STOCK PRICE PREDICTION

A Project Report

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

BACHELOR OF ENGINEERING

IN

ARTIFICIAL INTELLIGENCE AND

MACHINE LEARNING

Submitted by:

ABHISHEK VERMA

20BCS6707

Under the Supervision of:

RAMANJOT KAUR

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB

FEBRUARY 2022

Table of Contents

DECLARATION

We, **'ABHISHEK VERMA'** student of **'Bachelor of Engineering in Artificial Intelligence & Machine Learning'** , **session: _2020-2024,** Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work presented in this Project Work entitled **'Stock Price Prediction'** is the outcome of our bona fide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

ABHISHEK VERMA

Student UID's:
(20BCS6707)

Date: 08/03/2022 Place: Chandigarh university

ACKNOWLEDGEMENT

We would like to earnestly acknowledge the sincere efforts and valuable time given by Professor Ramanjot Kaur. Their valuable guidance and feedback have helped us in completing this project. Also, we would like to thank our group members who supported each other in making and completing this project successively. Our sincere efforts finally produced something that we imagined. We would also like to thank faculty who provided us this golden chance. Last but not least we would like to thank everyone who supported and motivated us for pushing ourself beyond our strength.

Thank You.

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." 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’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.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

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

INTRODUCTION

An advancement in the fundamental aspects of information technology over the last few decades has altered the route of businesses. As one of the most captivating inventions, financial markets have a pointed effect on the nation's economy. The World Bank reported in 2018 that the stock market capitalization worldwide has surpassed 68.654 trillion US$. Over the last few years, stock trading has become a center of attention, which can largely be attributed to technological advances. Investors search for tools and techniques that would increase profit and reduce the risk.

However, Stock Market Prediction (SMP) is not a simple task due to its non-linear, dynamic, stochastic, and unreliable nature. SMP is an example of time-series forecasting that promptly examines previous data and estimates future data values. Financial market prediction has been a matter of worry for

analysts in different disciplines, including economics, mathematics, material science, and computer science. Driving profits from the trading of stocks is an important factor for the prediction of the stock market. The stock market is dependent on various parameters, such as the market value of a share, the company's performance, government policies, the country's Gross Domestic Product (GDP), the inflation rate, natural calamities, and so on.

The Efficient Market Hypothesis explains that stock market costs are significantly determined by new information, and follow a random walk pattern, such that they cannot be predicted solely based on past information. This was a widely accepted theory in the past. With the advent of technology, researchers demonstrated that stock market prices could be predicted to a certain extent. Historical market data, combined with the data extracted from social media platforms, can be analyzed to predict the changes in the economic and business sectors. The performance of stock market prediction systems relies intensely on the quality of the features it is using. While researchers have used some strategies for enhancing the stock-explicit features, more attention needs to be paid to feature extraction and selection mechanisms.

The financial market is a dynamic and composite system where people can buy and sell currencies, stocks, equities and derivatives over virtual platforms supported by brokers. The stock market allows investors to own shares of public companies through trading either by exchange or over the counter markets. This market has given investors the chance of gaining money and having a prosperous life through investing small initial amounts of money, low risk compared to the risk of opening new business or the need of high salary career. Stock markets are affected by many factors causing the uncertainty and high volatility in the market. Although humans can take orders and submit them to the market, automated trading systems (ATS) that are operated by the implementation of computer programs can perform better and with higher momentum in submitting orders than any human.

However, to evaluate and control the performance of ATSs, the implementation of risk strategies and safety measures applied based on human judgements are required. Many factors are incorporated and considered when developing an ATS, for instance, trading strategy to be adopted, complex mathematical functions that reflect the state of a specific stock, machine learning algorithms that enable the prediction of the future stock value, and specific news related to the stock being analyzed.

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 timeseries 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.

Recurrent Neural Networks

Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.

Recurrent Neural Networks have loops

In the above diagram, a chunk of neural network, AA, looks at some input $x_t$xt and outputs a value $h_t$ht. A loop allows information to be passed from one step of the network to the next.
These loops make recurrent neural networks seem kind of mysterious. However, if you think a bit more, it turns out that they aren't all that different than a normal neural network. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.

An unrolled recurrent neural network

This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data.

And they certainly are used! In the last few years, there have been incredible success applying RNNs to a variety of problems: speech recognition, language modeling, translation, image captioning… The list goes on. I'll leave discussion of the amazing feats one can achieve with RNNs to Andrej Karpathy's excellent blog post, The Unreasonable Effectiveness of Recurrent Neural Networks. But they really are pretty amazing.

Essential to these successes is the use of "LSTMs," a very special kind of recurrent neural network which works, for many tasks, much much better than the standard version. Almost all exciting results based on recurrent neural networks are achieved with them. It's these LSTMs that this essay will explore.

The Problem of Long-Term Dependencies

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. If RNNs could do this, they'd be extremely useful. But can they? It depends.

Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in "the clouds are in the *sky*," we don't need any further context – it's pretty obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information.

But there are also cases where we need more context. Consider trying to predict the last word in the text "I grew up in France… I speak fluent *French*." Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back. It's entirely possible for the gap between the relevant information and the point where it is needed to become very large.

Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.

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

Thankfully, LSTMs don't have this problem!

LSTM Networks

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.[1] They work tremendously well on a large variety of problems, and are now widely used.

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

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

The repeating module in a standard RNN contains a single layer

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

The repeating module in an LSTM contains four interacting layers

Don’t worry about the details of what’s going on. We’ll walk through the LSTM diagram step by step later. For now, let’s just try to get comfortable with the notation we’ll be using.

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

The Core Idea Behind LSTMs

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It’s very easy for information to just flow along it unchanged.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”

An LSTM has three of these gates, to protect and control the cell state.

Step-by-Step LSTM Walk Through

The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It looks at ht−1ht−1 and xtxt, and outputs a number between 00 and 11 for each number in the cell state Ct−1Ct−1. A 11 represents "completely keep this" while a 00 represents "completely get rid of this."
Let's go back to our example of a language model trying to predict the next word based on all the previous ones. In such a problem, the cell state might include the gender of the present subject, so that the correct pronouns can be used. When we see a new subject, we want to forget the gender of the old subject.

The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, C~tC~t, that could be added to the state. In the next step, we'll combine these two to create an update to the state.
In the example of our language model, we'd want to add the gender of the new subject to the cell state, to replace the old one we're forgetting.

It's now time to update the old cell state, Ct−1Ct−1, into the new cell state CtCt. The previous steps already decided what to do, we just need to actually do it.
We multiply the old state by ftft, forgetting the things we decided to forget earlier. Then we add it∗C~tit∗C~t. This is the new candidate values, scaled by how much we decided to update each state value.
In the case of the language model, this is where we'd actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps.

Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanhtanh (to push the values to be between −1−1 and 11) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.
For the language model example, since it just saw a subject, it might want to output information relevant to a verb, in case that's what is coming next. For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that's what follows next.

Variants on Long Short Term Memory

What I've described so far is a pretty normal LSTM. But not all LSTMs are the same as the above. In fact, it seems like almost every paper involving LSTMs uses a slightly different version. The differences are minor, but it's worth mentioning some of them.

One popular LSTM variant, introduced by Gers & Schmidhuber (2000), is adding "peephole connections." This means that we let the gate layers look at the cell state.

