STOCK PRICE PREDICTION

A PROJECT REPORT

Submitted by

Amarpreet Singh - 20BCS4438
Mohit Kumar Daga - 20BCS4416

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING - BIG DATA ANALYTICS

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Chandigarh University

NOVEMBER 2023

BONAFIDE CERTIFICATE

Certified that this project report "Stock Price Prediction" is the bonafide work of "Amarpreet Singh " and "Mohit Kumar Daga" who carried out their project work under our supervision.

Submitted for the project viva - voce examination held on:

SIGNATURE OF HEAD

SIGNATURE OF SUPERVISOR
Prof. Bhavna Nayyer
Professor
Computer Science Engineering

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide Prof. Bhavna Nayyer, sir Assistant Professor, Department of Computer Science and Engineering, CU, for his guidance with unsurpassed knowledge and immense encouragement.

We thank all the teaching faculty of Department of AIT - CSE, whose suggestions during reviews helped me in accomplishment of our project.

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

PROJECT STUDENTS

- AMARPREET SINGH - 20BCS4438
- MOHIT KUMAR DAGA - 20BCS4416

TABLE OF CONTENTS

List of Figures

ABSTRACT

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

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

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

List of Abbreviations

LSTM	Long Short-Term Memory
ML	Machine Learning
SVM	Support Vector Machine
AI	Artificial Intelligence
NN	Neural Networks
ARMA	Autoregressive Moving Average
LMS	Least Mean Square
UML	Unified modelling Language
MSE	Mean Squared Error
RMSE	Root Mean Squared Error

List of Symbols

Xt	Input at current state
X(t-1)	Input at Previous state
Ct	Current Cell State
C(t-1)	Previous Cell State
ht	Current hidden/output State
h(t-1)	Previous hidden/output State
σ	Sigmoid Function
tanh	Hyperbolic tangent function

CHAPTER 1
INTRODUTION

The financial market is a dynamic and multifaceted system where participants engage in buying and selling various financial instruments, including currencies, stocks, equities, and derivatives. This activity occurs on virtual platforms facilitated by brokers, providing investors with the means to trade on exchanges or over-the-counter markets. The stock market, in particular, serves as a conduit for investors to acquire ownership in public companies through the trading of shares. This market has democratized wealth-building, affording individuals the opportunity to prosper by investing modest initial amounts with relatively lower risk compared to the uncertainties associated with starting a new business or pursuing a high-salary career.

The allure of the stock market lies in its potential for financial gain and the promise of a prosperous life through strategic investments. However, this realm is not without its challenges. Stock markets are susceptible to myriad factors that introduce uncertainty and contribute to high volatility. The complexity of these markets necessitates a nuanced understanding of the forces at play, ranging from economic indicators and market sentiment to geopolitical events. While human traders have traditionally executed orders and navigated the market terrain, the emergence of automated trading systems (ATS) has introduced a paradigm shift.

Automated trading systems, operated by computer programs, have demonstrated superior efficiency and momentum in executing orders compared to their human counterparts. These systems leverage algorithms and computational power to swiftly analyze market conditions and execute trades with precision. Despite their advantages, evaluating and controlling the performance of ATSs requires the implementation of risk management strategies and safety measures, often guided by human judgment. Striking a balance between the autonomy of automated systems and the need for human oversight is crucial to navigate the intricate dynamics of financial markets successfully.

The development of an ATS involves careful consideration of various factors. Key components include defining the trading strategy to be adopted, incorporating complex mathematical functions that reflect the state of specific stocks, integrating machine learning algorithms for predicting future stock values, and monitoring relevant news affecting the stocks being analyzed. The amalgamation of these elements aims to create a robust automated system that can adapt to changing market conditions and optimize trading outcomes.

One notable challenge in the financial market is the inherent unpredictability, and as such, time-series prediction has emerged as a valuable technique for forecasting future market trends. This method leverages continuous data over a specific time period to predict outcomes in subsequent time units. Time-series prediction has found wide application in real-world scenarios, including weather forecasting and financial market prediction. Various algorithms have demonstrated effectiveness in this domain, with Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) standing out as common and powerful tools.

The stock market, characterized by its dynamic and time-dependent nature, presents an ideal candidate for time-series prediction models. Researchers and practitioners alike have delved into the development of models that can anticipate stock price movements, taking into account the historical patterns and trends exhibited by financial instruments. In this project, the focus is on the LSTM model, a variant of RNN designed to capture long-term dependencies in sequential data.

LSTM's architecture enables it to retain and learn from information over extended periods, making it well-suited for modeling the intricate dynamics of stock prices. The model incorporates memory cells that can selectively retain or forget information, allowing it to capture and remember patterns that are essential for accurate predictions. By leveraging the power of LSTM, this project aims to enhance the accuracy of stock price predictions and contribute to the broader field of financial forecasting.

The process involves training the LSTM model on historical stock price data, allowing it to learn the underlying patterns and relationships. The training phase involves exposing the model to a substantial amount of past data, enabling it to discern subtle nuances and trends. Once trained, the LSTM model can then be used to make predictions on future stock prices, providing valuable insights for investors and financial analysts.

In conclusion, the intersection of automated trading systems and time-series prediction models, particularly the LSTM, represents a compelling avenue for enhancing the efficiency and accuracy of stock market predictions. The dynamic nature of financial markets, coupled with the vast amount of data available, necessitates sophisticated approaches to decision-making. By harnessing the capabilities of ATS and leveraging advanced predictive models, stakeholders in the financial realm can gain a competitive edge in navigating the complexities of stock markets, ultimately leading to more informed and strategic investment decisions.

1.1 MOTIVATION FOR WORK

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

1.2 PROBLEM STATEMENT

Time Series forecasting and modeling constitute a pivotal facet of data analysis, finding specialized applications in disciplines such as Econometrics and Operation Research. In the realm of analytics and data science, Time Series analysis is pervasive, with its utility extending to diverse fields. One particularly compelling application is the prediction of stock prices, an area where volatility is inherent, and prices are contingent on a myriad of influencing factors.

At the core of this project lies the utilization of Long Short-Term Memory (LSTM), an advanced recurrent neural network architecture. LSTM is particularly well-suited for tasks involving sequential data, making it an ideal candidate for modeling the temporal dependencies inherent in time series data, such as stock prices. Its capacity to retain and recall information over extended periods enables it to capture nuanced patterns and trends in the historical stock price data, contributing to more accurate predictions.

The significance of this endeavor is underscored by the dynamic nature of stock markets, where informed decision-making relies on the ability to anticipate future price movements. By leveraging LSTM, the project seeks to enhance the precision of stock price predictions, providing valuable insights to stakeholders in the financial domain. The model's capability to discern intricate patterns in historical stock data positions it as a potent tool for navigating the complexities of financial markets.

