# STOCK PRICE PREDICTION

*A project report submitted in partial fulfillment of the requirements for the*

*Award of degree of*

**Bachelor of Technology**

in

**Computer Science and Engineering**

By

**UVAIS AHMAD (2103400109022)**
**MOHD SHAFAQ (2003400100021)**
**QAYAMUDDIN KHAN (2003400100025)**
**HIRA SHAREEF (2103400139002)**

Under the supervision of

**MR. WASEEM KHAN**
**Assistant Professor**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**VIVEKANAND COLLEGE OF TECHNOLOGY & MANAGEMENT, ALIGARH**
**Affiliated to**

**DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW**
**( 2023-2024)**

# Certificate

I hereby declare that the work which is being presented in the project report entitled, **“STOCK PRICE PEDICTION”**, in partial fulfillment of the requirements for the award of degree of Bachelor of Technology submitted in Computer Science and Engineering of Vivekananda College of Technology & Management ALIGARH, is an authentic record of our own work carried out under the supervision of Mr. Waseem Khan and refers other researcher’s works which are duly listed in the reference section.

The matter presented in this Project has not been submitted for the award of any other degree of this or any other university.

***Uvais Ahmad***
***Mohd Shafaq***
***Qayamuddin Khan***
***Hira Shareef***

This is to certify that the above statement made by the candidates is correct and true to the best of my knowledge.

**Waseem Khan**
Assistant Professor
Vivekananda College of Technology & Management
ALIGARH

**Countersigned by**

Mr. Ajai Verma
HOD - CSE
Vivekananda College of Technology & Management
ALIGARH

# TABLE OF CONTENTS

Countersigned by ........ i
LIST OF TABLES AND FIGURES ........ iv
LIST OF SYMBOLS ........ v
CHAPTER 1 - INTRODUCTION ........ 1
1.1 Background ........ 1
1.2 Motivation ........ 1
1.3 Objective ........ 2
1.4 Outline ........ 3
CHAPTER 2 - EXPERIMENTAL SETUP ........ 4
2.1 Tools and libraries used ........ 4
2.2 Hardware and Software Specification ........ 5
CHAPTER 3 - LITERATURE REVIEW ........ 6
3.1 Overview of Stock Price Prediction Techniques ........ 6
3.2 Traditional Methods vs. Machine Learning Approaches ........ 6
3.3 LSTM (Long Short-Term Memory) Networks ........ 7
3.4 Previous Studies on Stock Price Prediction using LSTM ........ 10
CHAPTER 4 – METHODOLOGY ........ 12
4.1 Data Collection ........ 12
4.2 Description of Dataset ........ 12
4.3 Data Preprocessing ........ 13
4.4 Data Splitting ........ 14
4.5 Model Selection ........ 14
4.6 Model Architecture ........ 14
4.7 Model Compilation ........ 17
4.8 Training the Model ........ 17
4.9 Evaluation of Model ........ 18
CHAPTER 5 - CODING ........ 19
5.1 Backend ........ 19
5.2 Frontend ........ 31
CHAPTER 6 - RESULT AND CONCLUSION ........ 62
6.1 LSTM Performance ........ 62
6.2 Conclusion ........ 64
6.3 Future Scope ........ 64

CHAPTER 7 – SNAPSHOTS ........................................................................................ 66
7.1 Home Page:................................................................................................... 66
7.2 Training Page:.............................................................................................. 66
7.3 Prediction Page: ........................................................................................... 67
CHAPTER 8 - REFERENCES.................................................................................... 71

# ACKNOWLEDGEMENT

First and foremost, we would like to express our deepest gratitude to ALLAH Almighty for His infinite guidance, strength, and blessings throughout this journey. Without His grace and mercy, this thesis would not have been possible.

We would like to extend our heartfelt appreciation to our supervisor, Waseem Khan, for his invaluable guidance, insightful feedback, and unwavering support. His expertise and encouragement have been crucial in the completion of this project. His dedication and mentorship have inspired us to strive for excellence and have greatly contributed to our academic and personal growth.

We also take the opportunity to acknowledge the contribution of Mr. Ajai Verma, Head, Department of Computer Science & Engineering, Vivekananda College of Technology & Management, ALIGARH for his full support and assistance during the development of the project.

We also do not like to miss the opportunity to acknowledge the contribution of all project co-coordinators and faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our families and friends for their contribution in the completion of the project.

| | |
|---|---|
| *Signature:* | *Signature:* |
| *Name:* | *Name:* |
| *Roll No.:* | *Roll No.:* |
| *Date:* | *Date:* |
| *Signature:* | *Signature:* |
| *Name:* | *Name:* |
| *Roll No.:* | *Roll No.:* |
| *Date:* | *Date:* |

# ABSTRACT

Stock market is a very volatile sector where sudden changes in the share value are common. There are many factors responsible for this change like – physical factors vs. physiological, rational and irrational behavior, investor sentiment, market rumors, government policies etc. These changes may be small but their impact on the market can be drastically huge.

Predicting stock price is a very tough task when such variabilities are there and getting high accuracy is much more difficult. Stock price prediction is a time series problem as the stock value fluctuates very often. To deal with time series problems we have several tools like SVM (Support Vector Machines), LSTM (Long Short-Term Memory).

Stock data is non-linear in nature and to handle such data and forecasting desired result SVM (Support Vector Machines) and LSTM (Long Short-Term Memory) are the most efficient techniques. LSTM is a type of RNN which is used to learn and process sequential data.

We made an attempt to understand stock market functionalities and to predict stock prices by means of machine learning techniques. In this report we have discussed about time series model, LSTM neural network, real time stocks in the market then use the root mean square error (RMSE) to calculate the error and compare the prediction results.

We have obtained historical data from Yahoo finance website. We take Google as our practice dataset. The dataset has more than 4000 records over 15 years which makes it very versatile to be implemented on machine learning algorithms for a better result. Dataset contains 5 columns - Open, Close, High, Low, Population.