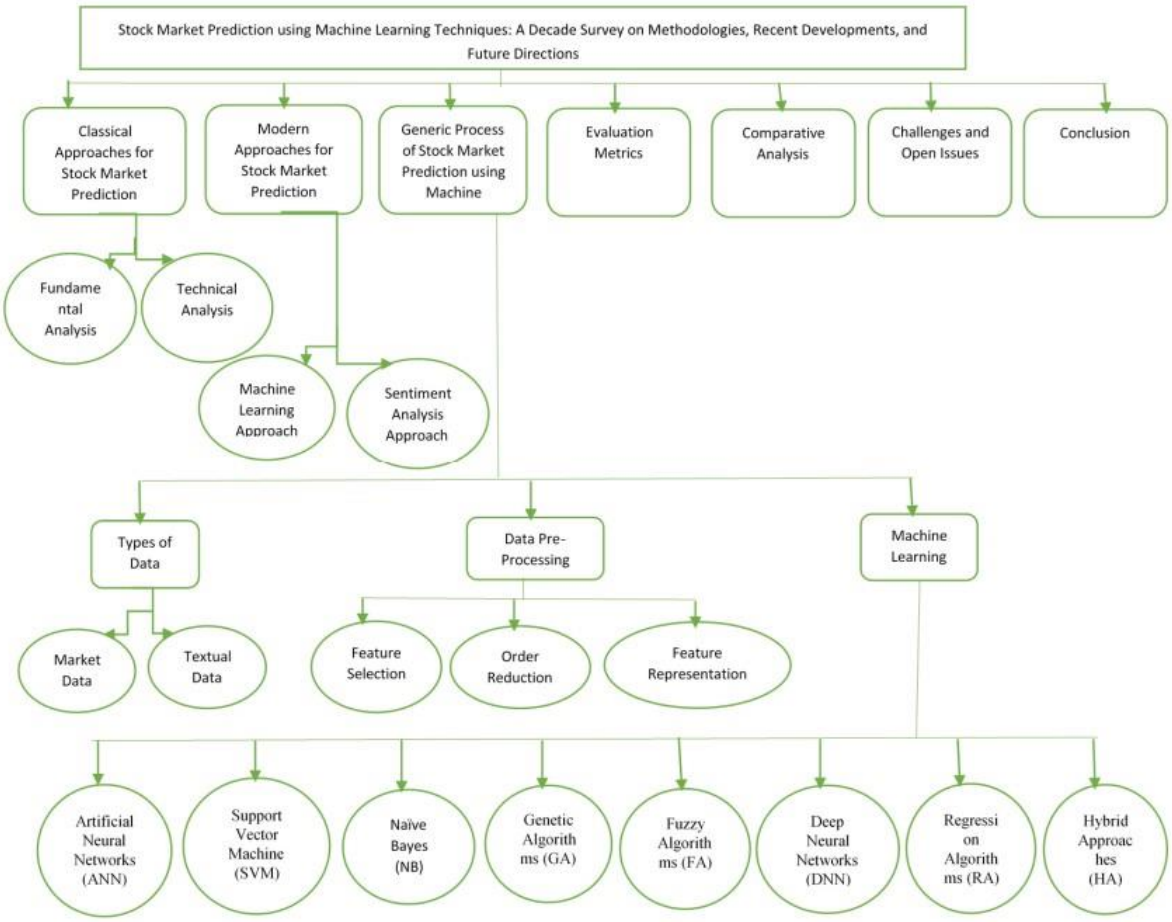
The above diagram adds peepholes to all the gates, but many papers will give some peepholes and not others.

Another variation is to use coupled forget and input gates. Instead of separately deciding what to forget and what we should add new information to, we make those decisions together. We only forget when we're going to input something in its place. We only input new values to the state when we forget something older.

A slightly more dramatic variation on the LSTM is the Gated Recurrent Unit, or GRU, introduced by Cho, et al. (2014). It combines the forget and input gates into a single "update gate." It also merges the cell state and hidden state, and makes some other changes. The resulting model is simpler than standard LSTM models, and has been growing increasingly popular.

These are only a few of the most notable LSTM variants. There are lots of others, like Depth Gated RNNs by Yao, et al. (2015). There's also some completely different approach to tackling long-term dependencies, like Clockwork RNNs by Koutnik, et al. (2014).

Which of these variants is best? Do the differences matter? Greff, et al. (2015) do a nice comparison of popular variants, finding that they're all about the same. Jozefowicz, et al. (2015) tested more than ten thousand RNN architectures, finding some that worked better than LSTMs on certain tasks.



There exist two main traditional approaches to the analysis of the stock markets:
(1) Fundamental Analysis
(2) Technical Analysis

- Fundamental Analysis

Fundamental analysis calculates a genuine value of a sector/company and determines the amount that one share of that company should cost. A supposition is made that, if given sufficient time, the company will move to a cost agreeing with the prediction. If a sector/company is undervalued, then the market value of that company should rise, and conversely, if a company is overvalued, then the market price should fall. The analysis is performed considering various factors, such as yearly fiscal summaries and reports, balance sheets, a future prospectus, and the company's work environment . If stocks are overvalued, then the market price will fall. The two most common metrics used to predict long-term price movements yearly for fundamental analysis are (a) the Price to Earnings ratio (P/E) and (b) the Price by Book ratio (P/B). The P/E ratio is used as a predictor. The companies with a lower P/E ratio yield higher returns than companies with a high P/E ratio . Financial analysts also use this to prove their stock recommendations . Fundamental analysis can be used for the consideration of financial ratios to distinguish poor stocks from quality stocks. The P/B ratio compares the company value specified by the market to the company value specified on paper. If the ratio is high, the company may be overvalued, and the company's value might fall with time. Conversely, if the ratio is low, the company may be underestimated, and the price may rise with time. Of course, fundamental analysis is a powerful method. Still, it has some drawbacks. Fundamental analysis, firstly, lacks adequate knowledge of the rules governing the workings of the system, and secondly, there is non-linearity in the system.

- Technical Analysis

Technical analysis is the study of stock prices to make a profit, or to make better investment decisions. Technical analysis predicts the direction of the future price movements of stocks based on their historical data, and helps to analyze financial time series data using technical indicators to forecast stock prices. Meanwhile, it is assumed that the price moves in a trend and has momentum . Technical analysis uses price charts and certain formulae, and studies patterns to predict future stock prices; it is mainly used by short-term investors. The price would be considered high, low or open, or the closing price of the stock, where the time points would be daily, weekly, monthly, or yearly. Dow theory puts forward the main principles for technical analysis, which are that the market price discounts everything, prices move in trends, and historic trends usually repeat the same patterns. There are several technical indicators, such as the Moving Average (MA), Moving Average Convergence/Divergence (MACD), the Aroon indicator, and the money flow index, etc. The evident flaws of technical analysis are that expert's opinions define rules in technical analysis, which are fixed and are reluctant to change. Various parameters that affect stock prices are ignored. The prerequisite is to overcome the deficiencies of fundamental and technical analysis, and the evident advancement in the modelling techniques has motivated various researchers to study new methods for stock price prediction. A new form of collective intelligence has emerged, and new innovative methods are being employed for stock value forecasting. The methodologies incorporate the work of machine learning algorithms for stock market analysis and prediction.

LITERATURE REVIEW

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) 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 endof-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.

3) 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.

4) 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 nonparametric, 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

5) Stock Price Correlation Coefficient Prediction with ARIMALSTM 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.

6) 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.

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.

1.1 Literature Review Summary

Year and citation	Article Title	Purpose of study	Tools/ Software used	Comparison of technique done	Findings	Data set (if used)	Evaluation parameters
2018	Stock Market Prediction Using Machine Learning	To build a model that predicts stock prices.	• Python 2.7 or higher • Kaggle • Google Collab	exploit stock data to predict future price in real-time.	automatically suggested the stock price	stock.csv	• Accuracy • Complexity
2016	Automated Stock Price Prediction Using Machine Learning	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	• Python 2.7 or higher • Kaggle • Google Collab	accurately predict how investors will react to certain prices and forecast demand for a given stocks	pricing optimization app	stpck.csv	• Accuracy • Complexity

2010	Forecasting the Stock Market Index Using Artificial Intelligence *Techniques*	to use AI techniques to model and predict the future price of a	• Python 2.7 or higher • Kaggle • Google Collab	AI techniques: Neural Networks, SVM and Neuro Fuzzy	Performances of NN,SVM and Neuro Fuzzy	Johannesburg Stock Exchange	• Accuracy • Complexity

		stock market index					
2019	Indian stock market prediction using artificial neural networks on tick data		• Python 2.7 or higher • Kaggle • Google Collab	AI neural networks	Improved predicted prices of stocks		• Accuracy • Complexity
2018	Stock Price Correlation Coefficient Prediction with ARIMALSTM Hybrid Model		• Python 2.7 or higher • Kaggle • Google Collab	RNN: ARIMALSTM	Superior than other financial techniques		• Accuracy • Complexity

2018	An innovative neural network approach for stock market prediction		• Python 2.7 or higher • Kaggle • Google Collab	Deep Learning LSTM	contributions in IMMT for neural networkbased financial analysis		• Accuracy • Complexity
2019	Event Representation Learning Enhanced with External Common-sense Knowledge		• Python 2.7 or higher • Kaggle • Google Collab	External Common sense knowledge	Intent and Sentiment of events		• Accuracy • Complexity

PROBLEM FORMULATION

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%. 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.
The main aim of this project is to predict stock prices using Long short term memory (LSTM).