As stock prices are influenced by a plethora of variables, ranging from economic indicators to market sentiment, the LSTM model's ability to discern and adapt to changing patterns becomes crucial. Through a rigorous analysis of historical stock data and the training of the LSTM model, the project endeavors to contribute to the broader landscape of financial forecasting, with potential implications for investment strategies and risk management. Ultimately, the aim is to harness the power of LSTM to unravel the intricate dynamics of stock prices, facilitating more informed decision-making in the ever-evolving financial landscape.

CHAPTER 2
LITERATURE SURVEY

2.1 INTRODUCTION

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

2.2 EXISTING METHODS

2.2.1 Stock Market Prediction Using Machine Learning

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

2.2.2 Forecasting the Stock Market Index Using Artificial Intelligence Techniques

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

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

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

The research work done by Dharmaraja Selvamuthu, Vineet Kumar and Abhishek Mishra Department of Mathematics, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India. A stock market is a platform for trading of a company's stocks and derivatives at an agreed price. Supply and demand of shares drive the stock market. In any country stock market is one of the most emerging sectors. Nowadays, many people are indirectly or directly related to this sector. Therefore, it becomes essential to know about market trends. Thus, with the development of the stock market, people are interested in forecasting stock price. But, due to dynamic nature and liable to quick changes in stock price, prediction of the stock price becomes a challenging task. Stock m Prior work has proposed effective methods to learn event representations that can capture syntactic and semantic information over text corpus, demonstrating their effectiveness for downstream tasks such as script event prediction. On the other hand, events extracted from raw texts lacks of common-sense knowledge, such as the intents and emotions of the event participants, which are useful for distinguishing event pairs when there are only subtle differences in their surface realizations. To address this issue, this paper proposes to leverage external common-sense knowledge about the intent and sentiment of the event.

Experiments on three event-related tasks, i.e., event similarity, script event prediction and stock market prediction, show that our model obtains much better event embeddings for the tasks, achieving 78% improvements on hard similarity task, yielding more precise inferences on subsequent events under given contexts, and better accuracies in predicting the volatilities of the stock market1. Markets are mostly a non-parametric, non-linear, noisy and deterministic chaotic system (Ahangar et al. 2010). As the technology is increasing, stock traders are moving towards to use Intelligent Trading Systems rather than fundamental analysis for predicting prices of stocks, which helps them to take immediate investment decisions. One of the main aims of a trader is to predict the stock price such that he can sell it before its value decline, or buy the stock before the price rises. The efficient market hypothesis states that it is not possible to predict stock prices and that stock behaves in the random walk. It seems to be very difficult to replace the professionalism of an experienced trader for predicting the stock price. But because of the availability of a remarkable amount of data and technological advancements we can now formulate an appropriate algorithm for prediction whose results can increase the profits for traders or investment firms. Thus, the accuracy of an algorithm is directly proportional to gains made by using the algorithm.

2.2.4 The Stock Market and Investment

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

2.2.5 Automated Stock Price Prediction Using Machine Learning

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

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

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

2.2.7 Event Representation Learning Enhanced with External Common-sense Knowledge

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

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

The research work done by Pushpendu Ghosh, Ariel Neufeld, Jajati Keshari SahooDepartment of Computer Science & Information Systems, BITS Pilani K.K.Birla Goa campus, India bDivision of Mathematical Sciences, Nanyang Technological University, Singapore cDepartment of Mathematics, BITS Pilani K.K.Birla Goa campus, India. We employ both random forests and LSTM networks (more precisely CuDNNLSTM) as training methodologies to analyse their effectiveness in forecasting out-of-sample directional movements of constituent stocks of the S&P 500 from January 1993 till December 2018 for intraday trading. We introduce a multi-feature setting consisting not only of the returns with respect to the closing prices, but also with respect to the opening prices and intraday returns. As trading strategy, we use Krauss et al. (2017) and Fischer & Krauss (2018) as benchmark and, on each trading day, buy the 10 stocks with the highest probability and sell short the 10 stocks with the lowest probability to outperform the market in terms of intraday returns – all with equal monetary weight. Our empirical results show that the multi-feature setting provides a daily return, prior to transaction costs, of 0.64% using LSTM networks, and 0.54% using random forests. Hence, we outperform the single-feature setting in Fischer & Krauss (2018) and Krauss et al. (2017) consisting only of the daily returns with respect to the closing prices, having corresponding daily returns of 0 .41% and of 0 .39% with respect to LSTM and random forests, respectively. 1 Keywords: Random forest, LSTM, Forecasting, Statistical Arbitrage, Machine learning, Intraday trading.

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

The research work done by Xiao-Yang 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, Imperial College, 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 that decides where to trade, at what price, and what quantity involves error-prone and arduous development and debugging. In this paper, we introduce a DRL library FinRL that facilitates beginners to expose themselves to quantitative finance and to develop their own stock trading strategies. Along with easily-reproducible tutorials, FinRL library allows users to streamline their own developments and to compare with existing schemes easily.

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

2.2.10 An innovative neural network approach for stock market prediction

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

Based on the development of word vector in deep learning, we demonstrate the concept of “stock vector.” The input is no longer a single index or single stock index, but multi-stock high-dimensional historical data. We propose the deep long short-term memory neural network (LSTM) with embedded layer and the long short-term memory neural network with automatic encoder to predict the stock market. In these two models, we use the embedded layer and the automatic encoder, respectively, to vectorize the data, in a bid to forecast the stock via long short-term memory neural network. The experimental results show that the deep LSTM with embedded layer is better. Specifically, the accuracy of two models is 57.2 and 56.9%, respectively, for the Shanghai A-shares composite index. Furthermore, they are 52.4 and 52.5%, respectively, for individual stocks. We demonstrate research contributions in IMMT for neural network-based financial analysis. 2.2.11 An Intelligent Technique for Stock Market Prediction

2.2.11 An Intelligent Technique for Stock Market Prediction

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

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

CHAPTER 3
METHODOLOGY

3.1 PROPOSED SYSTEMS

Our proposed system is a new approach to predicting stock prices. We're looking to predict the future price of the stock. We want to be able to tell you what's happening right now in the market and how it will affect your investment 30 days later.

The way our system works is by analyzing past data about the stock price of a company and making predictions based on that data. The more data we have, the better our predictions will be.

Our system will use machine learning techniques (differential equations and convex optimization) to understand what drives a company's stock price. This allows us to make predictions based on current events, which helps us create an accurate model of how people behave when they invest money into a company's stocks.