# LIST OF TABLES AND FIGURES

Table 1 Dataset of stock price of Google

Table 2 Accuracy of Google's stock

Figure 1 Internal Architecture of LSTM Algorithm

Figure 2 Layered Architecture of LSTM model

Figure 3 Google's stock visual accuracy graph

Figure 4 interface for predicting next day closing price

Figure 5 Home Page

Figure 6 Training page

Figure 7 Selecting dataset from local drive

Figure 8 Prediction page

Figure 9 Prediction of TCS.NS stock

Figure 10 Prediction of Wipro.NS stock

Figure 11 Prediction of HCLTECH.NS

Figure 12 Prediction of Reliance.NS stock

Figure 13 Prediction of Google stock

# LIST OF SYMBOLS

| | |
|---|---|
| $\mathbf{x}$ | Input feature vector |
| $y$ | Target output vector |
| $w$ | Weight matrix for the neural network |
| $b$ | Bias term |
| $\alpha$ | Learning rate |
| $\epsilon$ | Error term |
| $E$ | Total error or loss |
| $\sigma$ | Activation function |
| $\delta$ | Change in parameters during backpropagation |
| $L$ | Loss function |
| $t$ | Time step |
| $\mu$ | Mean of a distribution |
| $\sigma 2$ | Variance of a distribution |
| $D$ | Dataset |
| $MS\mathrm{E}$ | Mean Squared Error |
| $M\mathrm{AE}$ | Mean Absolute Error |
| $R2$ | Coefficient of determination |
| $LS\mathrm{TM}$ | Long Short-Term Memory |

# CHAPTER 1 – INTRODUCTION

## 1.1 Background

Stock Market is very volatile as the prices for the stocks may fluctuate very often due to enormous factors. Predicting behavior of such factors may seem to be impossible and hence people crave for methods which are capable of doing so. Stock market fascinates mass for a long time and experiments to find the methods which are capable in predicting stock price are being carried out since then.

As a large population invest in stock market, accurate prediction of stock price becomes very important. A small inconsistency may trigger the butterfly effect and result in loss of huge money. A model considered for the stock price prediction is supposed to be accurate enough to forecast the closest predicted price of stock.

When data is highly dependent on closer event in time than farther event in time then these types of problems are called time series problems. Stock price prediction is a time series problem because stock price on one day is highly dependent on the stock price on the previous day.

## 1.2 Motivation

Due to high volatility of data stock price prediction becomes a complex and challenging task. This task can be handled by various machine learning algorithms. However, some challenges are there that need to be addressed manually to predict stock price accurately. Data quality, nonlinear factors like economic indicators, market sentiment, news, and events; data volume, overfitting are some major challenges that need to be handled while predicting the stock price.

Machine learning and deep learning methods are capable of identifying structure and pattern of data such as non-linearity and complexity in time series forecasting. In particular, LSTM has been used in many application domains such as natural language processing (Tarwani), speech recognition (Eyben), time-series prediction (S. Hochreiter and J. Schmidhuber);  (A. Giusti et al.); (Roondiwala), as well as its applications in economics and finance data such as predicting the volatility of the S&P 500 (Nowrouz Kohzadi; Dematos) and measuring the impact of incorporating news for selected companies (Ding).

## 1.3 Objective

Stock price prediction using machine learning is a dynamic approach to forecast the prices of stocks. Stock price prediction can be done in many ways like news analysis, using historical price of stock, market trends etc. Many popular traditional methods, like ARIMA, PROPHET, GARCH, VAR etc., are there which are capable in prediction of stock price. But deep learning models outperforms these traditional methods in terms of accuracy.

Long Short Term Memory (LSTM} is a deep learning method which has outperformed other traditional as well as machine models and gave fascinated results. We have trained the LSTM network with the historical data of stocks. As we know news articles, financial news, market news etc., have a big impact on the future trends in stock market. Predicting nature of these factors is also possible but we have used historical data to predict future trends of stock.

We have to handle stock dataset and extract valuable insights from them. Then this cleaned dataset is used to train LSTM network to create LSTM model. We have set epochs 50 for doing so. Number of times a dataset is passed to the algorithm to train a model is called epoch. It has been seen that low data with high epoch gives the lowest error in predicting future price of stock.

## 1.4 Outline

In this report we have 7 chapters Introduction, Experimental Setup, Literature Review, Methodology, Coding,  Results and Conclusion and snapshots.

We have discussed thoroughly about stock price prediction model using LSTM network in the introduction. Stock market volatility, machine learning approaches to tackle the problems in predicting future trends in the market, an overview of LSTM network is provided in this chapter.

Tools and framework used for successfully working of model are listed in second chapter. Software and hardware requirements are also discussed in this section.

Literature review contains previous researches in the application of LSTM network to predict stock price. Various methods are discussed in this section that are proved to be efficient in stock price prediction. Structural architecture of LSTM network is explained in this chapter.

In methodology chapter, roadmap of prediction of stock price is explained. Data is collected, preprocessed and split in training and testing then model is trained using this cleaned data. Predicted price is achieved and accuracy is calculated in this chapter.

Next chapter is coding which contains all the source code of frontend and backend.

Snapshots are shown in next chapter. A clear visualization in provided for all web pages and predicted results of different dataset of stocks.

In last chapter we have provided references to our study and citation. This include some research articles as well as books and websites.

# CHAPTER 2 – EXPERIMENTAL SETUP

## 2.1 Tools and libraries used

**Pandas 2.2.1**: Pandas is a python library primarily used in data preprocessing. Pandas provides dataframe which makes it easier to manipulate raw data such as insertion or deletion of columns, eliminating null and duplicate values, encoding categorical columns etc.

**NumPy 1.24.2**: NumPy is also a python library which is used in mathematical operations. It is very helpful when handling large dataset. Multidimensional data can be handled easily by using NumPy library. In feature engineering and model training NumPy turns out very helpful.

**Scikit-learn 1.3.0**: Scikit-learn is a popular and powerful machine learning library for python It features various classification, regression, clustering, and dimensionality reduction algorithms. It is easy to import and implement the desired machine learning algorithm.

**TensorFlow 2.13.0**: TensorFlow is a powerful tool for machine learning and deep learning. It can be used to create and train models for a variety of tasks, including natural language processing, speech recognition, and image recognition. It is easy to load models using TensorFlow as it provides a surface to the Keras models.