OBJECTIVES

The proposed work is aimed to carry out work leading to the development of an approach for the Stock Price Prediction Model. The proposed aim will be achieved by dividing the work into the following objectives:

1. Minimizes the risk usually involved in changing prices thanks to its prediction capabilities.
2. Investor can essentially use machine learning to test out various promotions or pricing strategies to understand what their impact may be.
3. To make the pricing decisions of pricing managers more profitable.

METHODOLOGY

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

Long short-term memory network:

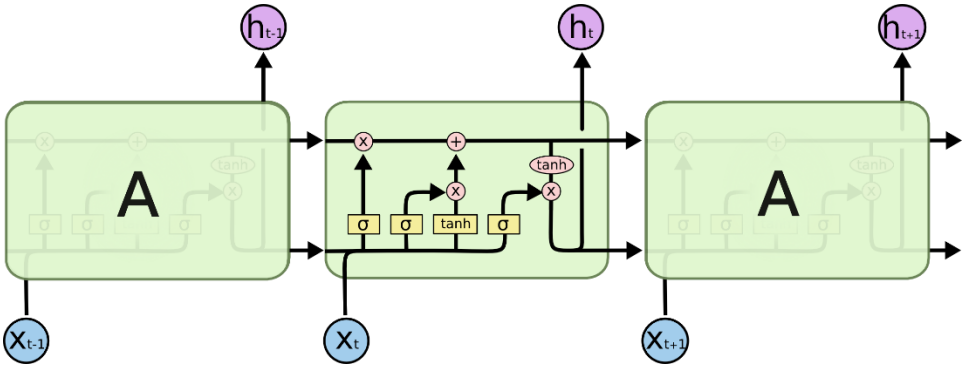
Long short-term memory network (LSTM) is a particular form of recurrent neural network (RNN).

Working of LSTM:

LSTM is a special network structure with three “gate” structures. Three gates are placed in an LSTM unit, called input gate, forgetting gate and output gate. While information enters the LSTM’s network, it can be selected by rules. Only the information conforms to the algorithm will be left, and the information that does not conform will be forgotten through the forgetting gate. The experimental data in this paper are the actual historical data downloaded from the Internet. Three data sets were used in the experiments. It is needed to find an optimization algorithm that requires less resources and has faster convergence speed.

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

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:

- 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.
- 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.
- 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:

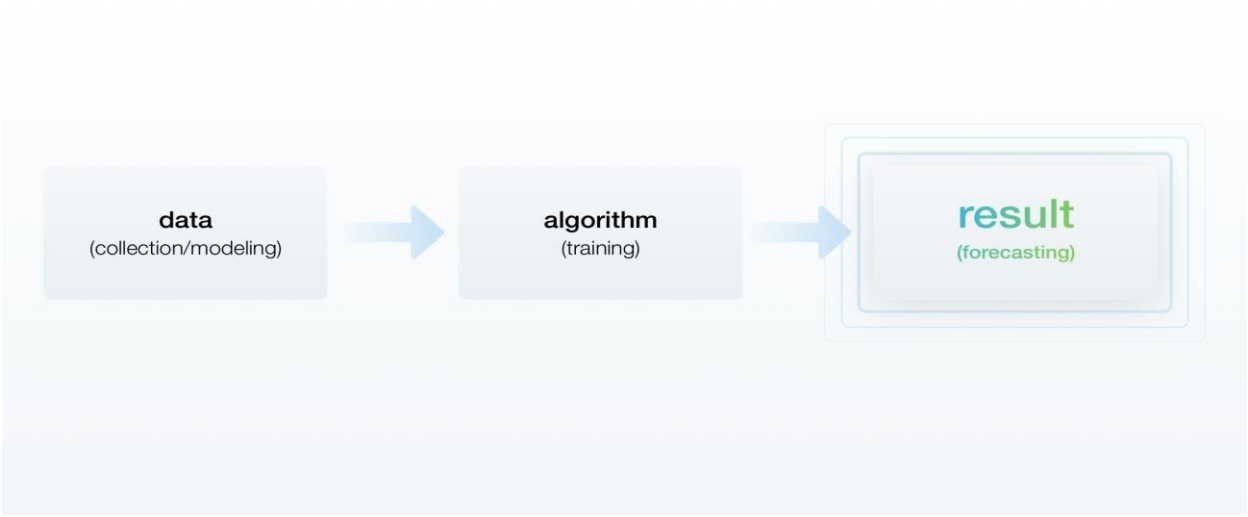
- Creating a vector after applying tanh function to the cell state, thereby scaling the values to the range -1 to +1.
- Making a filter using the values of h_t-1 and x_t, such that it can regulate the values that need to be output from the vector created above. This filter again employs a sigmoid function.

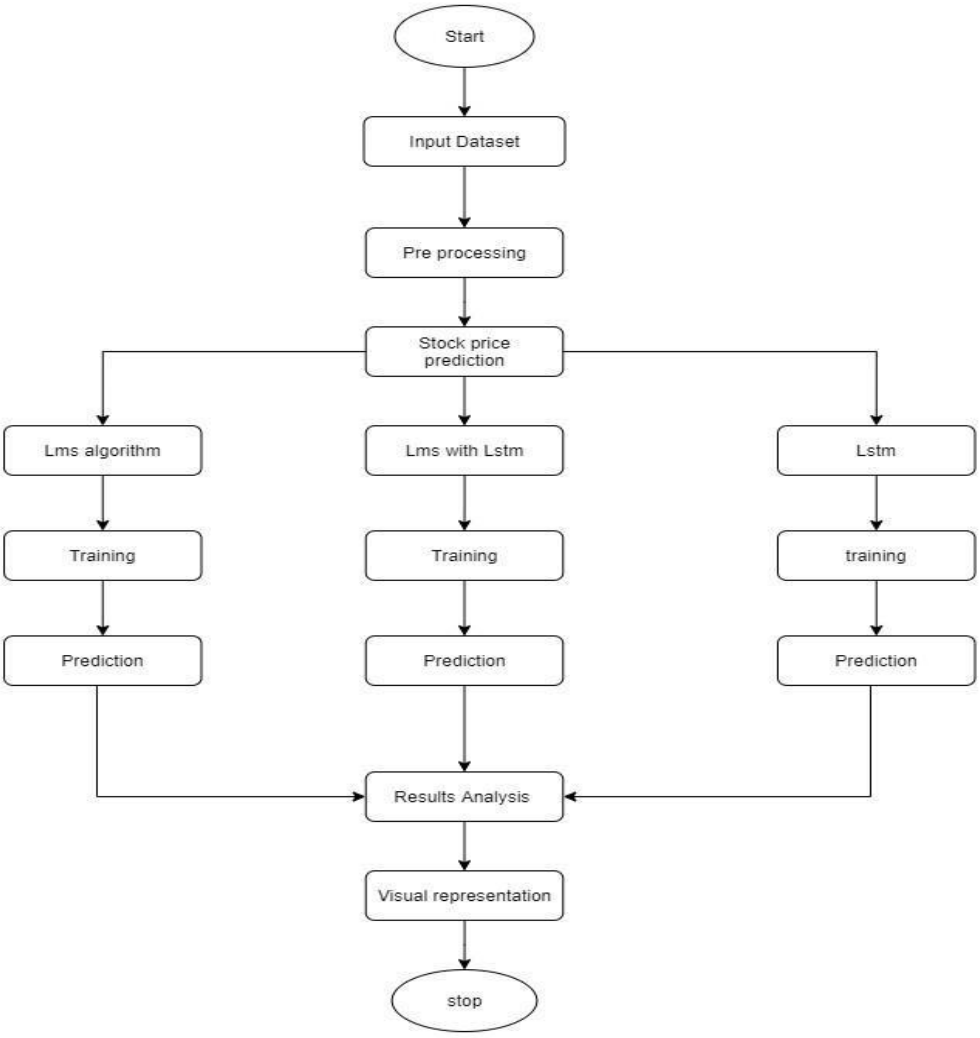
- Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as a output and also to the hidden state of the next cell.

```
# 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 getPredictonsFromModel (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
```