The proposed stock price prediction project aims to create a system that can predict the price of a stock before it is released to the public. The system will be developed using a model called as 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:

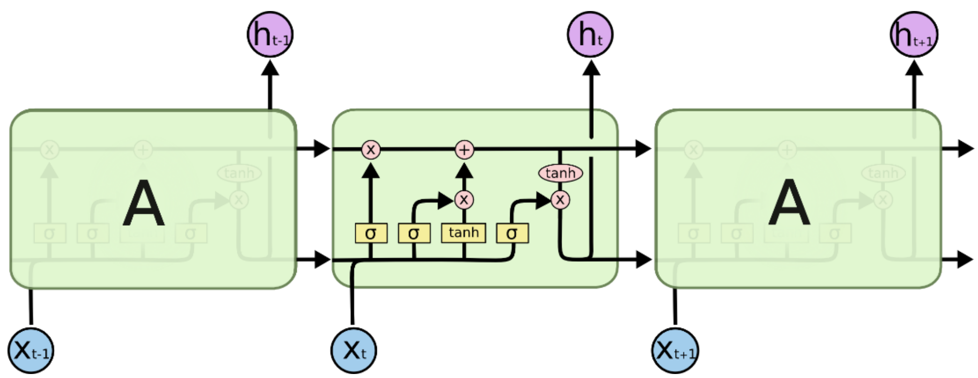


Fig. 4: LSTM Architecture

Forget Gate:

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

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

Input Gate:

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

Output Gate:

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

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

```
• # LSTM
• Inputs: dataset
• Outputs: RMSE of the forecasted data
•
• # Split dataset into 70% training and 30% testing data
• size = length(dataset) * 0.70
• 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, activation='relu', return_sequences=True,
•     input_shape = (time_step, n_features)))
•     model.add(LSTM(50, return_sequences = True, activation='relu'))
•     model.add(LSTM(50))
•     model.add(Dense(1))
•     model.compile(loss='mean_squared_error', optimizer='adam')
•     model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=50,
      batch_size=64, verbose = 1)
•     return model
•
• # Procedure to make predictions
• Procedure getPredictonsFromModel (model, X)
•     predictions = model.predict(X)
• # Fit the LSTM model
• model = LSTMAlgorithm (train, epoch, neurons)
•
• # Make predictions
• pred = model.predict(train)
•
• # Validate the modeln
• = len(dataset)
•
• error = 0
• for i in range(n): error += (abs(real[i] - pred[i])/real[i]) * 100
• accuracy = 100 - error/n
```

Hardware Requirements:

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

Software Requirements:

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

Functional requirements

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

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

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

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

Non-Functional requirements

Product properties

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

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

CHAPTER 4
DESIGN FLOW/PROCESS

4.1 Structure Chart

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

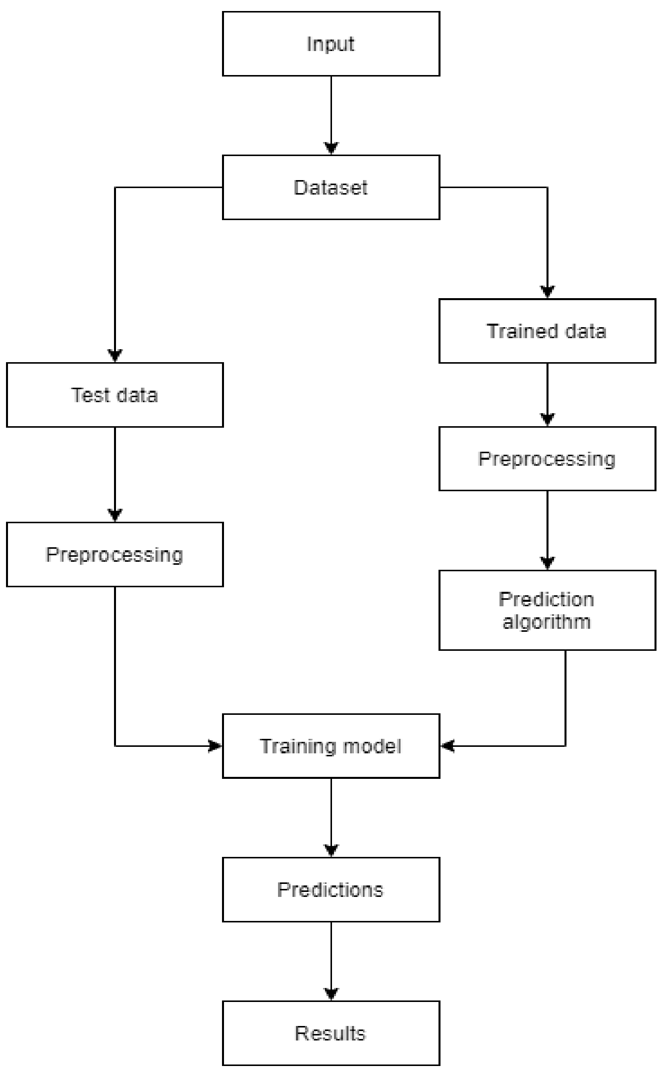


Fig. 7: Training and prediction

4.2 UML Diagrams

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

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

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

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

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

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

4.2.1 Sequence Diagram

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

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

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

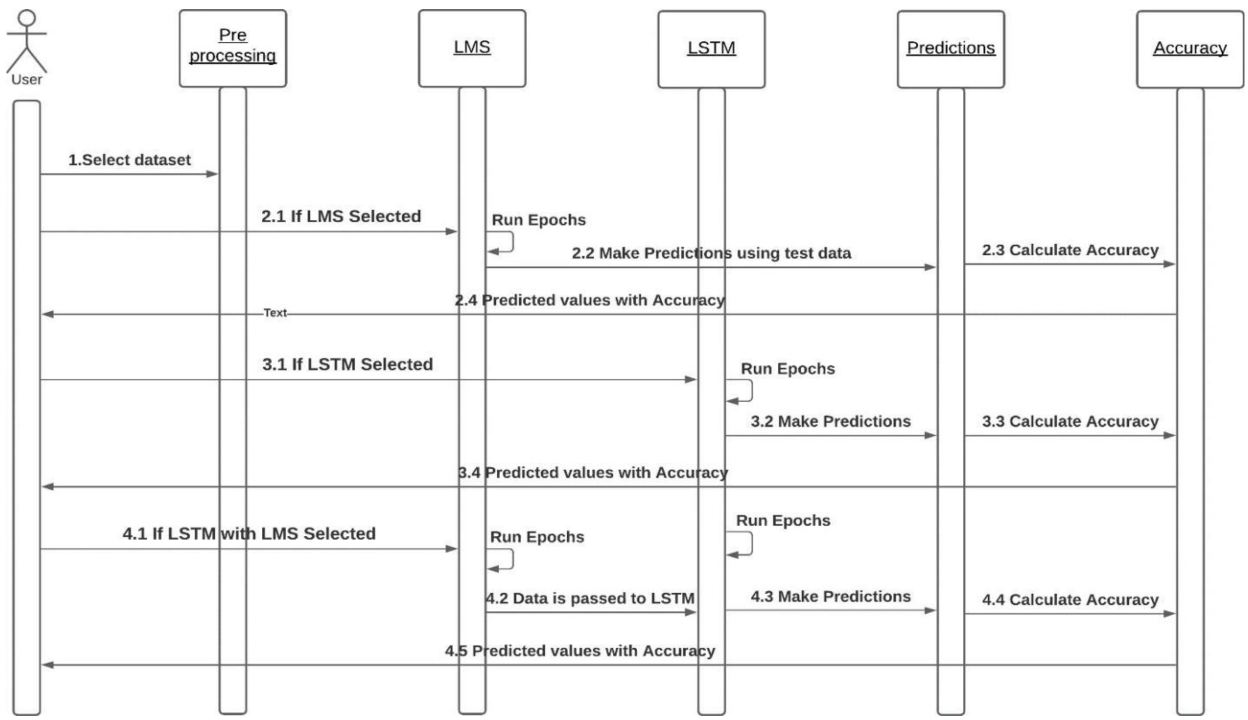


Fig. 9: Execution based on model selection

4.2.2 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.