**Keras 2.13.1:** Keras is a high level API of TensorFlow used to build and train deep learning models. Keras provide an easy way to define and train variety of neural networks like CNNs, RNNs etc.

**Flask 2.2.2:** Flask is a python library used in web development. It can be used in creating small and simple as well as large and complex web applications. Flask can host a web page on a temporary server.

**Flask CORS 4.0.0:** Flask-CORS is a powerful tool that can help you to secure your Flask application and enable cross-origin communication.

## 2.2 Hardware and Software Specification

- Window 10 or above
- Python 3.8 or above
- 4GB RAM or more
- Minimum 2 GB free memory for dataset storage
- Install Microsoft Visual C++ Redistributable

# CHAPTER 3 – LITERATURE REVIEW

## 3.1 Overview of Stock Price Prediction Techniques

For a long time, it has been a complicated challenge for the mathematicians to analyze the stock price patterns and to forecast the future trends in the stock market. Luckily now we have several techniques to predict the future price of stocks. These techniques include traditional methods like Auto– Regressive Moving Average (ARMA) model, auto– Regressive Integrated Moving Average (ARIMA) model, the generalized autoregressive conditional heteroscedasticity (GARCH) model, the vector auto regression (VAR) model etc. as well as machine learning techniques like Support Vector Machines (SVMs), Genetic Algorithms (GAs), Fuzzy Logic (FL) etc.
As new researches are being done in predicting the future trends of the stock market, new approaches are coming into existence which make forecasting the stock prices better and better. In comparison of different stock price prediction techniques RMSE (Root Mean Square Error} plays as an indicator to choose among various techniques.

## 3.2 Traditional Methods vs. Machine Learning Approaches

Traditional methods are capable in predicting the stock prices but their accuracies are not good. Traditional methods can predict the market trends but when it comes to predict the future prices of stocks, they are not reliable. In traditional methods we analyze economic factors, financial factors and other quantitative and qualitative factors.
Machine learning methods are very good alternative for stock price prediction. Linear regression decision trees, support vector machines (SVM}, convolutional neural network, recurrent neural network are some machine learning approaches used in stock price prediction. These techniques are capable in handling more

complex data to train the model and extract meaningful insights. These techniques are much better than traditional methods as their accuracy in comparison of traditional methods is significantly higher than traditional methods. In machine learning approaches we use can use market news, financial news or historical data of stock value.

## 3.3 LSTM (Long Short-Term Memory) Networks

LSTM was first proposed by German researchers Sepp Hochreiter and Juergen Schmidhuber in 1997. Long Short Term Memory network is a type of deep learning neural network. It is different from ordinary deep learning networks as it has an extra memory space for storing the past fluctuations of data. It consists of an input gate, a forget gate and an output gate. Forget gate returns a value between 0 and 1 where 0 means to forget all the information from last cell and 1 means store all information from last cell.

LSTM can be used in time series problem like stock price prediction, weather forecasting, sales and demand forecasting etc. LSTM is capable in detecting unusual patterns in network traffic that may indicate security threats, in machine translation, speech recognition and other NLP models.

LSTM cell is a basic component of a large LSTM network. Each cell is made up of three gates – input gate, forget gate and output gate. All these gates are the decision makers in the cell. The LSTM architecture consists of a set of recurrently connected sub-networks, known as memory blocks. The idea behind the memory block is to maintain its state over time and regulate the information flow thought nonlinear gating units. Fig. 1 displays the architecture of a LSTM block, which involves the gates, the input signal $x^{(t)}$, the output $y^{(t)}$ and the activation functions. The output of the block is recurrently connected back to the block input and all of the gates.

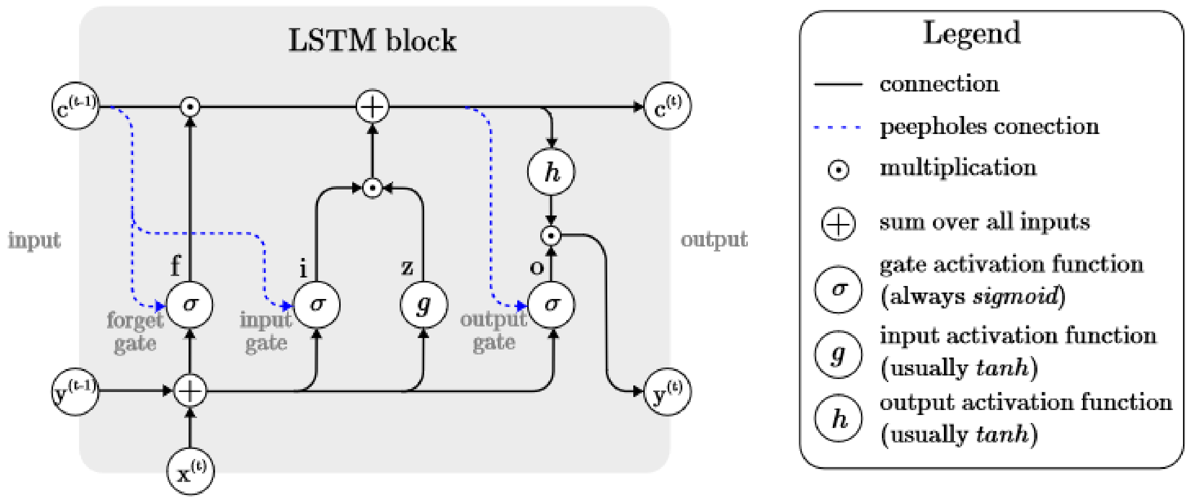


*Figure 14 Internal Architecture of LSTM Algorithm*

Aiming to clarify how the LSTM model works, let us assume a network comprised of N processing blocks and M inputs. The forward pass in this recurrent neural system is described below.

**Block input :** This step is devoted to updating the block input component, which combines the current input $x^{(t)}$ and the output of that LSTM unit $y^{(t-1)}$ in the last iteration. This can be done as depicted below:

$$z^{(t)} = g(W_z x^{(t)} + R_z y^{(t-1)} + b_z)$$

where $W_z$ and $R_z$ are the weights associated with $x^{(t)}$ and $y^{(t-1)}$, respectively, while $b_z$ stands for the bias weight vector.