FLOW CHART



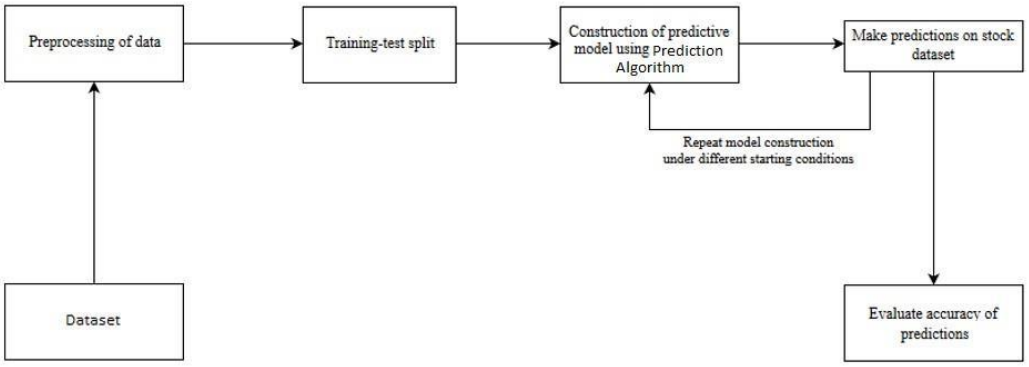


3.1.1 SYSTEM ARCHITECTURE

1) Preprocessing of data

Pre-processing of data

2) Overall Architecture

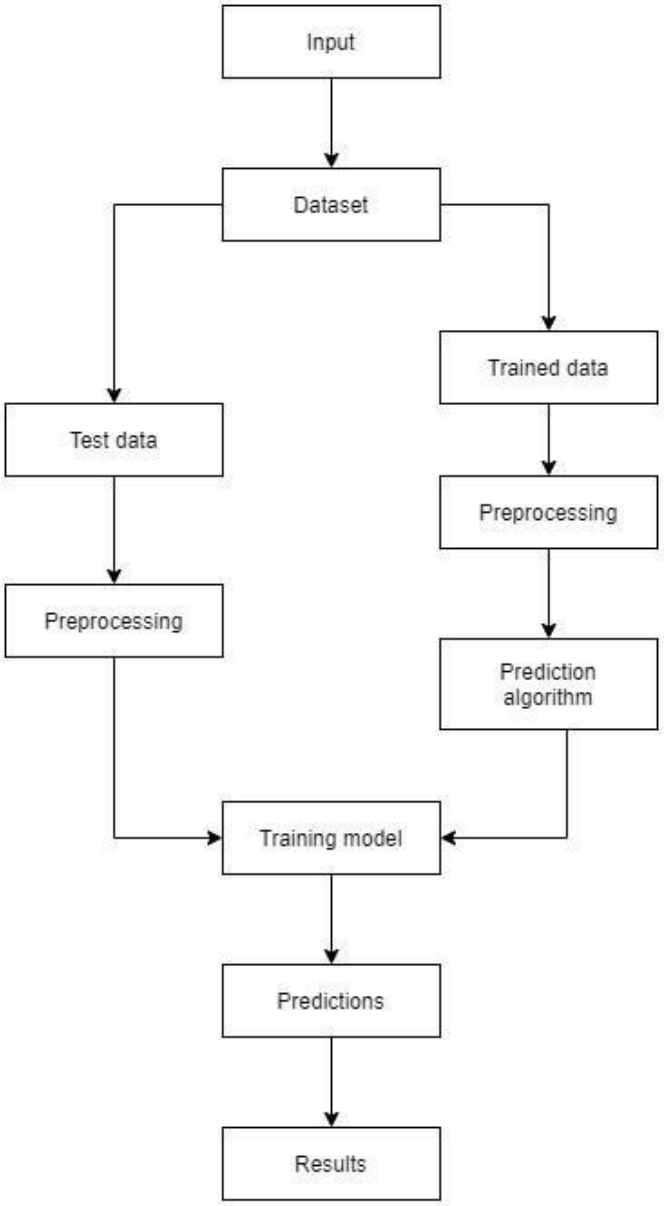


Overall Architecture

DESIGN

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.



Training and prediction

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.

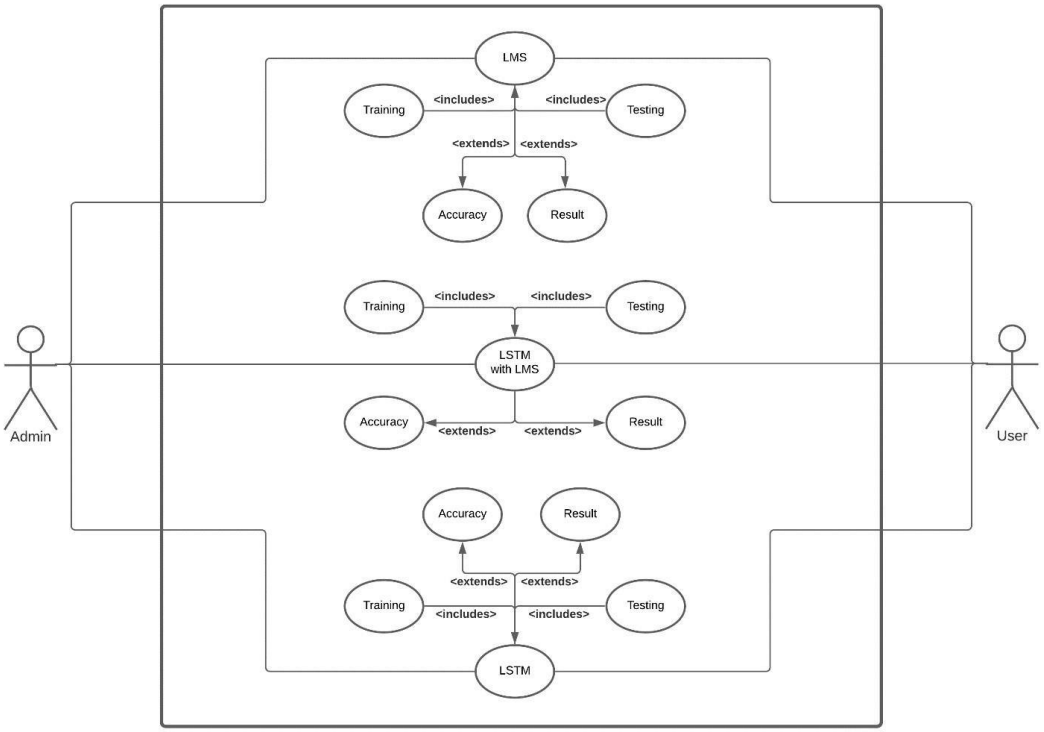
Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation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

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.



