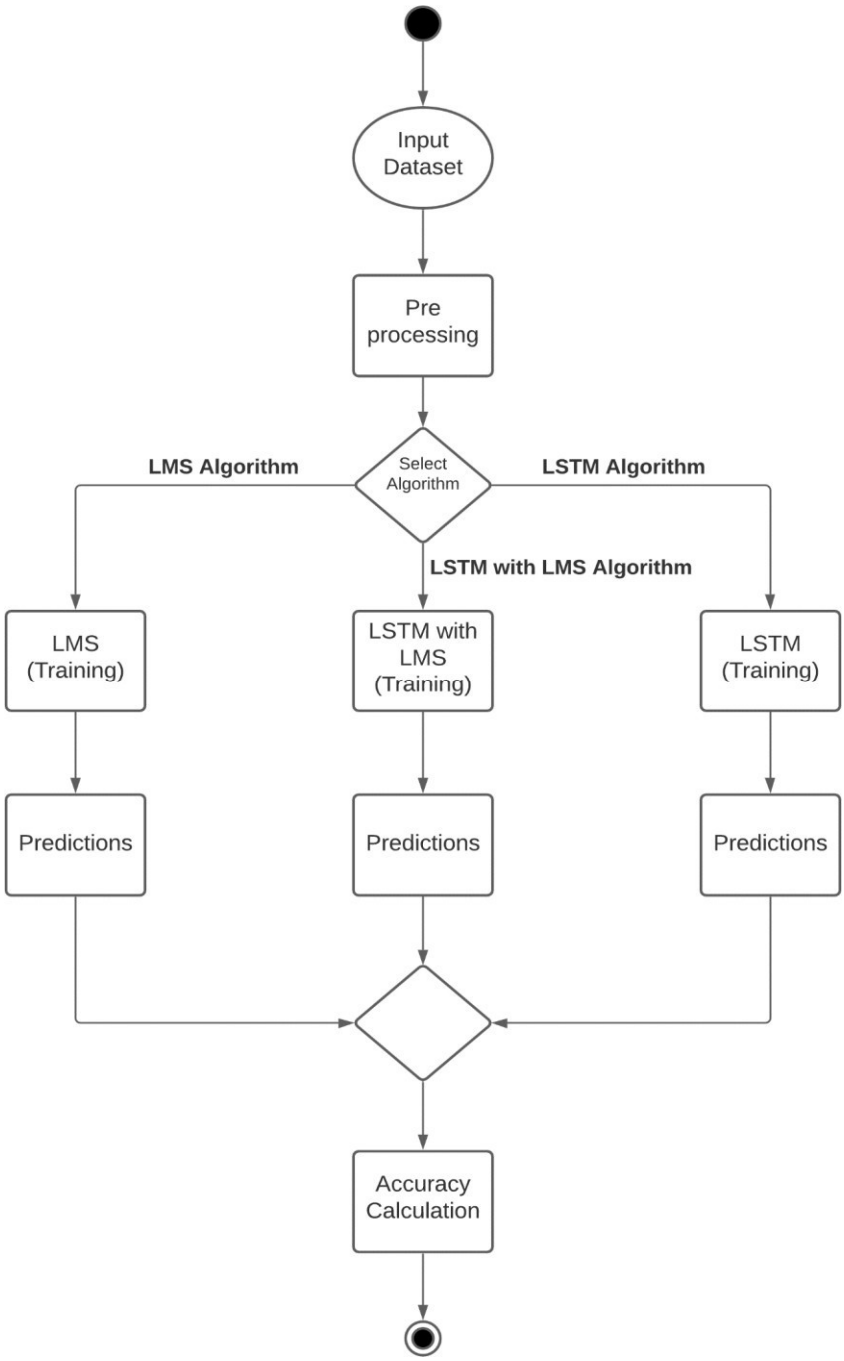
- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.