**Input gate** : In this step, we update the input gate that combines the current input $x^{(t)}$, the output of that LSTM unit $y^{(t-1)}$ and the cell value $c^{(t-1)}$ in the last iteration. The following equation shows this procedure:

$$i^{(t)} = \sigma(W_i x^{(t)} + R_i y^{(t-1)} + p_i \cdot c^{(t-1)} + b_i)$$

where denotes point-wise multiplication of two vectors, $W_i$, $R_i$ and $p_i$ are the weights associated with $x^{(t)}$, $y^{(t-1)}$ and $c^{(t-1)}$, respectively, while $b_i$ represents for the bias vector associated with this component.

In the previous steps, the LSTM layer determines which information should be retained in the network's cell states $c^{(t)}$. This included the selection of the candidate values $z^{(t)}$ that could potentially be added to the cell states, and the activation values $i^{(t)}$ of the input gates.

**Forget gate :** In this step, the LSTM unit determines which information should be removed from its previous cell states $c^{(t-1)}$. Therefore, the activation values $f^{(t)}$ of the forget gates at time step $t$ are calculated based on the current input $x^{(t)}$, the outputs $y^{(t-1)}$ and the state $c^{(t-1)}$ of the memory cells at the previous time step *(t − 1)*, the peephole connections, and the bias terms bf of the forget gates. This can be done as follows:

$$f^{(t)} = \sigma(W_f x^{(t)} + R_f y^{(t-1)} + p_f \cdot c^{(t-1)} + b_f)$$

where $W_f$ , $R_f$ and $p_f$ are the weights associated with $x^{(t)}$, $y^{(t-1)}$ and $c^{(t-1)}$, respectively, while bf denotes for the bias weight vector.

**Cell :** This step computes the cell value, which combines the block input $z^{(t),}$ the input gate $i^{(t)}$ and the forget gate $f^{(t)}$ values, with the previous cell value. This can be done as depicted below:

$$c^{(t)} = z^{(t)} \cdot i^{(t)} + c^{(t-1)} \cdot f^{(t)}$$

**Output gate :** This step calculates the output gate, which combines the current input $x^{(t)}$, the output of that LSTM unit $y^{(t-1)}$ and the cell value $c^{(t-1)}$ in the last iteration. This can be done as depicted below:

$$o^{(t)} = \sigma(W_o x^{(t)} + R_o y^{(t-1)} + p_o \cdot c^{(t)} + b_o)$$

where $W_o$ , $R_o$ and $p_o$ are the weights associated with $x^{(t)}$, $y^{(t-1)}$ and $c^{(t-1)}$ respectively, while $b_o$ denotes for the bias weight vector.

Block output. Finally, we calculate the block output, which combines the current cell value $c^{(t)}$ with the current output gate value as follows:

$$y^{(t)} = g(c^{(t)}) \cdot o^{(t)}$$

In the above steps, $\sigma$, $g$ and $h$ denote point-wise non-linear activation functions. The logistic sigmoid $\sigma(x) = \frac{1}{1+e^{1-x}}$ is used as a gate activation function, while the hyperbolic tangent *g(x) = h(x) = tanh(x)* is often used as the block input and output activation function.

## 3.4 Previous Studies on Stock Price Prediction using LSTM

Adil Moghar et al. used the LSTM model on GOOG and NKE stocks to predict future trends[3]. They divided dataset into 8:2 ratio for training and testing purpose. They concluded that training with less data with more epochs can improve the forecasting values. As we increase the number of epoch accuracy increase.

C.K.-S. Leung et al. experimented with data mining and machine learning techniques to train structured support vector machine (SSVM) to predict stock price[4]. The result shows that the best outcome was achieved by using 3-fold cross validation with C=1000. Accuracy of SSVM on training samples was more than 78% and on the testing samples was about 50%.

Chouskey and Hoare implemented three models ARIMA, PROPHET and LSTM to predict the stock price of HDFC bank[1]. Dataset was divided into training and testing subsets in different ratios (6:4, 7:3, 8:2, 9:1). ARIMA and PROPHET shows lowest RMSE value for 9:1 training testing dataset split and highest RMSE value for 6:4 training testing dataset split. LSTM shows lowest RMSE value for 8:2 training testing dataset split and highest RMSE value for 6:4 training testing dataset split. It can be noticed that RMSE values for LSTM model are significantly low than that of ARIMA and PROPHET. Overall LSTM outperformed other two methods which suggests that LSTM is capable to predict stock prices with satisfactory outcomes.

Shen and Shafiq proposed a deep neural network model to predict stock price of by providing 29 features[5]. They used different methods like Support Vector Machine, Multilayer perceptron artificial neural network, Naïve Bayes Classifier, Random Forest Classifier, Logistic Regression Classifier. Long Short-Term Memory (LSTM) gives the maximum accuracy of 93.25% which is significantly high.

In 2021, Surayagari Hari Kiran Sai et al. published his research in prediction of stock prices by using machine learning techniques[6]. Stock dataset was used from NASDAQ and news insights was collected from famous financial websites. They attempt to establish a correlation between news and historical data of the stocks. Since Google, Microsoft, Amazon are some of the most stable stocks, they showed lowest RMSE means high accuracy. More than 96% accuracy was achieved in this research.

Roondiwala et al. in 2017 experimented LSTM with NIFTY 50 index on a dataset from 2011 to 2016[7]. The authors used historical data instead of macroscopic factors to predict the future price of stock and achieved very low RMSE values.

Siami-Namini and Namin compared the LSTM and traditional algorithm ARIMA and concluded that LSTM outperforms ARIMA with reduction in error rate of 84 – 87 percent[8]. The authors also noticed that number of epochs has no significant effect on trained data.

Chen et. al. proposed a GRU based model to predict the stock price of liquor industry, pharmaceutics industry, banking industry and cinema industry[2]. The authors used various methods to do so and LSTM showed promising result. Although LSTM had not the best RMSE score, it had enough accuracy to use it in prediction.