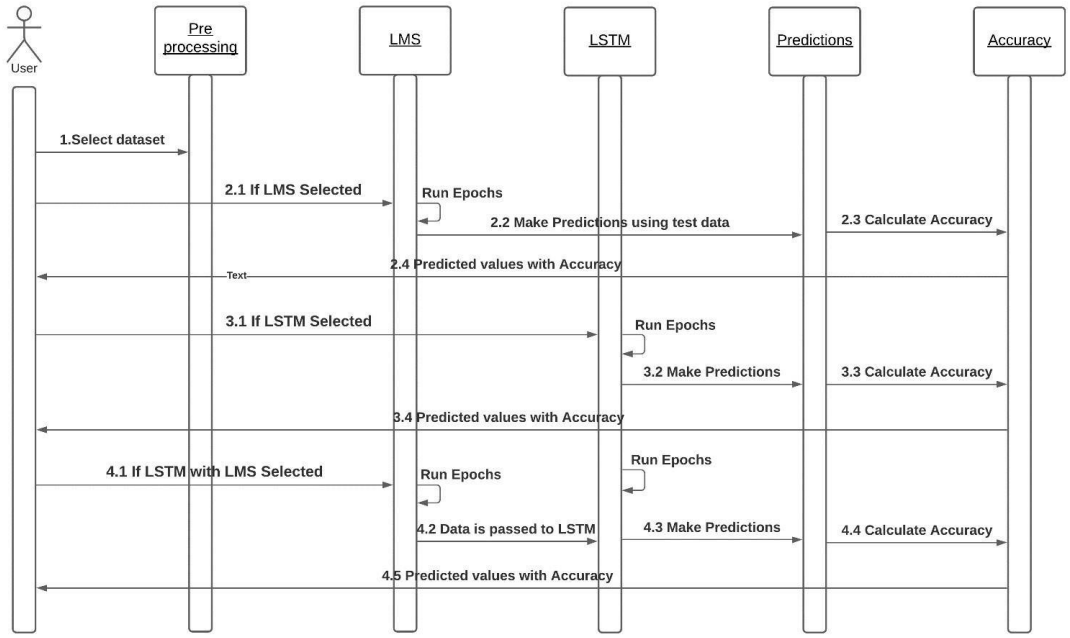
Using LMS, LSTM and LSTM with LMS in the system

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.

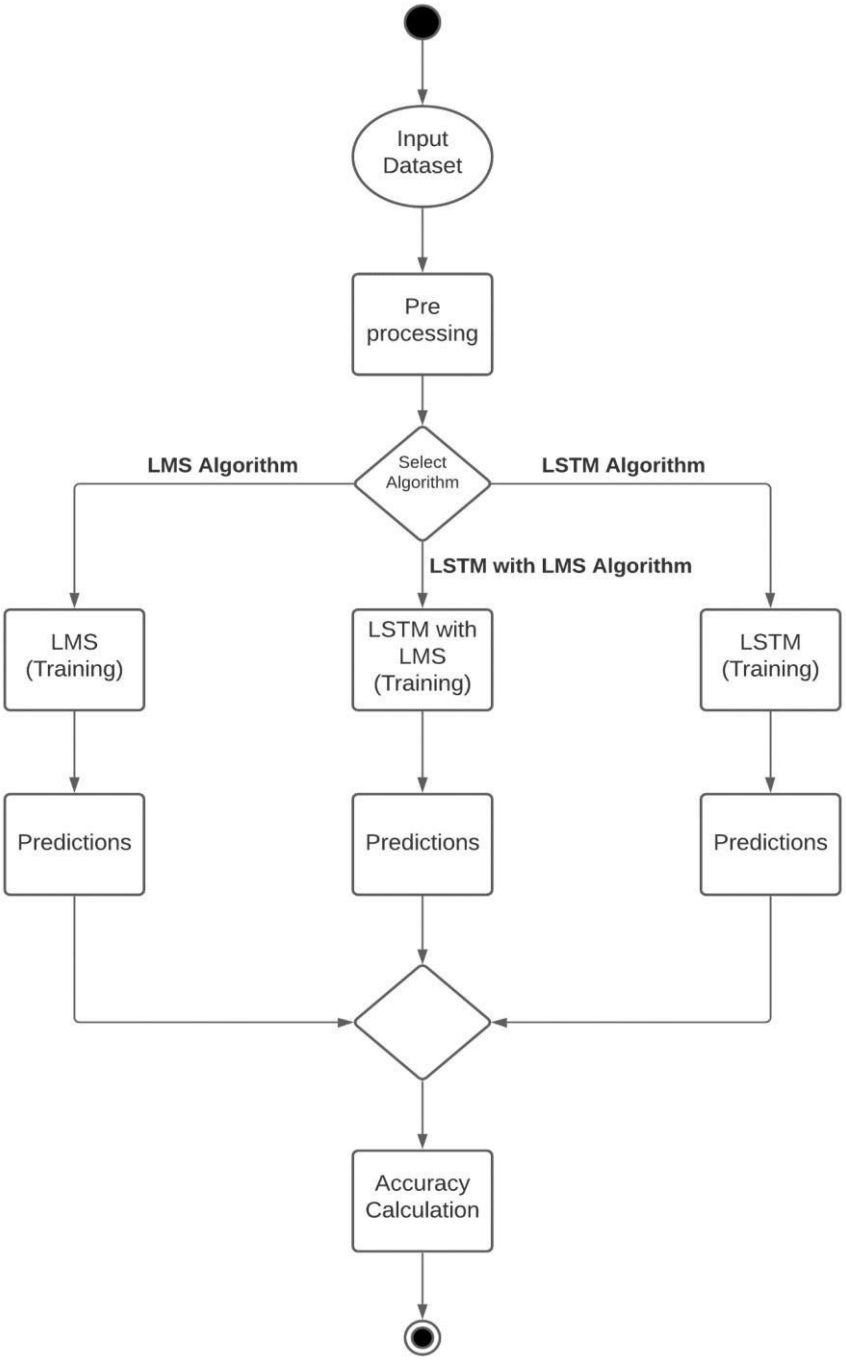


Execution based on model selection

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.



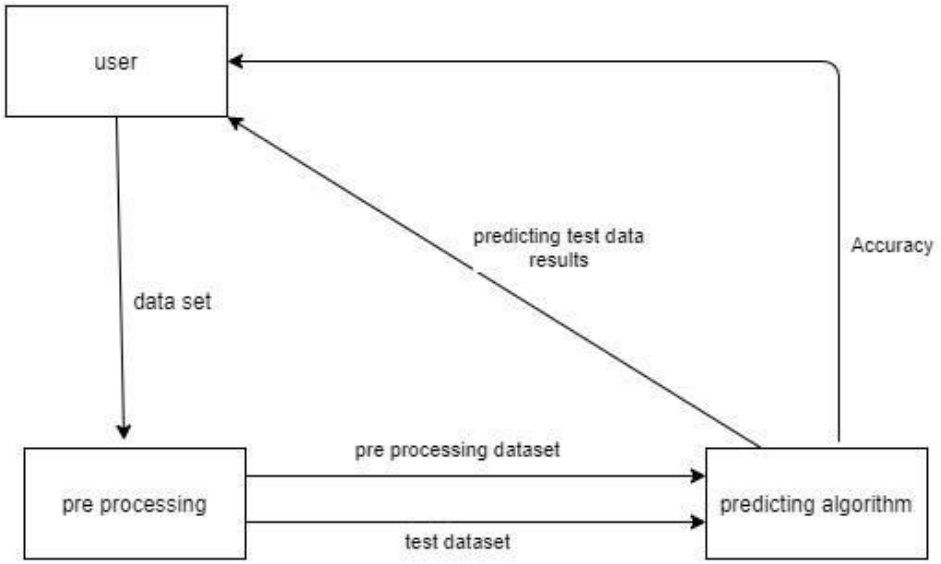
Execution based on algorithm selection

Collaboration Diagram

Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration are used by designers to define and clarify the

roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.