Execution based on model selection

### 4.2.3 Activity Diagram

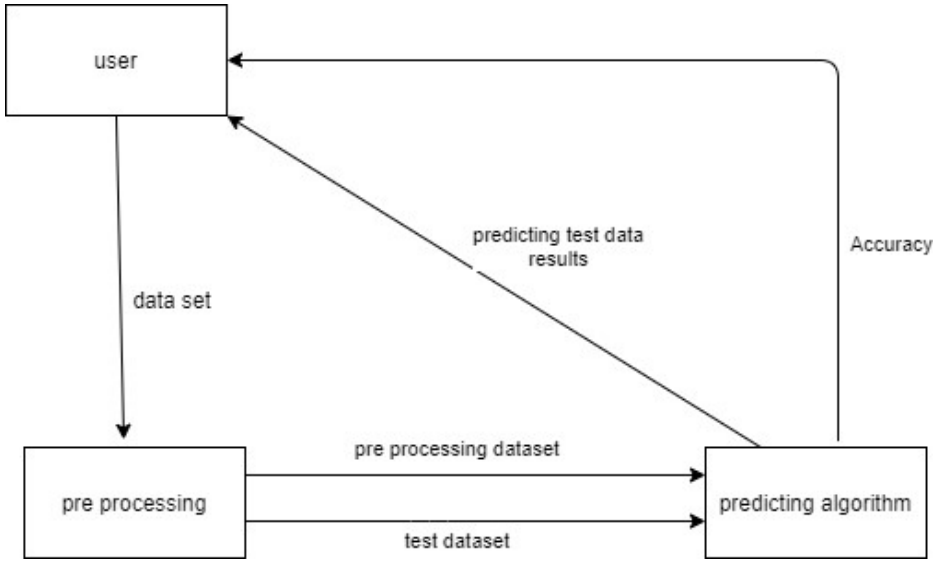
An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showingthe various decision paths that exist while the activity is being executed.



### 4.2.4 Collaboration Diagram

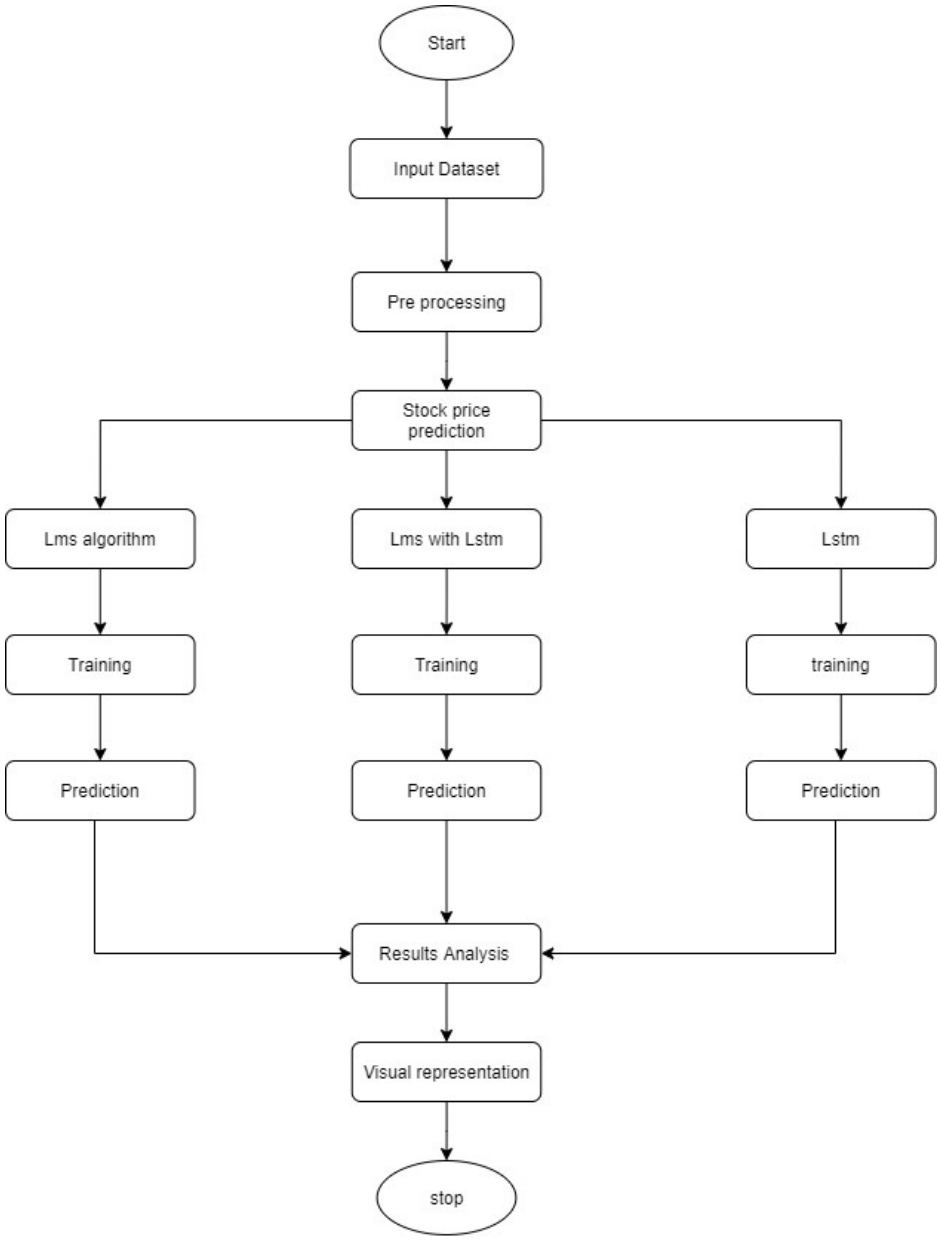
Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration is used by MODULE Descriptions to define and clarify the roles of the objects that perform a particular flow of events of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