# CHAPTER 4 – METHODOLOGY

## 4.1 Data Collection

Historical dataset for the purpose of analyzing market trends is available on various website in many formats. There is a large number of websites including nasdaq.com, investing.com, fidelity.lu, finance.yahoo.com, morningstar.in that provide historical data of the stock over a large period of time. Stock price data for the desired stock can be accessed by various methods including web scrapping and manual data extraction.

Some websites facilitate users to download the dataset. We have used data of each stock from yahoo finance website as it is easiest to access and data is almost cleaned and well organized. This dataset covers the period from the date the stock was listed in the market to the current date.

## 4.2 Description of Dataset

| Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| 2023-04-10 | 107.389999 | 107.970001 | 105.599998 | 106.949997 | 106.949997 | 19741500 |
| 2023-04-11 | 106.919998 | 107.220001 | 105.279999 | 106.120003 | 106.120003 | 18721300 |
| 2023-04-12 | 107.389999 | 107.586998 | 104.970001 | 105.220001 | 105.220001 | 22761600 |
| 2023-04-13 | 106.470001 | 108.264999 | 106.440002 | 108.190002 | 108.190002 | 21650700 |
| 2023-04-14 | 107.690002 | 109.580002 | 107.589996 | 109.459999 | 109.459999 | 20758700 |
| 2023-04-17 | 105.43 | 106.709999 | 105.32 | 106.419998 | 106.419998 | 29043400 |
| 2023-04-18 | 107 | 107.050003 | 104.779999 | 105.120003 | 105.120003 | 17641400 |

*Table 2 Dataset of stock price of Google*

In each file of dataset seven columns are there – Date, Open, High, Low, Close, Adj Close and Volume.

- Date column indicates the date of price fluctuations.
- Open column stores the opening price of stock on the specified date.
- High column stores the highest stock price touched by the stock index on that particular day.
- Similarly Low column includes lowest hit price by the stock on that day.
- Close column stores the price at the end of the day.
- Adj Close column stores the adjusted closing price of stock and
- Volume column stores the total number of shares that have been bought or sold in that specific trading day.

There is always the issue of overfitting with machine learning, and it is also possible that the algorithm can find correlations without causation. There is a possibility that there may not be enough data concerning stock prices. There is no real way to know if there is too much or too little data. For each of the stocks from the yahoo finance, Open, Close, High, Low, Adj Close and volume values were used.

## 4.3 Data Preprocessing

As we have already mentioned data collected from yahoo finance website is free from null values. But if some dataset which is downloaded from other website given to our model there is a chance that it may have null values. So first of all, we drop all null columns if any.

Except for date column all columns stored in *res* list and close column is stored in *closeColName*. Data normalization is another crucial step in data preprocessing. The 'Close' prices were normalized using Min-Max scaling to ensure consistency across different stocks.

## 4.4 Data Splitting

Data is split in training set and testing set as last 60 days data is used as testing data and whole data is given to the algorithm to train the network.

## 4.5 Model Selection

Stock price prediction can be done by various means of approaches. Many researches have been done in prediction of stock market trends using SVM (Support Vector Machine}, ARIMA (Autoregressive Integrated Memory Average} as well as LSTM.

It has been found that deep learning based LSTM outperformed all of these methods and comes out with most accurate results. As stock market is very sensitive, we cannot risk in terms of money. Hence, we choose LSTM model for predicting stock price with high accuracy and reliability.

LSTM gives around 78% less RMSE value than the popular support vector machine model. LSTM is a recurrent neural network with an extra memory for storing data from long term fluctuations in the dataset.

The LSTM algorithm was chosen due to its ability to capture temporal dependencies and patterns in time-series data, making it suitable for stock price prediction.

Stock price prediction is time series problem and hence Sequential model of Keras is imported and LSTM network is used as neural network. Split dataset is then provided to the network

## 4.6 Model Architecture

In this section, we provide a comprehensive description of the Long Short-Term Memory (LSTM) model architecture used for stock price prediction. This includes the specific layers, the number of neurons in each layer, activation functions, and additional configurations like dropout and dense layers.

**Input Layer:** This layer accepts sequences of past stock prices. Data from training set is given to the input layer where the data is organized to pass to the next layer. The input layer of the model is designed to accept sequences of past stock prices. Each input sequence consists of 60 past stock price values (one for each day in the sequence). The shape of the input data for the LSTM model is therefore (60, 1), where 60 is the number of time steps and 1 is the number of features per time step (i.e., the stock price).

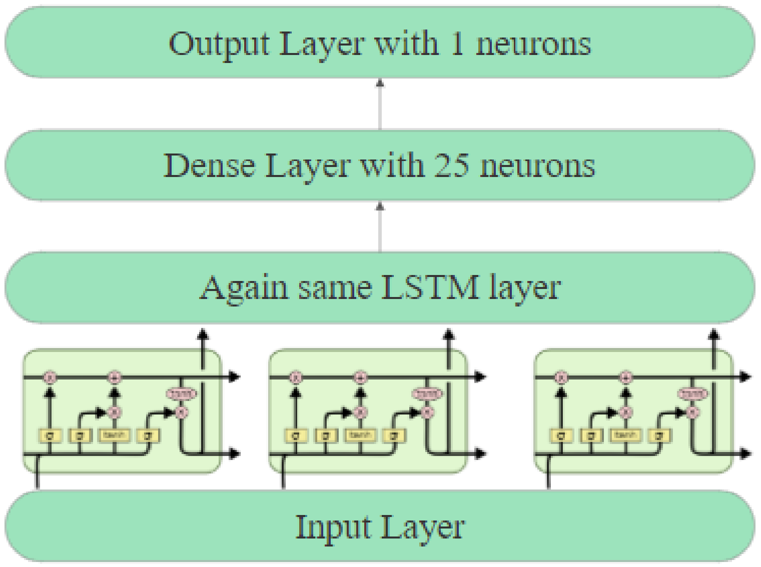


*Figure 15 Layered Architecture of LSTM model*

**LSTM Layers:** Two LSTM layers with 50 neurons each are used and a dropout layer to prevent overfitting is organized followed by them. Each LSTM layer uses default activation function 'tanh'. In first layer return_sequences is True which means output of this layer will be a sequence that fed to the next layer and in second layer return_sequences is False which means output of

**Dense Layers:** Following the LSTM layers, the model includes Dense (fully connected) layers to perform the final prediction. Dense layers help in interpreting the learned features from the LSTM layers and map them to the output space.