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



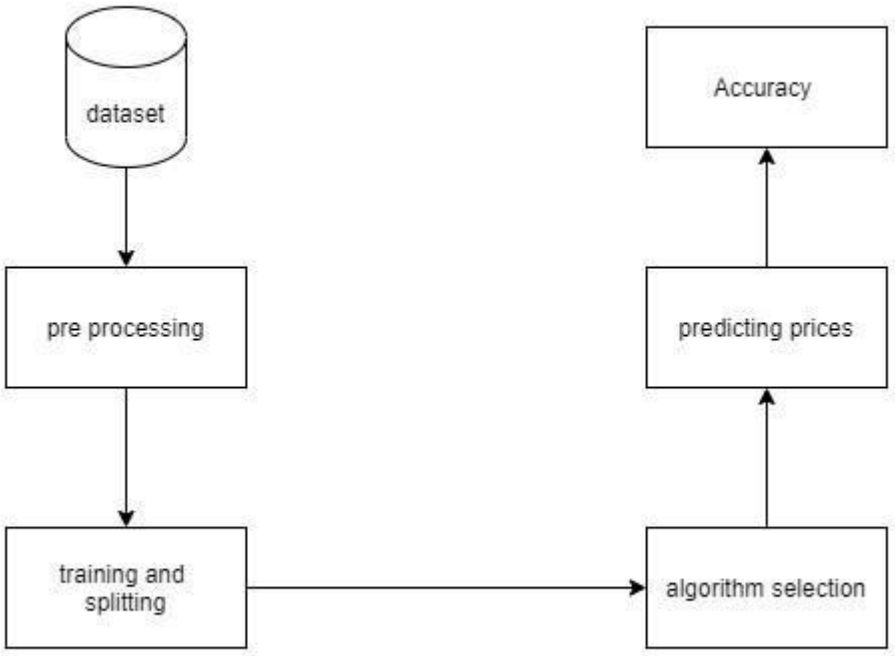
Data transfer between modules

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.



Components present in the system

Sample Code Of ML Model

```
import numpy as np
import pandas as pd
import pandas_datareader as web
import math
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, LSTM
plt.style.use('ggplot')

import warnings

warnings.filterwarnings('ignore')
```

```
df = pd.read_csv("WMT.csv")

#show the dataset
df
```

```
#visualize the closing prices
plt.figure(figsize=(16,8))
plt.title('closing price of Stock')
plt.plot(df['Close'])
plt.xlabel('Date')
plt.ylabel('Closing price')
plt.show()
```

```
#create new database with only required coloumns
data=df.filter(['Close'])
#convert the dataframe to numpy array
dataset=data.values
#get the number of rows to train 80 percent
training_data_len=math.ceil(len(dataset)*0.8)
training_data_len
```

```
#scale the data
scaler=StandardScaler()
scaled_data=scaler.fit_transform(dataset)
print("MEAN of processed data: ",scaled_data.mean())
print("Standard deviation of processed data: ",scaled_data.std())
```

```
#create the training data
#create scaled training dataset
train_data = scaled_data[0:training_data_len, :]
#split the data to x_train and y_train
x_train=[]
y_train=[]
for i in range(60,len(train_data)):
  x_train.append(train_data[i-60:i])
  y_train.append(train_data[i])
```

```
#convert x_train and y_train into numpy arrays
x_train,y_train=np.array(x_train),np.array(y_train)
x_train.shape
```

```
#reshape the data
print("x_train shape before reshaping",x_train.shape)
x_train = np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))
print("x_train shape after reshaping",x_train.shape)
```

```
#build LSTM model
model= Sequential()
model.add(LSTM(200,return_sequences=True,input_shape=(x_train.shape[1], 1)))
model.add(LSTM(200,return_sequences=False))
model.add(Dense(100))
model.add(Dense(50))
model.add(Dense(1))
```

```
#compile the model
model.compile(optimizer='adam',loss='mean_squared_error')
```

```
model.summary()
```

```
#train the model
history = model.fit(x_train,y_train,epochs=20)
```

```
plt.plot(history.history['loss'])
```

```
#create the testing dataset
#createt new array
test_data=scaled_data[training_data_len-60:, :]
#create the dataset x_test and y_test
x_test=[]
y_test=dataset[training_data_len: , :]
for i in range(60,len(test_data)):
  x_test.append(test_data[i-60:i, 0])
```

```
#convert the data to numpy
x_test=np.array(x_test)
```

```
#reshape the data
x_test=np.reshape(x_test,(x_test.shape[0],x_test.shape[1],1))
```

```
#get the model predicted value
predictions =model.predict(x_test)
predictions=scaler.inverse_transform(predictions)
# print(predictions)
```

```
#plot the data
train=data[:training_data_len]
valid=data[training_data_len:]
valid['Predictions']=predictions
```

```
#visualize the data
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Data')
plt.ylabel('Close prise INR')
plt.plot(train['Close'])
plt.plot(valid[['Close','Predictions']])
plt.legend(['Train','Test','Predictions'],loc='lower right')
plt.show()
```

```
#show the valid and predicted value
valid
```

Sample Code of Deployment of ML Model on website

index.js

pages > index.js > ...

```
import React, { useEffect, useRef, useState } from 'react'
import Microsoft from '../Data/Microsoft.json'
import Amazon from '../Data/Amazon.json'
import Netflix from '../Data/Netflix.json'
import BAC from '../Data/BAC.json'
import Coca from '../Data/Coca.json'
import Fdex from '../Data/Fdex.json'
import Ford from '../Data/Ford.json'
import Gmotors from '../Data/Gmotors.json'
import Google from '../Data/Google.json'
import Ibm from '../Data/Ibm.json'
import Intel from '../Data/Intel.json'
import JP from '../Data/JP.json'
import Qualcom from '../Data/Qualcom.json'
import Tesla from '../Data/Tesla.json'
import Wallmart from '../Data/Wallmart.json'
import Lists from '../components/Lists'
import { BallTriangle } from 'react-loader-spinner'
import Head from 'next/head'

```

```
function index() {
  const [loading, setLoading] = useState(true)
  let options = useRef(null)
  let scrolls = useRef(null)
  const [set, setSet] = useState('')

  let count = 0

  function changes() {
    setLoading(true)
    setSet(options.current.value)
    setLoading(false)
    scrolls.current.scrollIntoView({
      behavior: 'smooth',
      block: 'start',
      inline: 'nearest',
    })
  }

  function changeData() {
    setLoading(true)
    setTimeout(changes, 3000)
  }

  function SetData() {
    setLoading(true)
    setSet('Microsoft')
    setLoading(false)
    count = 1
    scrolls.current.scrollIntoView({
      behavior: 'smooth',
      block: 'start',
      inline: 'nearest',
    })
  }
```

```
  if (count === 0) {
    useEffect(() => {
      setTimeout(SetData, 3000)
    }, [])
  }

  return (
    <>
      <Head>
        <title>Stock Price Predictor</title>
      </Head>
      <div className="m-auto  h-[100%] w-[100%] max-w-[80rem] border-0 border-white px-3 md:px-5    ">
        <img src="/title.png" alt="" className=" m-auto my-10  w-[90%] " />