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



### 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 variouskinds, and their order by connecting the boxes with arrows.

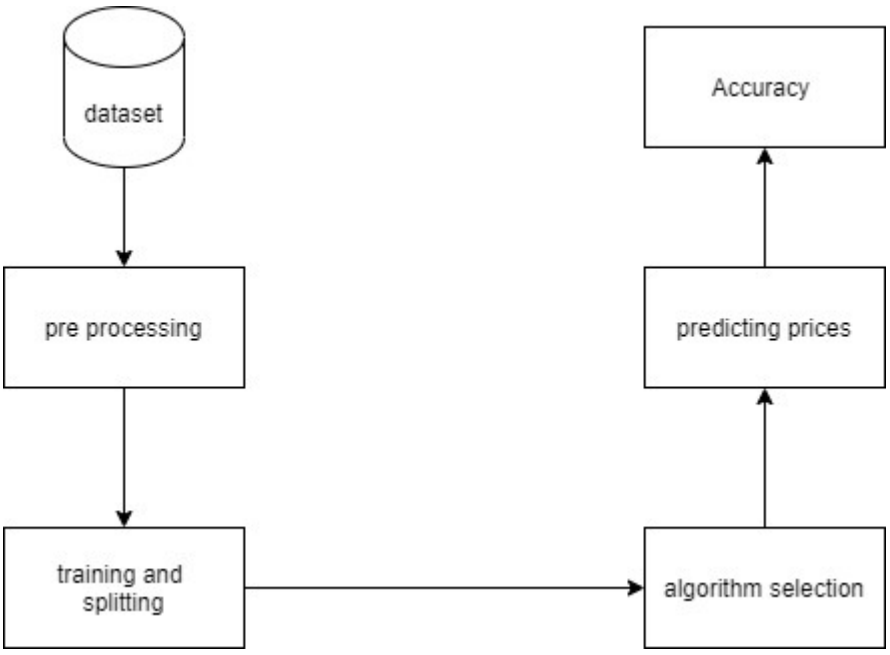



Flow of execution

### 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 for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus ona system's components that are often used to model the static implementation view of a system.