A dense layer with 25 neurons is imposed followed by a final output layer with a single neuron to predict the closing price.

**Activation Functions:** The activation function is a fixed mathematical operator; it takes a single number and performs certain fixed mathematical operations on it. 'ReLU' (Rectified Linear Unit) activation for hidden layers and linear activation for the output layer. There are different activation functions, some of the more common activation functions are:

**Sigmoid**: It gives an output in the range between 0 and 1.

$$\sigma(x) \;=\; \frac{1}{1+e^{1-x}}$$

**Tanh**: It gives an output in the range between -1 and 1.

$$tanh(x) \;=\; 2\sigma(2x) \;-\; 1$$

**ReLU**: In this type of function zero is the threshold of activation.

$$f\;(x) \;=\; max(0, x)$$

**Summary of Model Parameters**

- **Input Shape:** (60, 1)
- **First LSTM Layer:** 50 neurons, return sequences=True
- **Second LSTM Layer:** 50 neurons, return sequences=False
- **First Dense Layer:** 25 neurons, ReLU activation
- **Output Layer:** 1 neuron, Linear activation
- **Optimizer:** Adam
- **Loss Function:** Mean Squared Error (MSE)

## 4.7 Model Compilation

The model is compiled using the Adam optimizer and Mean Squared Error (MSE) as the loss function.

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

## 4.8 Training the Model

The training process involves fitting the model on the training data and validating it on the validation split. The model is trained for 100 epochs with a batch size of 50, using 20% of the training data as validation data. This validation data is used to monitor the model’s performance on unseen data.

```
history_data = model.fit(Xtrain, Ytrain, batch_size=50,
        epochs=epochs, validation_split=0.2, verbose=0,
        callbacks=[EpochPrintingCallback(updateEpochs=updateEpochs)])
```

Once the model is trained, it is saved in the local disk for the future use.

```
model.save('pretrained/stock_model.h5')
```

## 4.9 Evaluation of Model

Several metrics are commonly used to evaluate the performance of predictive models. In the context of stock price prediction, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) are widely used metrics.

- **Mean Squared Error (MSE):** MSE measures the average squared difference between the predicted values and the actual values. It provides a measure of the model's accuracy, with lower values indicating better performance.

$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n}$$

- **Root Mean Squared Error (RMSE):** RMSE is the square root of the MSE and provides an interpretable measure of the average prediction error. Like MSE, lower values of RMSE indicate better model performance.

$$RSME = \sqrt{\frac{\sum(y_i - \hat{y}_i)^2}{N - P}}$$

- **Mean Absolute Error (MAE):** MAE measures the average absolute difference between the predicted values and the actual values. It provides a more robust measure of error that is less sensitive to outliers.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} | x_i - x |$$

# CHAPTER 5 – CODING

## 5.1 Backend

### Code of app.py file

```
from flask import Flask, request
from flask_cors import CORS
import glob
import json
# import h5py

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

df = None
cols, dateColName, closeColName = None, None, None
train_size = 0.75
totalEpochs = 2

session = {
    "training": {
        "status": "ready",
        "fileUploaded": False,
        "fileName": None,
        "totalEpochs": totalEpochs
    },
    "prediction": {
        "status": "ready",
        "preTrainedModelNames": None
    }
}

def updateEpochs(epoch):
    global session

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

from api import *


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

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

        df = pd.read_csv(request.files['file'])
        df = df.dropna()

        cols, dateColName, closeColName = getRequiredColumns(df)
        print(cols)
        dfColVals = []
        dfDateVals = []
        dfCloseVals = []
        for row in df[[dateColName] + cols].values:
            dfColVals.append(list(row))
            dfCloseVals.append(row[4])
            dfDateVals.append(row[0])

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

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

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

        fileName = request.form['fileName']

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

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

        # json = LMS(df, closeColName, next_days=10, epochs=100,
updateEpochs=updateEpochs)
        model = LSTMAlgorithm(fileName, train_size, totalEpochs,
updateEpochs=updateEpochs)
```

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

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

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


# Prediction Page

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

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

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

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

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

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

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

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

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

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

        fileName = request.form['fileName']

        openValue = request.form['openValue']
        highValue = request.form['highValue']
        lowValue = request.form['lowValue']
        volumeValue = request.form['volumeValue']


        prediction = getManualPredictionForModel(fileName, 
train_size, openValue, highValue, lowValue, volumeValue)

        session['prediction']['manualPrediction'] = 
str(prediction)

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

if __name__ == '__main__':

    debug = False
    # debug = True
    port = 7676

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

Code for app.py

## Code of api.py file

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import numpy as np
import pandas as pd

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

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

    return X_scaled, omax, omin

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

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

    return X

def getColumnsData(df, cols):
    print("Retriving", ' '.join(cols), "Columnn(s)")
    return df[cols]

def getRequiredColumns(df):
    res = []
    dateColName = None
    closeColName = None

    for col in df.columns:
        if (('date' in col.lower()) or ('time' in col.lower())):
            dateColName = col
            break

    for col in df.columns:
        if ('open' in col.lower()):
            res.append(col)
            break

    for col in df.columns:
        if ('low' in col.lower()):
            res.append(col)
            break

    for col in df.columns:
        if ('high' in col.lower()):
            res.append(col)
            break

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

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

    return res, dateColName, closeColName

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

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

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

    x = np.array(tmp)

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

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

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

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

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

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

                wn = wn + 2*u*e*xn

        return y,wn;

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

    l=next_days;
    N=100;

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

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

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

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

import keras

class EpochPrintingCallback (keras.callbacks.Callback):
    def __init__(self, updateEpochs):
        self.updateEpochs = updateEpochs

    def on_epoch_end(self, epoch, logs=None):
        print(epoch)
        self.updateEpochs(epoch)
```

```
import warnings
warnings.filterwarnings('ignore')

import math
import pandas as pd
import numpy as np
import random
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from keras.layers import LSTM
from keras.layers import Dense
from keras.models import Sequential
import h5py

os.environ['CUDA_VISIBLE_DEVICES'] = '-1'