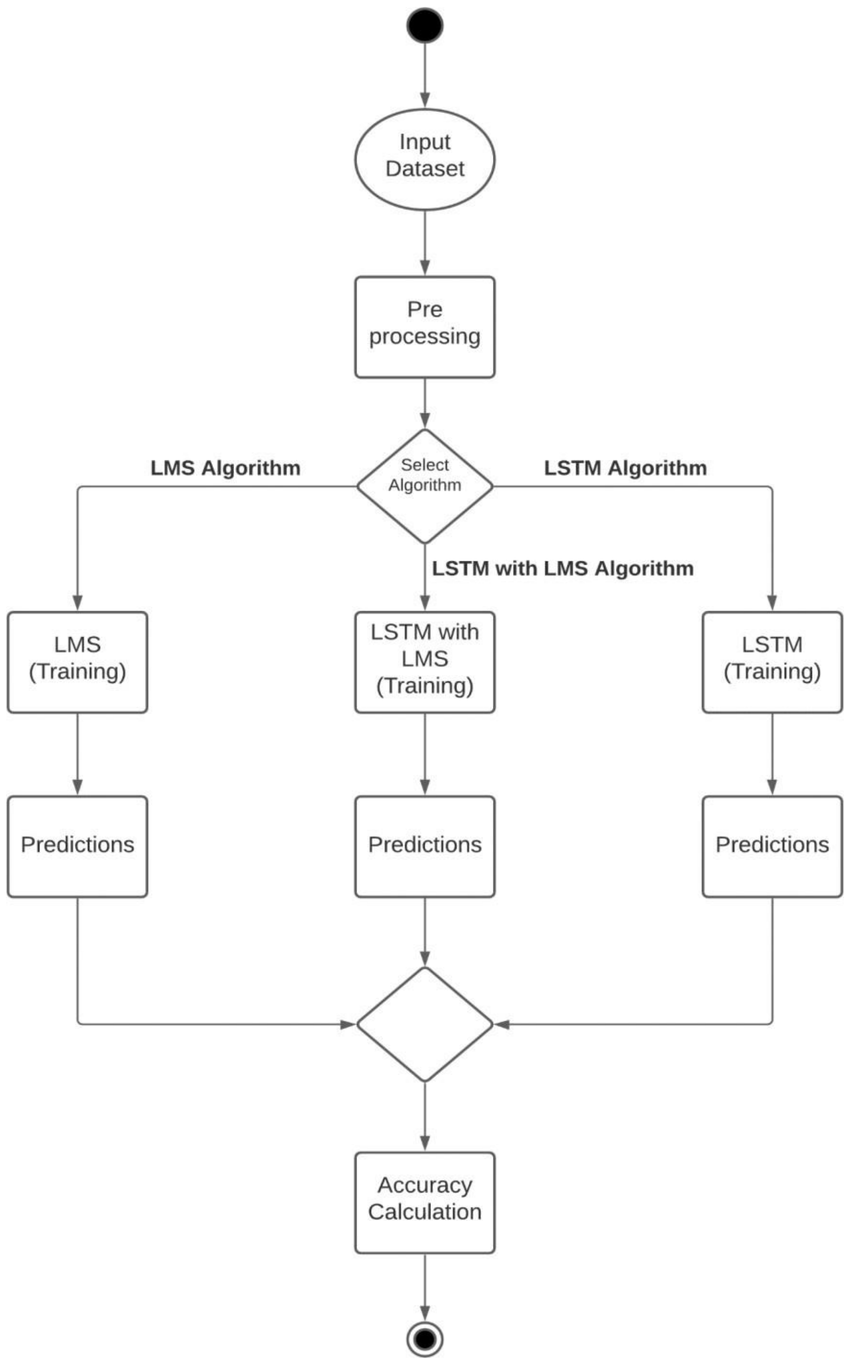


Fig. 10: Execution based on algorithm selection

4.2.3 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.