Components present in the system

# CHAPTER 5
# EXPERIMENT ANALYSIS

## 5.1 System configuration

This project can run on commodity hardware. We ran the entire projecton an Intel I5 processor with 8 GB Ram, 2 GB Nvidia Graphic Processor, it also has 2 cores which run at 1.7 GHz and 2.1 GHz respectively. The 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 Collab is used for data pre-processing, model training andprediction.
- Operating System: windows 7 and above or Linux based OS or MAC OS.

## 5.2 Sample code

```
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 getColumnsData(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-l-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[-l:].values,
        "predicted": y[-l:]
    }

    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
```

Epoch Printing Callback

```
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=co

    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
```

Get Predictions from Model

```
def getPredictonsFromModel(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=co

    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 *
```

```python
@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=updateEpoch

        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 = getPredictonsFromModel(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
    )
```

## 5.3 Sample Input and Output:

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

Min and Max of columns in Google Dataset

Nifty50

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

Min and Max of columns in Nifty50 Dataset

Reliance

| | 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 |

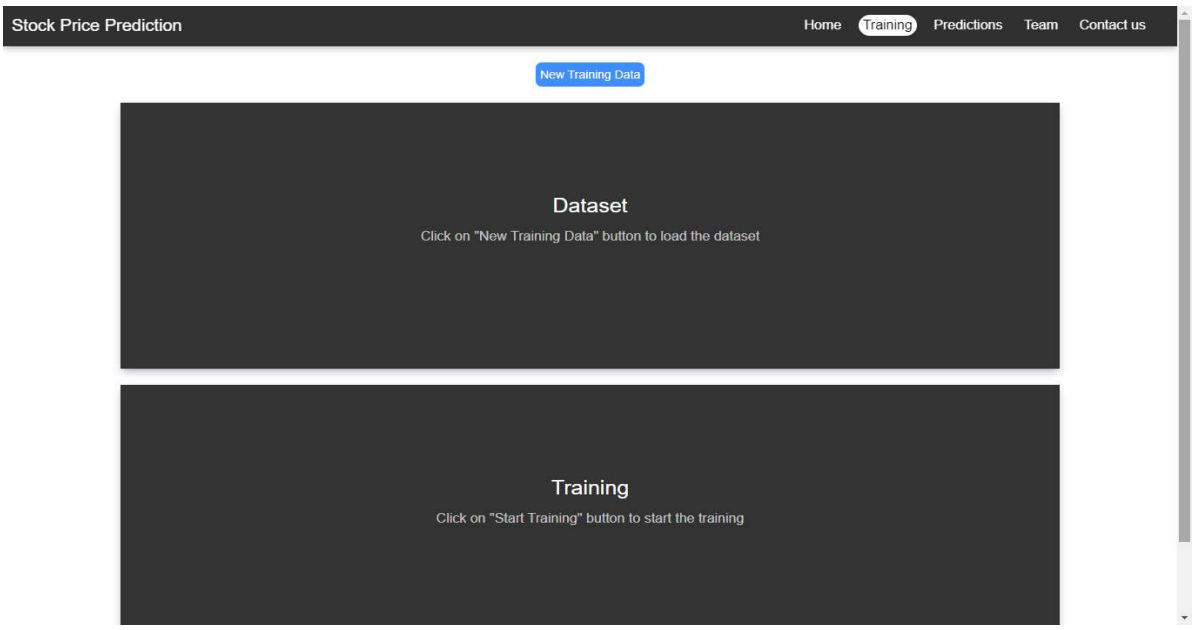
Min and Max of columns in Reliance

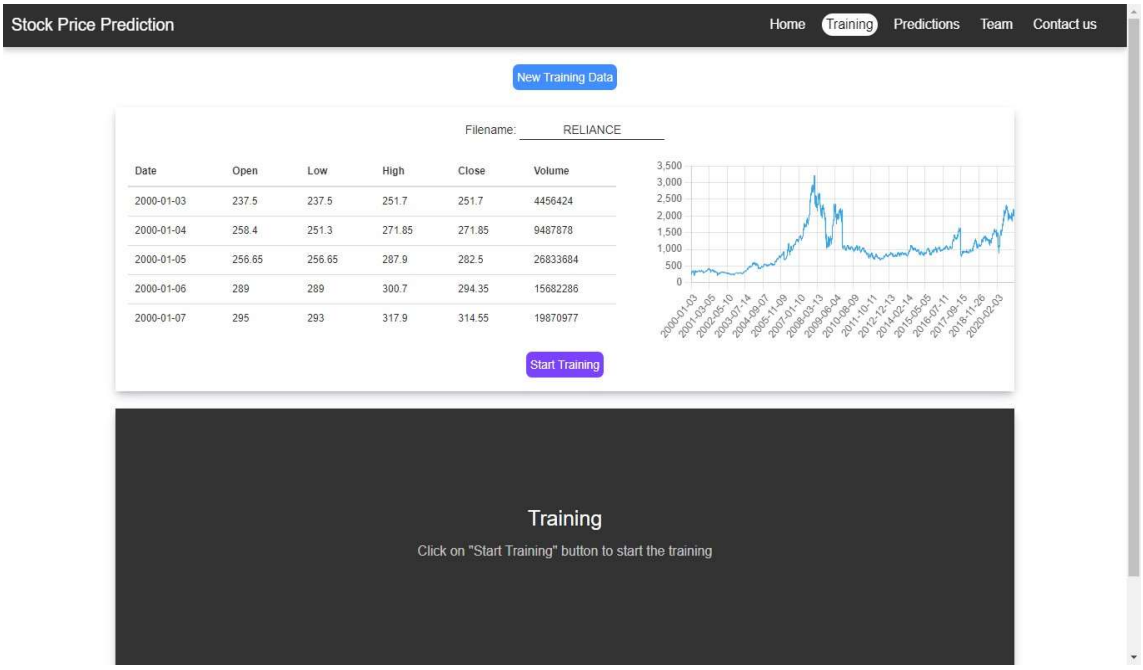
Sample Output:

Trade Close = 214.07

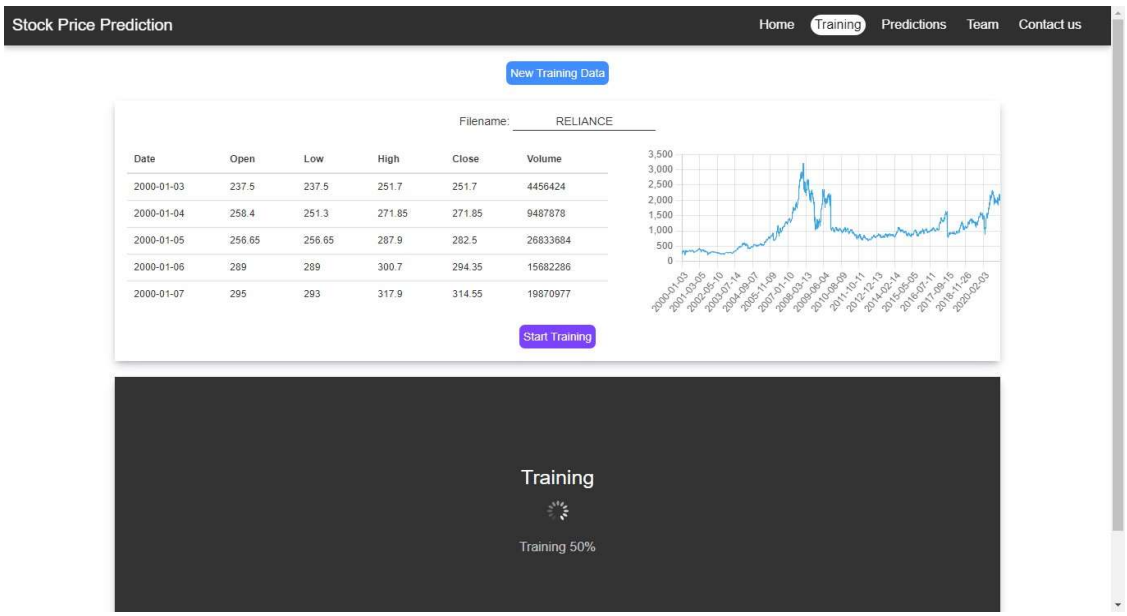
## 5.4 Website Pages

Home Page

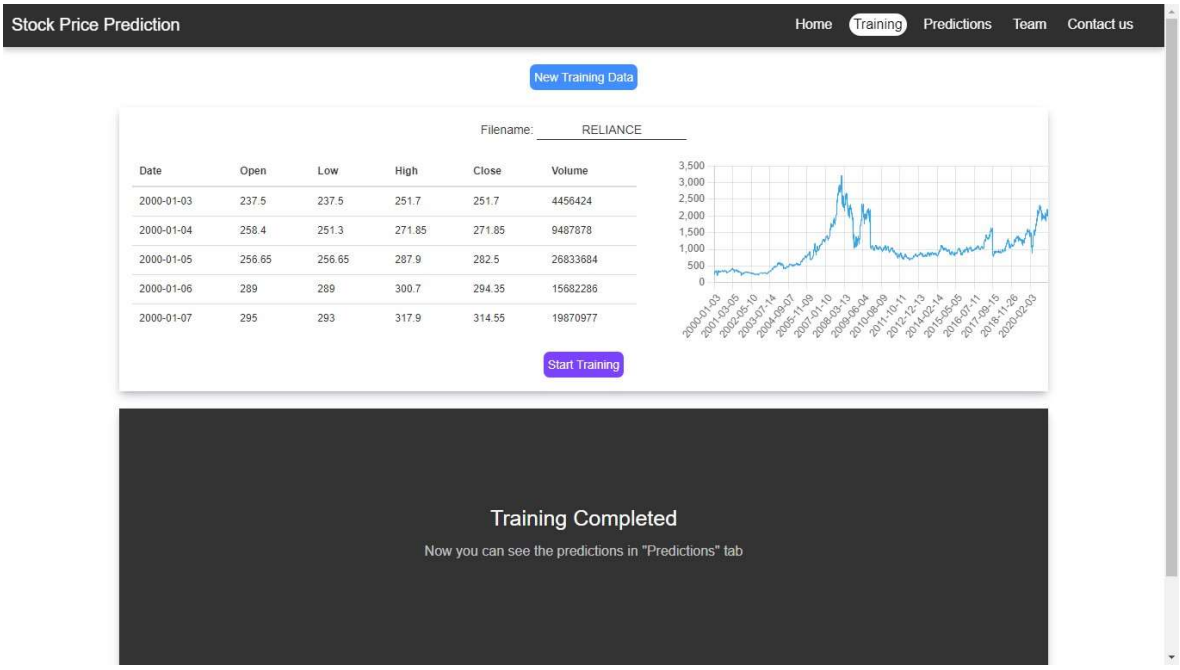





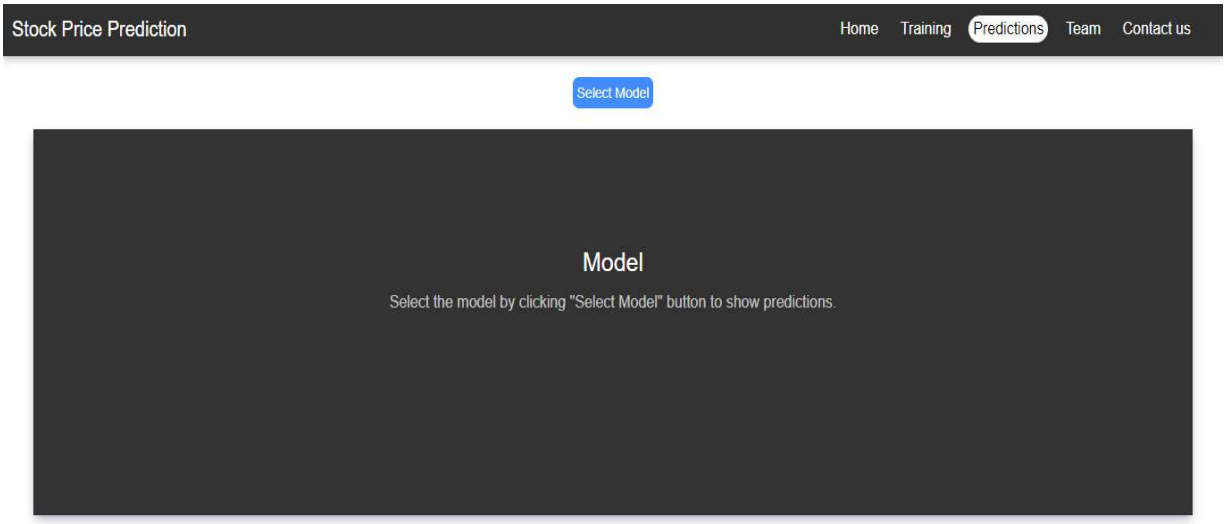

Training Page: After Selecting the dataset




Training Page: While Training

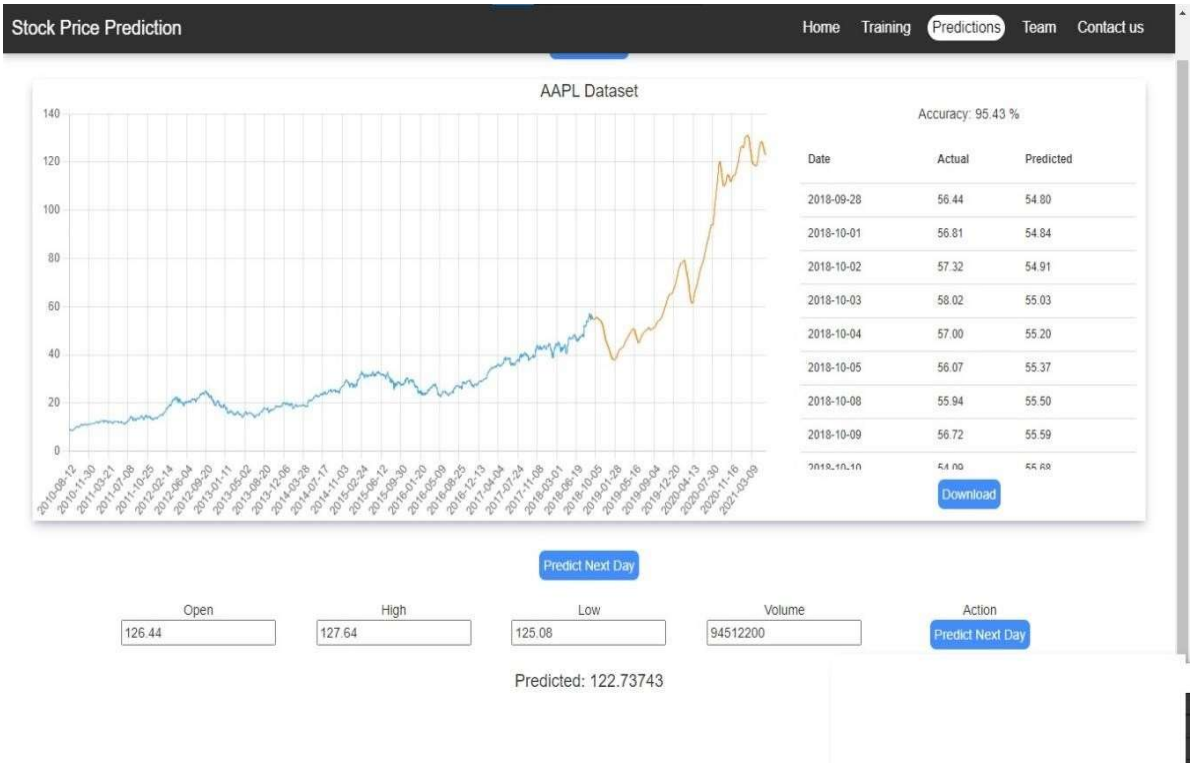



Training Page: Training Completed




Predictions Page

# PREDICTIONS 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 |

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 |

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 |

Epochs for Reliance Dataset using LSTM

### 5.5.2 LSTM with LMS

#### 5.5.2.1 Google

| epochs | Accuracy | MSE | RMSE |
|---|---|---|---|
| 10 | 92.5615 | 339.549 | 18.4269 |
| 20 | 94.2892 | 219.856 | 14.8276 |
| 30 | 94.6971 | 169.259 | 13.01 |
| 40 | 95.1746 | 141.106 | 11.8788 |
| 50 | 94.747 | 161.208 | 12.6968 |

| epochs | Accuracy |
|---|---|
| 100 | 95.50810593088478 |
| 200 | 93.24950439506634 |
| 300 | 94.68220175615014 |
| 400 | 96.29794053164949 |
| 500 | 95.589282359809 |