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

    scaling_data_frame = df.filter(cols)

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

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

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

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

    StockTrainData = scaledData[0:trainingDataLength , :]

    Xtrain = []
    Ytrain = []

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

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

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

    neurons = 50

    model = Sequential()

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

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

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

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

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

    return model

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

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

    scaling_data_frame = df.filter(cols)

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

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

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

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

    StockTrainData = scaledData[0:trainingDataLength , :]

    Xtrain = []
    Ytrain = []

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

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

    Xtrain = np.reshape(Xtrain, (Xtrain.shape[0], Xtrain.shape[1], 
1))
    testingData = scaledData[trainingDataLength - 60: , :]

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

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

    # predictions

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

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

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

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

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

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

    realData = list(real)

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

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

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

    return json

def getManualPredictionForModel(fileName, train_size, openValue, 
highValue, lowValue, volumeValue):
    df = pd.read_csv('./datasets/' + fileName + '.csv')
    cols, dateColName, trade_close_col = getRequiredColumns(df)
    close_idx = -1
    for col in df.columns:
        close_idx += 1
        if(col == trade_close_col): break

    row = []
    for i in range(df.shape[1]):
        if(i==close_idx):
row.append(random.randint(int(float(lowValue)),
int(float(highValue))))
        else: row.append(0)
    df.loc[df.shape[0]] = row

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

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

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

    trainingDataLength = math.ceil( len(stock_close_dataset) *
train_size )-1

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

    testingData = scaledData[trainingDataLength - 60: , :]

    Xtest = []
    for i in range(60, len(testingData)+1):
        Xtest.append(testingData[i-60:i, 0])

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

    predictions = model.predict(Xtest)
    predictions = scaler.inverse_transform(predictions)
    return predictions[-1][0]
```

Code for api.py

## 5.2 Frontend

### Code of home.html file

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

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

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

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

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

        <script>

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

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

        </script>

    </head>
    <body>
        <div id="header">
            <div class="container-fluid">
                <div class="row">
```

```
                    <div class="col-sm-4 header-title">Stock Price
Prediction</div>
                    <div class="col-sm-8">
                        <div class="navbar">
                            <div onclick="changePage(this)"
page="home" class="nav-item active">Home</div>
                            <div onclick="changePage(this)"
page="training" class="nav-item">Training</div>
                            <div onclick="changePage(this)"
page="predictions" class="nav-item">Predictions</div>
                        </div>
                    </div>
                </div>
            </div>
        </div>

        <div id="content">
            <link rel="stylesheet" href="./home.css">

            <div class="welcomeBanner">
                <img class="background"
src="../icons/background.jpg" />
                <div class="text">
                    <div class="title">Stock Price
Prediction</div>
                    <div class="subtitle">A place for better
predictions</div>
                </div>
            </div>

            <div class="container-fluid someStocks">
                <div class="title">Some Stocks</div>
                <div class="row">
                    <div class="col-sm-3"><img class="google icon"
src="../icons/companies/google.png" /></div>
                    <div class="col-sm-3"><img class="microsoft
icon" src="../icons/companies/microsoft.png" /></div>
                    <div class="col-sm-3"><img class="nifty50
icon" src="../icons/companies/nifty50.png" /></div>
                    <div class="col-sm-3"><img class="reliance
icon" src="../icons/companies/reliance.png" /></div>
                </div>
            </div>
        </div>
    </body>
</html>
```

Code for home.html

## Code of home.css file

```
.welcomeBanner {
    position: relative;
    width: 100%;
    height: 350px;
}

.welcomeBanner .background {
    position: absolute;
    width: 100%;
    height: 100%;
    z-index: -1;
}

.welcomeBanner .text {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}

.welcomeBanner .text .title {
    font-size: 25px;
    text-align: center;
    color: white;
}

.welcomeBanner .text .subtitle {
    text-align: center;
    color: white;
}

.welcomeBanner .companies .icon {
    position: absolute;
    width: 150px;
    opacity: 0.7;
}

.SomeStocks {
    margin-top: 40px;
}

.SomeStocks .title {
    text-align: center;
    font-size: 20px;
}

.SomeStocks .icon {
    width: 100%;
    padding: 0 70px;
    margin-top: 20px;
}

.SomeStocks .google {
    margin-top: 25px;
    padding: 0 80px;
}

.SomeStocks .microsoft {
    margin-top: 30px;
}

.SomeStocks .nifty50 {
    margin-top: 30px;
    padding: 0 95px;
}

.SomeStocks .reliance {
    margin-top: 30px;
    padding: 0 110px;
}
```

Code for home.css

## Code of training.html file

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

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

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

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

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

        <script>

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

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

        </script>

    </head>
    <body>
        <div id="header">
            <div class="container-fluid">
                <div class="row">
                    <div class="col-sm-4 header-title">Stock Price
Prediction</div>
                    <div class="col-sm-8">
```

```
                        <div class="navbar">
                            <div onclick="changePage(this)"
page="home" class="nav-item">Home</div>
                            <div onclick="changePage(this)"
page="training" class="nav-item active">Training</div>
                            <div onclick="changePage(this)"
page="predictions" class="nav-item">Predictions</div>
                        </div>
                    </div>
                </div>
            </div>
        </div>

        <div id="content">
            <link rel="stylesheet" href="./training.css">

            <div class="upload">
                <button onclick="$('.selectFile').click()">New
Training Data</button>
                <input class="selectFile" type="file" />
            </div>

            <center>
                <div class="container-fluid datasetProperties">
                    <div class="placeholder">
                        <div class="layer">
                            <div class="layerName">Dataset</div>
                            <div class="layerDesc">Click on "New
Training Data" button to load the dataset</div>
                        </div>
                    </div>
                    <div class="filename">Filename: <input
class="fileNameInput" type="text" /></div>
                    <div class="row">
                        <div class="col-sm-7">
                            <div class="dfHead"></div>
                        </div>
                        <div class="col-sm-5
closePriceGraphHolder">
                            <canvas id="closePriceGraph"></canvas>
                        </div>
                    </div>
                    <button class="startTraining">Start
Training</button>
                </div>
            </center>
            <center>
                <div class="container-fluid trainingProgress">
                    <div class="placeholder">
                        <div class="layer">
                            <div class="layerName">Training</div>
                            <div class="layerDesc">Click on "Start
Training" button to start the training</div>
                        </div>
                    </div>
                    <div class="inProgress">
                        <div class="layer">
                            <div class="layerName">Training</div>
                            <img src="../icons/loading.svg" />
                            <div class="layerDesc">Training</div>
                        </div>
                    </div>
                    <div class="trainingCompleted">
                        <div class="layer">
                            <div class="layerName">Training
Completed</div>
                            <div class="layerDesc">Now you can see
the predictions in "Predictions" tab</div>
                        </div>
                    </div>
                </div>
            </center>