```

```
71          <select
72            ref={options}
73            onChange={changeData}
74            className=" mt-5 rounded-sm   bg-transparent py-3 pr-5
75          >
76            <option value="Microsoft" className="bg-black">
77              Microsoft
78            </option>
79            <option value="Amazon" className="bg-black">
80              Amazon
81            </option>
82            <option value="Netflix" className="bg-black">
83              Netflix
84            </option>
85            <option value="BAC" className="bg-black">
86              Bank Of America
87            </option>
88            <option value="Coca" className="bg-black">
89              Coca Cola
90            </option>
91            <option value="Fdex" className="bg-black">
92              Fedex
93            </option>
94            <option value="Ford" className="bg-black">
95              Ford Motors
96            </option>
97            <option value="Gmotors" className="bg-black">
98              General Motors
99            </option>
100           <option value="Google" className="bg-black">
101             Google
102           </option>
103           <option value="Ibm" className="bg-black">
104             Ibm
105           </option>
106           <option value="Intel" className="bg-black">
107             Intel
```

```
            </option>
            <option value="JP" className="bg-black">
              JP Morgan Chase
            </option>
            <option value="Qualcom" className="bg-black">
              Qualcom
            </option>
            <option value="Tesla" className="bg-black">
              Tesla
            </option>
            <option value="Wallmart" className="bg-black">
              Wallmart
            </option>
          </select>

          <div className=" mt-10 w-[100%]  md:mt-10 ">
            {loading ? (
              <div className=" mt-36  flex h-fit w-[100%] items-center justify-center">
                <BallTriangle color="#00BFFF" height={80} width={80} />
              </div>
            ) : (
              <img
                src={`/${set}.png`}
                alt=""
                className="w-[100%] rounded-md shadow-md shadow-slate-300  "
              />
            )}
          </div>

        {!loading && (
          <div
            ref={scrolls}
            className=" my-14 flex w-[100%] flex-col items-center   md:my-32 "
          >
            {/*  */}

            <div className="flex w-[100%] flex-wrap  items-center  rounded-t-md bg-white shadow-sm shadow-slate-100 ">
              <p className=" w-[20%] border-[0.1rem] border-slate-200 py-4  px-4 text-left font-serif text-[1.3rem] font-semibold
                Index
              </p>
              <p className=" w-[40%] border-[0.1rem] border-slate-200 py-4  px-4 text-left font-serif text-[1.3rem] font-semibold
                Close
              </p>
              <p className=" w-[40%] border-[0.1rem] border-slate-200 py-4  px-4 text-left font-serif text-[1.3rem] font-semibold
                Prediction
              </p>
```

```
155                {set === 'Microsoft' &&
156                  Microsoft.map((data, index) => (
157                    <Lists
158                      key={index}
159                      index={data.index}
160                      close={data.Close}
161                      prediction={data.Predictions}
162                    />
163                  ))}
164
165                {set === 'Amazon' &&
166                  Amazon.map((data, index) => (
167                    <Lists
168                      key={index}
169                      index={data.index}
170                      close={data.Close}
171                      prediction={data.Predictions}
172                    />
173                  ))}
174                {set === 'Netflix' &&
175                  Netflix.map((data, index) => (
176                    <Lists
177                      key={index}
178                      index={data.index}
179                      close={data.Close}
180                      prediction={data.Predictions}
181                    />
182                  ))}
```

```
183                {set === 'BAC' &&
184                  BAC.map((data, index) => (
185                    <Lists
186                      key={index}
187                      index={data.index}
188                      close={data.Close}
189                      prediction={data.Predictions}
190                    />
191                  ))}
192
193                {set === 'Coca' &&
194                  Coca.map((data, index) => (
195                    <Lists
196                      key={index}
197                      index={data.index}
198                      close={data.Close}
199                      prediction={data.Predictions}
200                    />
201                  ))}
```

```
{set === 'Fdex' &&
  Fdex.map((data, index) => (
    <Lists
      key={index}
      index={data.index}
      close={data.Close}
      prediction={data.Predictions}
    />
  ))}

{set === 'Ford' &&
  Ford.map((data, index) => (
    <Lists
      key={index}
      index={data.index}
      close={data.Close}
      prediction={data.Predictions}
    />
  ))}

{set === 'Gmotors' &&
  Gmotors.map((data, index) => (
    <Lists
      key={index}
      index={data.index}
      close={data.Close}
      prediction={data.Predictions}
    />
  ))}

{set === 'Google' &&
  Google.map((data, index) => (
    <Lists
      key={index}
      index={data.index}
      close={data.Close}
      prediction={data.Predictions}
    />
  ))}
{set === 'Ibm' &&
  Ibm.map((data, index) => (
    <Lists
      key={index}
      index={data.index}
      close={data.Close}
      prediction={data.Predictions}
    />
  ))}
```

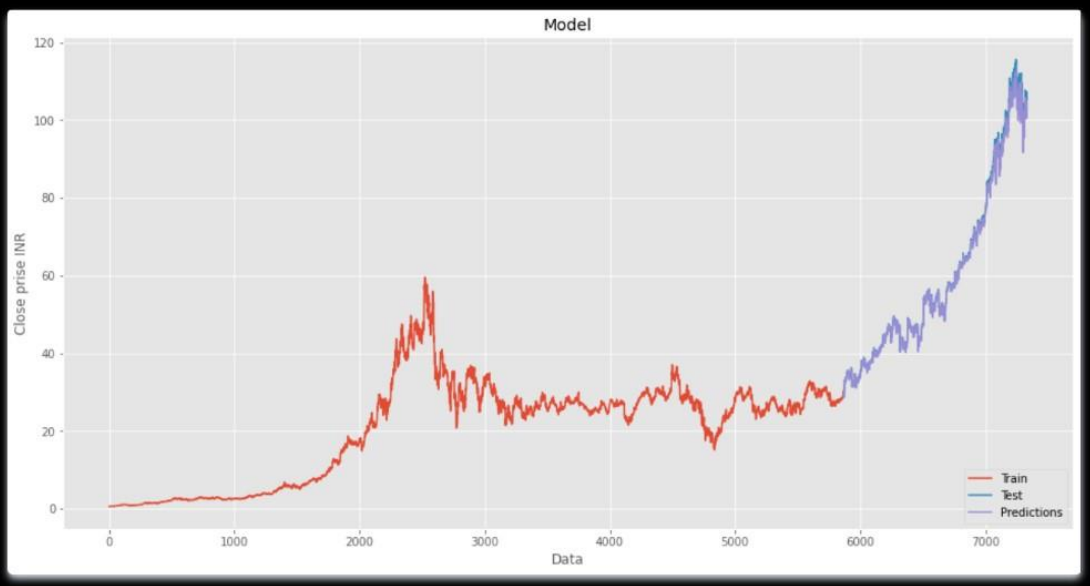
```
252                {set === 'Intel' &&
253                  Intel.map((data, index) => (
254                    <Lists
255                      key={index}
256                      index={data.index}
257                      close={data.Close}
258                      prediction={data.Predictions}
259                    />
260                  ))}
261
262                {set === 'JP' &&
263                  JP.map((data, index) => (
264                    <Lists
265                      key={index}
266                      index={data.index}
267                      close={data.Close}
268                      prediction={data.Predictions}
269                    />
270                  ))}
271
272                {set === 'Qualcom' &&
273                  Qualcom.map((data, index) => (
274                    <Lists
275                      key={index}
276                      index={data.index}
277                      close={data.Close}
278                      prediction={data.Predictions}
279                    />
280                  ))}
281
```

```
                {set === 'Tesla' &&
                  Tesla.map((data, index) => (
                    <Lists
                      key={index}
                      index={data.index}
                      close={data.Close}
                      prediction={data.Predictions}
                    />
                  ))}

                {set === 'Wallmart' &&
                  Wallmart.map((data, index) => (
                    <Lists
                      key={index}
                      index={data.index}
                      close={data.Close}
                      prediction={data.Predictions}
                    />
                  ))}
              </div>
            </div>
          )}

          {/*  */}
        </div>
      </>
    )
}

export default index