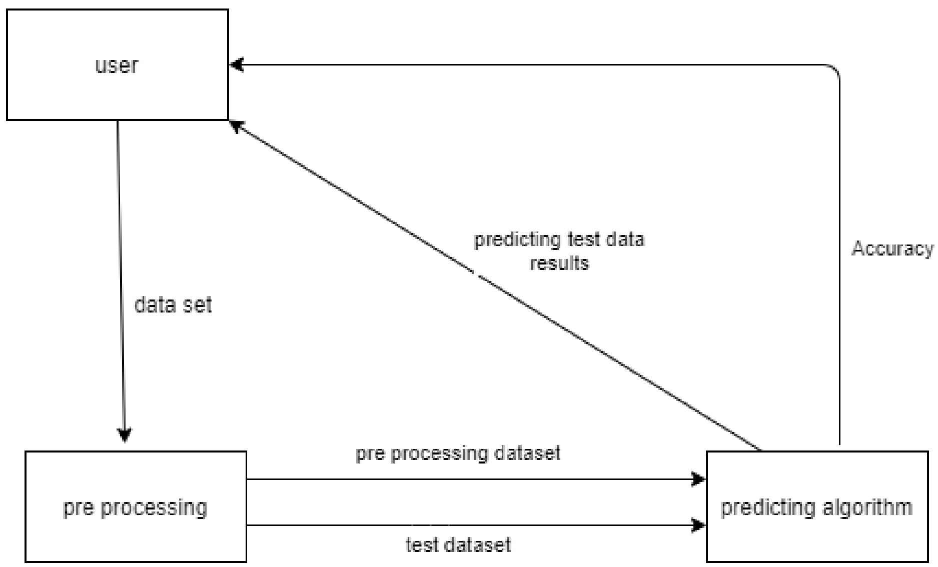


Fig. 11: Data transfer between modules

4.2.4 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, a

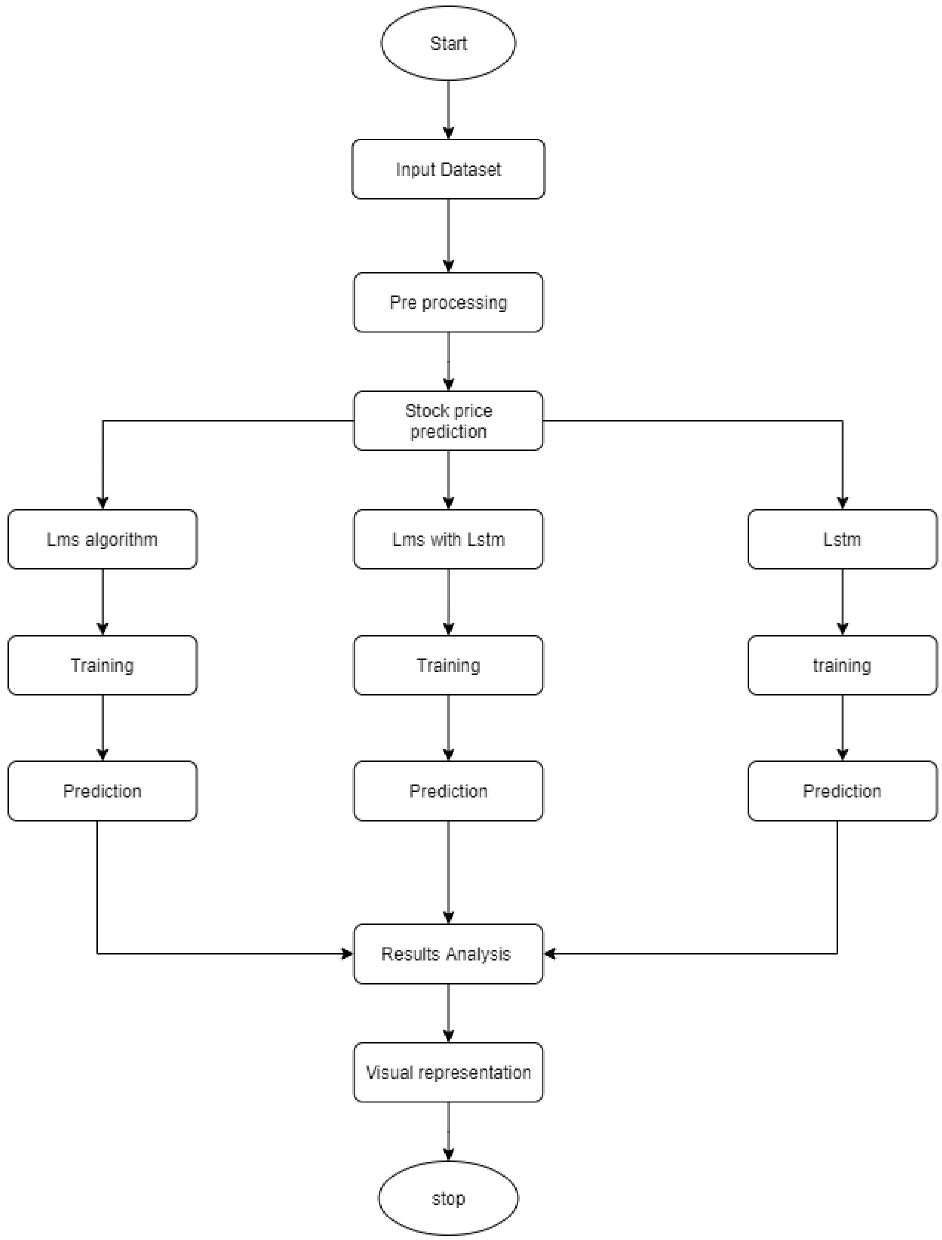


Fig. 12: Flow of execution

CHAPTER 5
EXPERIMENT ANALYSIS

5.1 System Configuration

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

5.1.1 Hardware Requirements:

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

5.1.2 Software requirements

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

5.2 Sample code

Stock Market Prediction

```
import math

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

1.1 Loading dataset

```
tsla_df = pd.read_csv('tsla.csv')
```

2.0 Exploratory Data Analysis

2.1 First 5 rows

```
tsla_df.head(5)
```

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2020-11-05 00:00:00-05:00	142.766663	146.666672	141.333328	146.029999	85243500	0	0.0
1	2020-11-06 00:00:00-05:00	145.366669	145.523331	141.426666	143.316666	65118000	0	0.0
2	2020-11-09 00:00:00-05:00	146.500000	150.833328	140.333328	140.419998	104499000	0	0.0
3	2020-11-10 00:00:00-05:00	140.029999	140.029999	132.009995	136.786667	90852600	0	0.0
4	2020-11-11 00:00:00-05:00	138.816666	139.566666	136.860001	139.043335	52073100	0	0.0

2.2 Dropping Unnecessary features

```
tsla_df = tsla_df.drop(['Dividends', 'Stock Splits'], axis=1)
tsla_df

# shape of the dataset = 504-rows x 6-cols
```

	Date	Open	High	Low	Close	Volume
0	2020-11-05 00:00:00-05:00	142.766663	146.666672	141.333328	146.029999	85243500
1	2020-11-06 00:00:00-05:00	145.366669	145.523331	141.426666	143.316666	65118000
2	2020-11-09 00:00:00-05:00	146.500000	150.833328	140.333328	140.419998	104499000
3	2020-11-10 00:00:00-05:00	140.029999	140.029999	132.009995	136.786667	90852600
4	2020-11-11 00:00:00-05:00	138.816666	139.566666	136.860001	139.043335	52073100
...	...	...	...	...	...	...
499	2022-10-31 00:00:00-04:00	226.190002	229.850006	221.940002	227.539993	61554300
500	2022-11-01 00:00:00-04:00	234.050003	237.399994	227.279999	227.820007	62688800
501	2022-11-02 00:00:00-04:00	226.039993	227.869995	214.820007	214.979996	63070300
502	2022-11-03 00:00:00-04:00	211.360001	221.199997	210.139999	215.309998	56538800
503	2022-11-04 00:00:00-04:00	222.600006	223.800003	203.080002	207.470001	98453100

504 rows × 6 columns

2.3 Checking the count of null / NA values in each feature

```
tsla_df.isnull().sum()
```

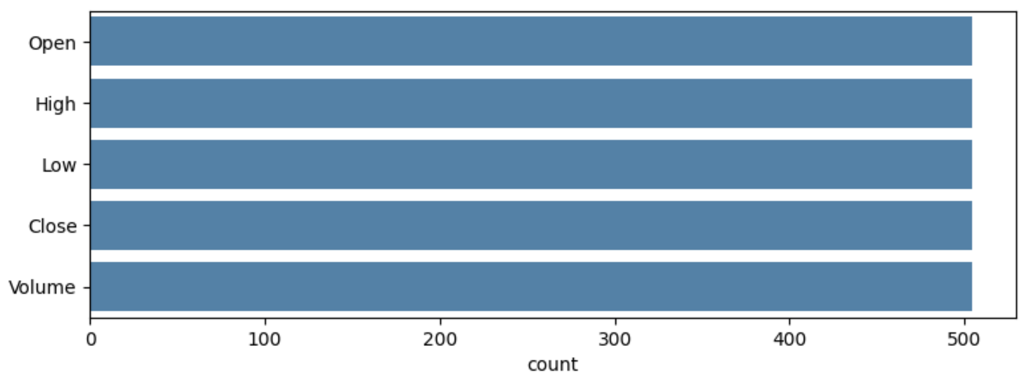
```
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