Epochs for Google Dataset using LSTM with LMS

#### 5.5.2.2 Nifty50

| epochs | Accuracy | MSE | RMSE |
|---|---|---|---|
| 10 | 90.14171 | 1.50E+06 | 1225.58 |
| 20 | 94.41587 | 5.52E+05 | 742.9043 |
| 30 | 94.54524 | 5.47E+05 | 739.3578 |
| 40 | 96.65602 | 2.68E+05 | 517.9008 |
| 50 | 96.79688 | 2.50E+05 | 500.3757 |

| epochs | Accuracy |
|---|---|
| 100 | 97.67512047219317 |
| 200 | 91.36568721761229 |
| 300 | 92.21746083762834 |
| 400 | 88.3763795882347 |
| 500 | 91.26283879244312 |

Epochs for Nifty50 Dataset using LSTM with LMS

#### 5.5.2.3 Reliance

| epochs | Accuracy | MSE | RMSE |
|---|---|---|---|
| 10 | 95.283656 | 8079.78008 | 89.887597 |
| 20 | 95.055813 | 7370.14221 | 85.849532 |
| 30 | 96.530505 | 4622.22982 | 67.986983 |
| 40 | 95.594117 | 5847.60569 | 76.469639 |
| 50 | 96.681513 | 3858.11477 | 62.113724 |

| epochs | Accuracy |
|---|---|
| 100 | 97.41168889605405 |
| 200 | 97.44153787870181 |
| 300 | 97.72196960171793 |
| 400 | 97.75577851463954 |
| 500 | 97.52291451371063 |