```

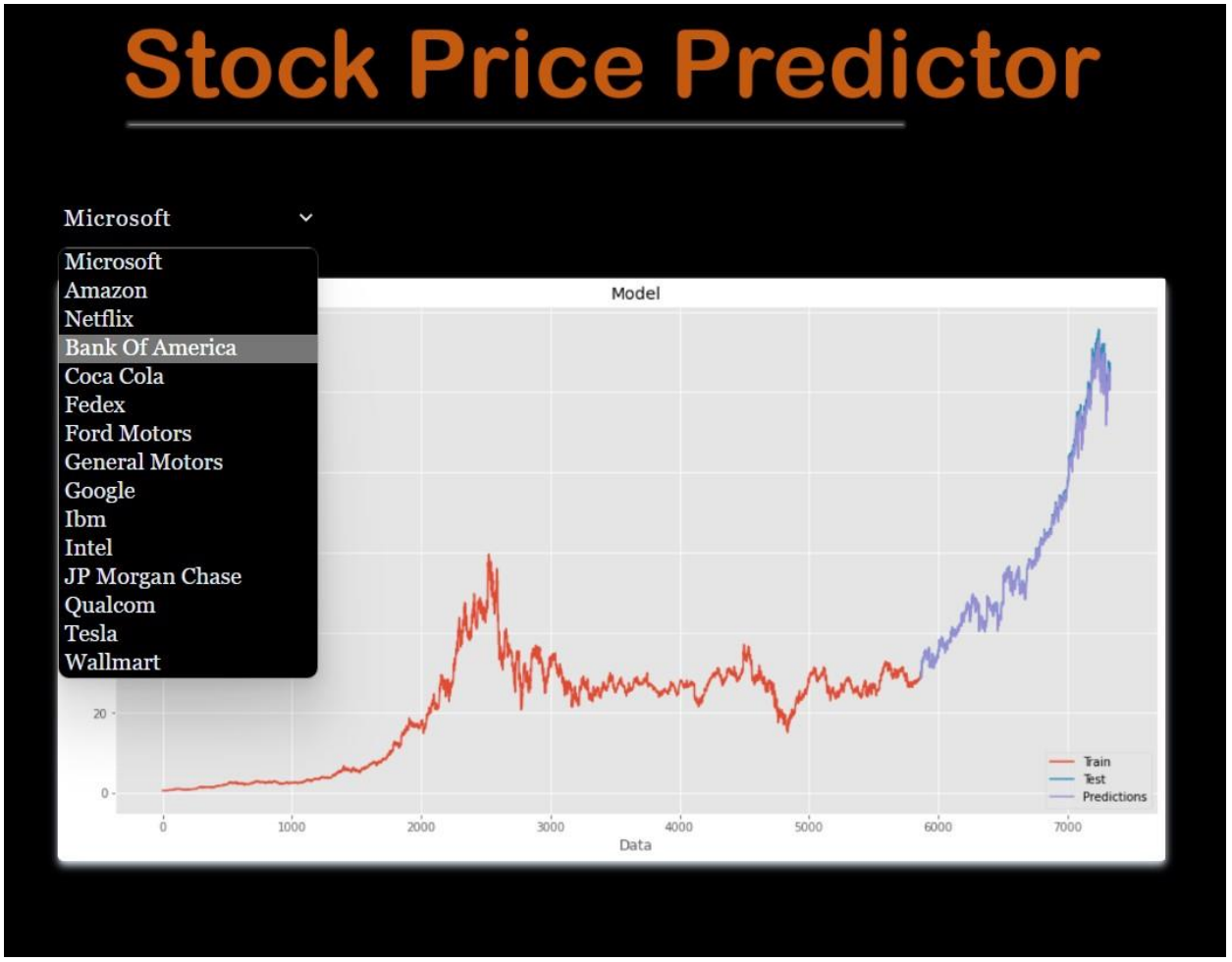
OUTPUT

Stock Price Predictor

Microsoft



Index	Close	Prediction
1	28.969999	28.62394142150879
2	28.83	28.897249221801758
3	28.790001	28.780990600585938
4	29.77	28.71715545654297
5	30.83	29.614383697509766
6	30.6	30.652509689331055
7	31.76	30.472152709960938
8	31.940001	31.528270721435547
9	31.790001	31.78553581237793
10	32.610001	31.662235260009766
11	33.099998	32.45459747314453
12	32.720001	32.99588394165039
13	33.16	32.66792297363281
14	33.490002	33.07095718383789



Sample Input and Output

1) Amazon

Attribute Name	Min	Max
High	1.447917	3773.080078
Low	1.312500	3696.790039
Open	1.406250	3744.000000
Close	1.395833	3731.409912

2) BAC

Attribute Name	Min	Max

High	36.910000	50.110001
Low	35.540001	49.029999
Open	36.270000	49.910000
Close	35.680000	49.380001

3) Coca Cola

Attribute Name	Min	Max
High	0.193359	60.130001
Low	0.182292	59.619999
Open	0.192708	59.810001
Close	0.192057	60.130001

4) F

Attribute Name	Min	Max
High	0.648804	37.300335
Low	0.638667	0.638667
Open	0.213001	36.819485
Close	0.638667	36.647751

5) FedX

Attribute Name	Min	Max
High	201.559998	319.899994
Low	196.149994	313.010010

Open	198.220001	318.269989
Close	198.740005	315.589996

6) GM

Attribute Name	Min	Max
High	65.519997	65.519997
Low	37.310001	62.689999
Open	38.290001	3744.000000
Close	37.910000	65.739998

7) Google

Attribute Name	Min	Max
High	50.920921	3030.929932
Low	48.028027	48.028027
Open	49.644646	3744.000000
Close	50.055054	2996.770020

8) IBM

Attribute Name	Min	Max
High	11.917	873.40078
Low	9.3500	12.790039
Open	10.6250	854.01400

Close	11.5833	870.4912

9) INTC

Attribute Name	Min	Max
High	99.245	2547.2148
Low	98.2410	2031.2445
Open	99.2897	2475.3246
Close	99.3471	2514.9874

10) JPM

Attribute Name	Min	Max
High	118.2415	1503.1596
Low	117.2587	1103.2458
Open	117.9872	1458.2873
Close	116.3256	1475.2369

11) Microsoft

Attribute Name	Min	Max
High	115.419998	116.180000
Low	116.180000	116.180000
Open	0.591146	115.419998
Close	0.598090	0.598090

12) Netflix

Attribute Name	Min	Max
High	0.374588	642.229980
Low	0.346429	630.859985
Open	0.598090	646.840027
Close	0.372857	0.372857

13) QCom

Attribute Name	Min	Max
High	125.470001	193.580002
Low	193.580002	185.190002
Open	125.199997	125.199997
Close	185.190002	189.279990

14) Tesla

Attribute Name	Min	Max
High	3.326000	900.400024
Low	2.996000	871.599976
Open	0.372857	891.380005
Close	3.160000	883.090027

15) Walmart

Attribute Name	Min	Max
High	57.000000	153.660004

Low	56.259998	151.660004
Open	56.389999	153.600006
Close	56.419998	152.789993

TENTATIVE CHAPTER PLAN FOR THE PROPOSED WORK

CHAPTER 1: INTRODUCTION

This chapter will cover the overview of the Stock Price Prediction Model.

CHAPTER 2: LITERATURE REVIEW

This chapter includes the literature available for the Stock Price Prediction Model. The findings of the researchers will be highlighted which will become the basis of the current implementation.

CHAPTER 3: PROBLEM FORMULATION

This chapter will cover the problem which make us to propose this model and aim of our project.

CHAPTER 4: OBJECTIVES

This chapter will cover what we are covering in this model.

CHAPTER 5: DESIGN

This chapter will cover all the diagrams related to project.

CHAPTER 6: METHODOLOGY

This chapter will cover the technical details of the proposed approach.

REFERENCES

REFERENCES

.

[1] F. a. o. Eugene, "Efficient capital markets: a review of theory and empirical work," Journal of finance, vol. 25, no. 2, pp. 383-417, 1970.

[2] Z. A. Farhath, B. Arputhamary and L. Arockiam, "A Survey on ARIMA Forecasting Using Time Series Model," Int. J. Comput. Sci. Mobile Comput, vol. 5, pp. 104-109, 2016.

[3] S. Wichaidit and S. Kittitornkun, "Predicting SET50 stock prices using CARIMA (cross correlation ARIMA)," in 2015 International Computer Science and Engineering Conference (ICSEC), IEEE, 2015, pp. 1-4.

[4] D. Mondal, G. Maji, T. Goto, N. C. Debnath and S. Sen, "A Data Warehouse Based Modelling Technique for Stock Market Analysis," International Journal of Engineering & Technology, vol. 3, no. 13, pp. 165-170, 2018.