2.4 Visualizing if it has null / NA values

```
plt.figure(figsize=(9, 3))

sns.countplot(data = tsla_df.iloc[0:], orient='h', color = 'steelblue')          # countplot does not include 'nan' values
```

```
C:\Users\user\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categorical.py:470: FutureWarning: iteritems is deprecated and will be removed in a future version. Use .items instead.
  plot_data = [np.asarray(s, float) for k, s in iter_data]
```

```
<AxesSubplot: xlabel='count'>
```



2.5 Basic Info of features

```
tsla_df.info()                                        # python treats date and datetime as object
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 504 entries, 0 to 503
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    504 non-null    object
 1   Open    504 non-null    float64
 2   High    504 non-null    float64
 3   Low     504 non-null    float64
 4   Close   504 non-null    float64
 5   Volume  504 non-null    int64
```

2.6 Statistical Analysis

```
tsla_df.describe().apply(lambda s: s.apply('{:.3f}'.format)).T          # or s.apply('{:.5f}'.format)
```

	count	mean	std	min	25%	50%	75%	max
Open	504.000	263.247	53.893	136.310	223.982	251.040	298.625	411.470
High	504.000	269.214	55.158	137.483	229.290	254.903	303.934	414.497
Low	504.000	256.662	52.211	132.010	218.187	243.980	290.100	405.667
Close	504.000	263.029	53.557	136.030	223.654	251.093	296.762	409.970
Volume	504.000	86568662.302	43105164.371	29401800.000	62005000.000	79020000.000	98997225.000	666378600.000

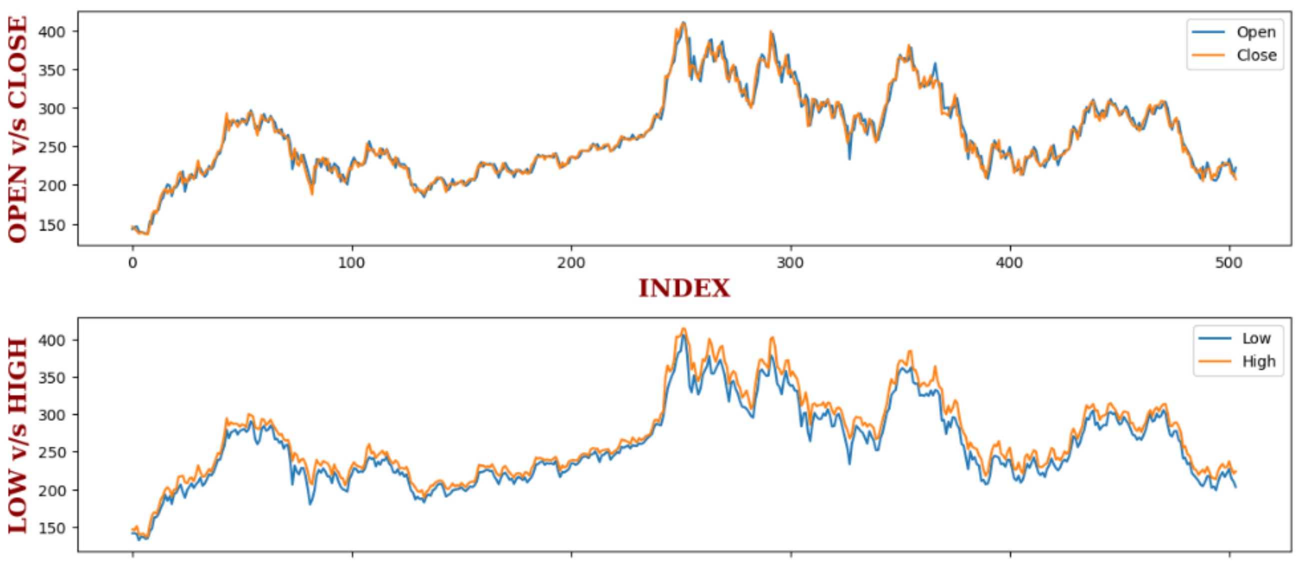
```
tsla_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 504 entries, 0 to 503
Data columns (total 6 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Date     504 non-null    object
 1   Open     504 non-null    float64
 2   High     504 non-null    float64
 3   Low      504 non-null    float64
 4   Close    504 non-null    float64
 5   Volume   504 non-null    int64
dtypes: float64(4), int64(1), object(1)
memory usage: 23.8+ KB
```

2.7 Stock Close as data series

```
df_close = tsla_df['Close']
df_close
```

```
font = {'family': 'serif',
        'color':  'darkred',
        'weight': 'bold',
        'size': 16,
        }

plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 1)
plt.plot(tsla_df['Open'], label='Open')
plt.plot(tsla_df['Close'], label='Close')
plt.xlabel('INDEX', fontdict=font)
plt.ylabel('OPEN v/s CLOSE', fontdict=font)
plt.legend()                                              # legend() : it is an area describing the elements of the graph

plt.subplot(3, 1, 2)
plt.plot(tsla_df['Low'], label='Low')
plt.plot(tsla_df['High'], label='High')
plt.xlabel('INDEX', fontdict=font)
plt.ylabel("LOW v/s HIGH", fontdict=font)
plt.legend()

plt.tight_layout()                                        # tight_layout() : automatically adjusts the subplots in the area
```



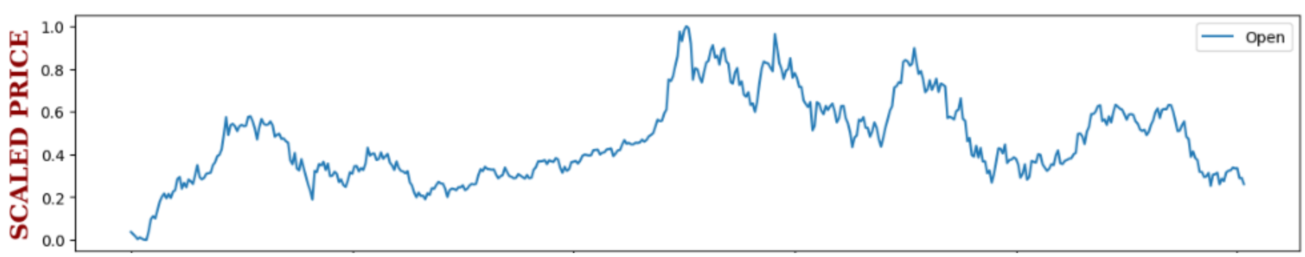
3.1 MIN_MAX scaler

```
# LSTM are sensitive to the scale of the data. MIN_MAX scaler will help to convert that data b/w 0 to 1.
# Min value = 0, Max val = 1
# X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
# eg : (224 - 204) / (409 - 204)  ==  0.0975

scaler = MinMaxScaler(feature_range=(0,1))

# df_close is of type pandas.series i.e, 1-D of shape (504, )
# but scaling object accepts 2-D frame.
# so used np.array to change it into 2-D frame

df_close = scaler.fit_transform(np.array(df_close).reshape(-1, 1))
df_close.shape

# now the shape of df_close is (504, 1)
```

```
(504, 1)
```

```
print(type(df_close))
font = {'family': 'serif',
        'color':  'darkred',
        'weight': 'bold',
        'size': 16,
        }

plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 1)
plt.plot(df_close, label='Open')
plt.xlabel('INDEX', fontdict=font)
plt.ylabel('SCALED PRICE', fontdict=font)
plt.legend()
plt.tight_layout()
```

```
<class 'numpy.ndarray'>
```