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 , 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.

Our proposed solution is a unique customization as compared to the previous works because rather than just proposing yet another state-of-the-art LSTM model, we proposed a fine-tuned and customized deep learning prediction system along with utilization of comprehensive feature engineering and combined it with LSTM to perform prediction.

By researching into the observations from previous works, we fill in the gaps between investors and researchers by proposing a feature extension algorithm before recursive feature elimination and get a noticeable improvement in the model performance.

Though we have achieved a decent outcome from our proposed solution, this research has more potential towards research in future. During the evaluation procedure, we also found that the RFE algorithm is not sensitive to term lengths other than 2-day, weekly, biweekly. Getting more in-depth research into what technical indices would influence the irregular term lengths would be a possible future research direction. Moreover, by combining latest sentiment analysis techniques with feature engineering and deep learning model, there is also a high potential to develop a more comprehensive prediction system which is trained by diverse types of information such as tweets, news, and other text-based data.

### 6.2 Future work

The project is to be extended to this application for predicting cryptocurrency trading.
Sentiment analysis to be added for better understanding.

# 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. Vijayakumar, 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 Rondale, Harshal Patel, Shraddha Varma, “Predicting Stock Prices Using LSTM” in Undergraduate Engineering Students, Department of Information Technology, Mumbai University, 2015.

[4] Xiong Wen Pang, Yunqiang Zhou, Pan Wang, Weiwei Lin, “An innovative neural network approach for stock market prediction”, 2018

[5] Ishita Parmar, Navanish Agarwal, Sehrish 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.