3.2 Train - Test split, size and data

```
train_size = int(len(df_close)*0.70)                        # train size = 70%
test_size = len(df_close) - train_size                      # test size = 30%

train_data, test_data = df_close[0:train_size, :], df_close[train_size:len(df_close), :]
```

```
train_size, test_size
```

```
(352, 152)
```

3.3 Dividing data

```
# Whenever we will have time series data, the next data is always dependent on the previous data

def create_dataset(dataset, time_step):
    dataX, dataY = [], []
    for i in range(len(dataset) - time_step - 1):
        a = dataset[i:(i+time_step), 0]
        dataX.append(a)
        dataY.append(dataset[(i+time_step), 0])

    return np.array(dataX), np.array(dataY)
```

```
time_step = 100
X_train, y_train = create_dataset(train_data, time_step)
X_test, y_test = create_dataset(test_data, time_step)
```

```
X_train.shape, y_train.shape
```

```
((251, 100), (251,))
```

3.4 Reshapping the dataset

```
n_features = 1
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], n_features)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], n_features)
```

4.1 Model Definition

```
# define Stacked LSTM model
# Models that input or output data sequences are called Sequential Model, eg: text stream, audio, video, time-series data etc.

model = Sequential()
model.add(LSTM(50, activation='relu', return_sequences=True, input_shape = (time_step, n_features)))
# added one LSTM model of 50 neurons
# Return sequence = True, the output of the hidden state of each neuron is used as an input to the next LSTM layer

model.add(LSTM(50, return_sequences = True, activation='relu'))
model.add(LSTM(50))
model.add(Dense(1))
# Dense layer is added to get output in format needed by the user. It is fully connected layer at the end of neural network

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

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 lstm_1 (LSTM)               (None, 100, 50)           20200

 lstm_2 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

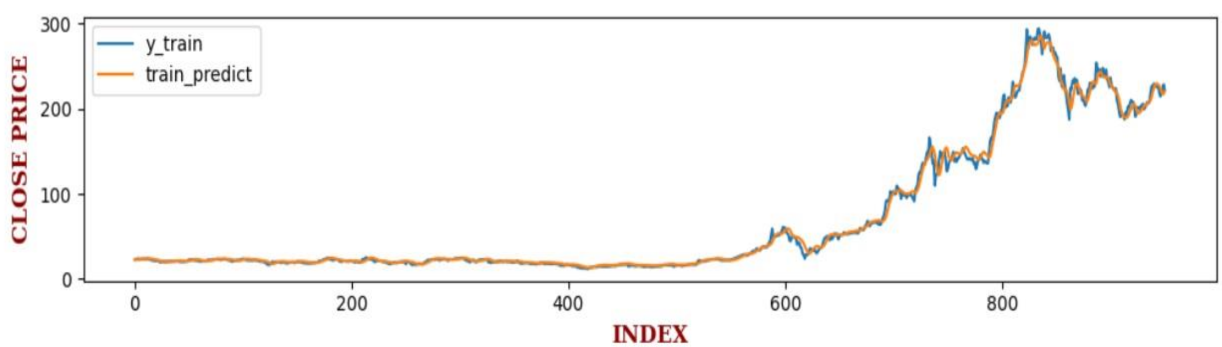
4.2 Fitting data to the model

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100, batch_size=64, verbose = 1)
```

4.5 Plot : Actual v/s Predicted - Train

In [29]:
```
font = {'family': 'serif',
        'color':  'darkred',
        'weight': 'bold',
        'size': 12,
        }

plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 1)
plt.plot(y_train, label='y_train')
plt.plot(train_predict, label='train_predict')
plt.xlabel("INDEX", fontdict=font)
plt.ylabel("CLOSE PRICE", fontdict=font)
plt.legend()
```

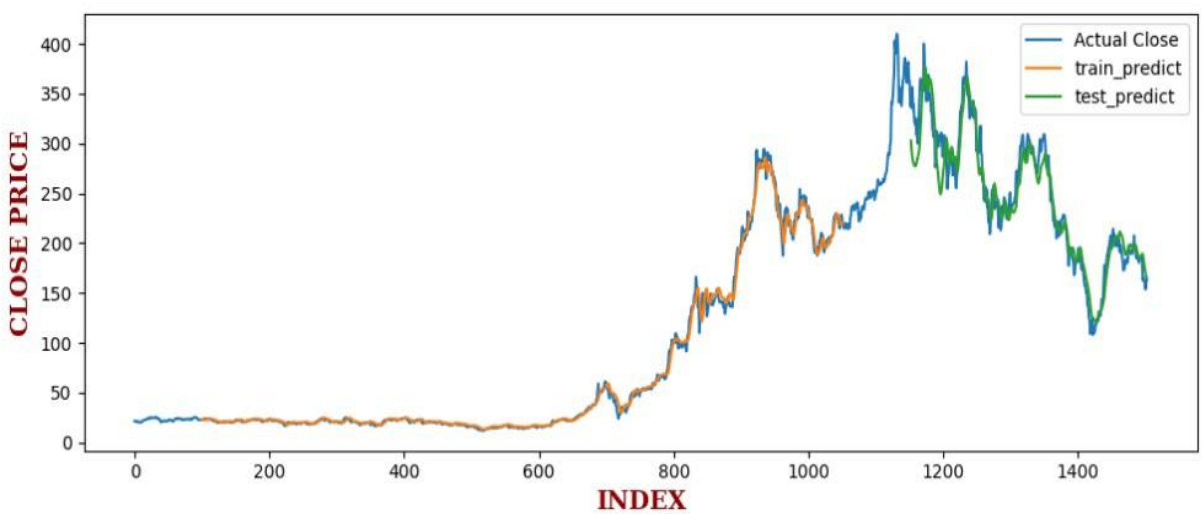
Out[29]: <matplotlib.legend.Legend at 0x1fb2711faf0>



4.8 Plot : Actual vs Predicted : Complete DataSet

In [32]:
```
# first 50 will be vacant
trainPredictPlot = np.empty_like(df_close)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[time_step:len(train_predict)+time_step, : ] = train_predict

testPredictPlot = np.empty_like(df_close)
testPredictPlot[:, :] = np.nan
testPredictPlot[len(train_predict)+2*time_step+1:len(df_close)-1, :] = test_predict
```

In [33]:
```
font = {'family': 'serif',
        'color':  'darkred',
        'weight': 'bold',
        'size': 14,
        }

plt.figure(figsize=(10, 4))
plt.subplot(1, 1, 1)
plt.plot(scaler.inverse_transform(df_close), label='Actual Close')
plt.plot(trainPredictPlot, label='train_predict')
plt.plot(testPredictPlot, label='test_predict')
plt.xlabel('INDEX', fontdict=font)
plt.ylabel('CLOSE PRICE', fontdict=font)
plt.legend()                                        # legend() : it is an area describing the elements of the graph

plt.tight_layout()                                  # tight_layout() : automatically adjusts the subplots in the area
```



5.0 Next 30 - Day Prediction

```
len(test_data)
```

```
152
```

```
x_input = test_data[52: ].reshape(1, -1)
print(type(x_input))
print(x_input.shape)
```

```
<class 'numpy.ndarray'>
(1, 100)
```

```
temp_input = list(x_input)
temp_input = temp_input[0].tolist()
print(len(temp_input))
print(type(temp_input))
```

```
100
<class 'list'>
```

```
# predicting the next 30 days

lst_output = []
i=0

while(i<30):
    if(len(temp_input)>100):
        x_input = np.array(temp_input[1:])
        x_input = x_input.reshape(1, -1)
        x_input = x_input.reshape((1, time_step, 1))
        y_input = model.predict(x_input, verbose=0)
        temp_input.extend(y_input[0].tolist())
        temp_input = temp_input[1:]
        lst_output.extend(y_input[0].tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, time_step, 1))
        y_input = model.predict(x_input)
        temp_input.extend(y_input[0].tolist())
        lst_output.extend(y_input[0].tolist())
        i=i+1
```

5.2 Plot : Next 30 Day

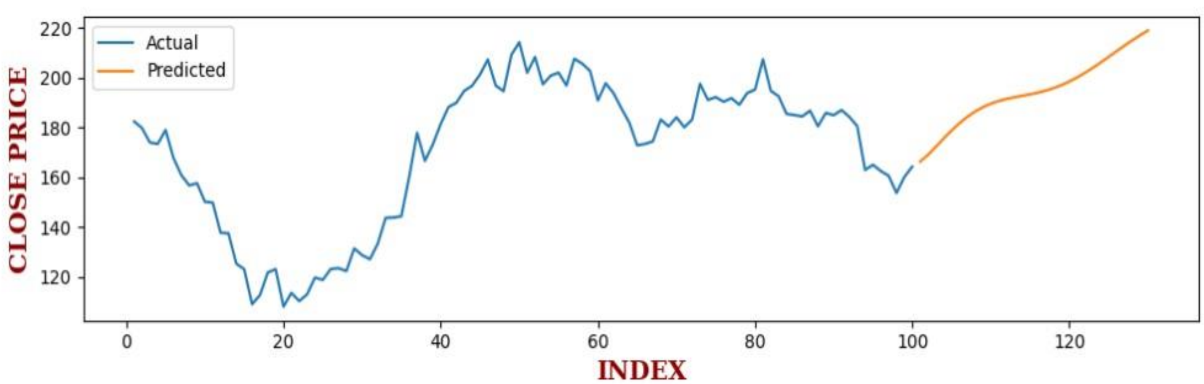
In [41]:
```
day_new = np.arange(1, 101)
day_pred = np.arange(101, 131)
```

In [42]:
```
len(df_close)
```

Out[42]: 1503

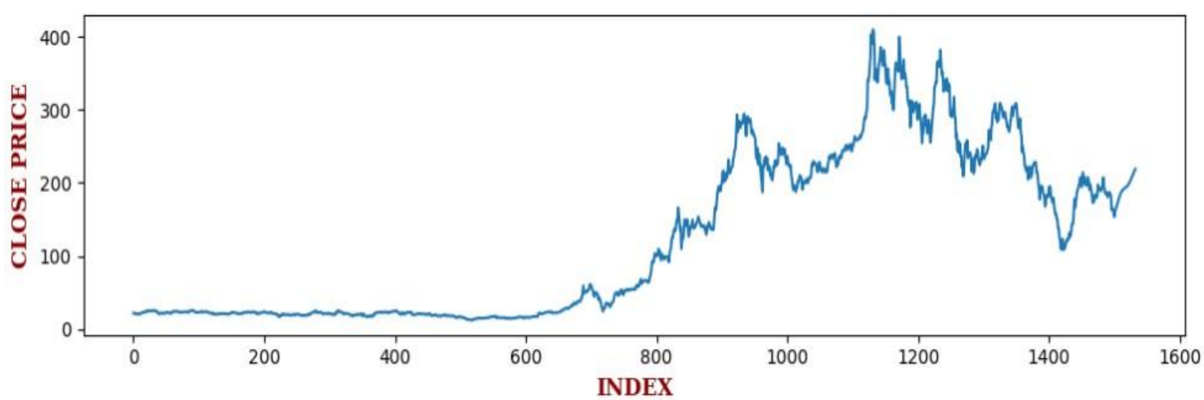
In [43]:
```
lst_df = pd.DataFrame(lst_output)
```

In [46]:
```
font = {'family': 'serif',
        'color':  'darkred',
        'weight': 'bold',
        'size': 14,
        }

plt.figure(figsize=(10, 3))
plt.subplot(1, 1, 1)
plt.plot(day_new, scaler.inverse_transform(df_close[1403:]), label = "Actual")
plt.plot(day_pred, scaler.inverse_transform(lst_df), label = "Predicted")
plt.xlabel('INDEX', fontdict=font)
plt.ylabel('CLOSE PRICE', fontdict=font)
plt.legend()                                    # legend() : it is an area describing the elements of the graph

plt.tight_layout()
```



5.4 FINAL PLOT - AFTER INDEX 500

In [49]:
```
font = {'family': 'serif',
        'color':  'darkred',
        'weight': 'bold',
        'size': 12,
        }

plt.figure(figsize=(10, 3))
plt.subplot(1, 1, 1)
plt.plot(scaler.inverse_transform(final_df))
plt.xlabel('INDEX', fontdict=font)
plt.ylabel('CLOSE PRICE', fontdict=font)
plt.tight_layout()
```



CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this project, we are predicting closing stock price of TESLA (TSLA) organization. We developed a machine learning model for predicting close stock price using LSTM algorithm.

We have used datasets belonging to Yahoo Finance, and achieved a good predicting model with very less difference in RMSE (Root Mean Square Error) values of training and testing data.

There is still a hope of improvement in the accuracy of the model, using different techniques and features.

6.2 Future work

- We want to add sentiment analysis for better analysis.
- Adding more parameters and factors like the financial ratios, multiple instances, etc.
- The more the parameters are taken into account more will be the accuracy.
- The algorithms can also be applied for analyzing the contents of public comments and thus determine patterns/relationships between the customer and the corporate employee.
- In the future, we plan to integrate neural network with some other techniques that can be used to identify optimal network architecture and training parameters and the ability to account for some uncertainty produced by the neural network predictions.