            <script>

                var checkTrainingStatusVar = null;

                function isInt(n){
                    return Number(n) === n && n % 1 === 0;
                }

                function isFloat(n){
                    return Number(n) === n && n % 1 !== 0;
                }

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

                    var body = `<tbody>`;
                    for (var row in tableData) {
                        var rowHTML = `<tr>`;
                        for (col in tableData[row]) {
                            if (isFloat(tableData[row][col]) ||
isInt(tableData[row][col])) rowHTML += `<td>` +
tableData[row][col].toFixed(2) + `</td>`;
                            else rowHTML += `<td>` +
tableData[row][col] + `</td>`;
                        }
                        rowHTML += `</tr>`;
                        body += rowHTML;
                    }
                    body += `</tbody>`;

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

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

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

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

                    const config = {
                        type: 'line',
                        data: {
                            datasets: [{
                                borderColor: "#3aa4eb",
                                borderWidth: 1,
                                radius: 0,
                                data: data,
                            }]
                        },
                        options: {
                            animation,
                            interaction: {
                                intersect: false
                            },
                            plugins: {
```

```
                            legend: false
                        },
                        scales: {
                            x: {
                                type: 'category',
                                labels: dfDateVals
                            }
                        }
                    }
                };

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

                return myChart;

            }

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

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

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

            function checkTrainingStatus() {
                checkTrainingStatusVar = setInterval(() => {
                    $.ajax({
                        url:
'http://localhost:7676/trainingStatus',
                        method: 'post',
                        success: (res) => {
                            console.log(res);
                            $('.trainingProgress .inProgress
.layerDesc').html("Training " +
Math.round((res.epochs/res.totalEpochs)*100) + "%");
                            if (res.status != "training") {
                                stopTrainingStatusCheck();
                            }
                        }
                    });
                }, 1000);
            }

            function stopTrainingStatusCheck() {
                clearInterval(checkTrainingStatusVar);

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

            $('.selectFile').change(() => {
                var file = $('.selectFile')[0].files[0];
                var extension =
file.name.substring(file.name.lastIndexOf(".")+1);

                if (extension != "csv") {
                    alert("Only CSV files are accepted as
training data");
                    return;
                }

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

                $.ajax({
                    url: 'http://localhost:7676/upload',
                    type: "POST",
                    processData: false,
                    contentType: false,
                    data: formData,
                    success: (res) => {
                        console.log(res);
                        loadDataset(res);
                    },
                    error: (res)=> {
                        // console.log(String(e.responseText))
                        //
console.log(JSON.parse(String(e.responseText)));
                        loadDataset(res);
                    }
                });
            });
            $('.startTraining').click(() => {
                $('.trainingProgress .placeholder').hide();
                $('.trainingProgress
.trainingCompleted').hide();

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

                    }
                });

                checkTrainingStatus();
            });

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

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

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

        </script>
    </div>
</body>
</html>
```

Code for training.html

## Code of training.css file

```css
.upload {
    text-align: center;
}

.upload button {
    border: none;
    outline: none;
    border-radius: 7px;
    background-color: rgb(67, 142, 255);
    color: white;
    padding: 5px;
}

.upload .selectFile {
    display: none;
}

.datasetProperties {
    position: relative;
    margin-top: 20px;
    width: 80%;
    height: 330px;
    box-shadow: rgba(50, 50, 93, 0.25) 0px 6px 12px -2px, rgba(0,
0, 0, 0.3) 0px 3px 7px -3px;
    padding: 0;
}

.datasetProperties .filename {
    margin-top: 15px;
}

.datasetProperties .filename .fileNameInput {
    border: none;
    outline: none;
    border-bottom: 1px solid;
    text-align: center;
}

.datasetProperties .dfHead {
    text-align: center;
    margin-top: 20px;
    width: 95%;
    height: 200px;
    overflow-y: scroll;
}

.datasetProperties .dfHead .table {
    font-size: 12px;
    height: 100px;
}

.datasetProperties .dfHead .table thead th {
    position: sticky;
    top: 0;
    background-color: white;
}

.datasetProperties .dfHead::-webkit-scrollbar {
    display: none;
}
#closePriceGraph {
    margin-top: 20px;
}

.datasetProperties .startTraining {
    border: none;
    outline: none;
    border-radius: 7px;
    background-color: rgb(123, 67, 255);
    color: white;
    padding: 5px;
    margin-top: 10px;
}

.datasetProperties .placeholder {
    position: absolute;
    background-color: white;
    height: 100%;
    width: 100%;
    z-index: 2;
}

.datasetProperties .placeholder .layer {
    position: absolute;
    background-color: rgba(0, 0, 0, 0.8);
    height: 100%;
    width: 100%;
    z-index: 2;
```

```
}

.datasetProperties .placeholder .layerName {
    color: white;
    font-size: 25px;
    margin-top: 110px;
}

.datasetProperties .placeholder .layerDesc {
    color: rgba(255, 255, 255, 0.7);
    font-size: 16px;
    margin-top: 10px;
}

.trainingProgress {
    position: relative;
    margin-top: 20px;
    width: 80%;
    height: 330px;
    box-shadow: rgba(50, 50, 93, 0.25) 0px 6px 12px -2px, rgba(0,
0, 0, 0.3) 0px 3px 7px -3px;
    padding: 0;
}

.trainingProgress .placeholder {
    position: absolute;
    background-color: white;
    height: 100%;
    width: 100%;
    z-index: 3;
}

.trainingProgress .placeholder .layer {
    position: absolute;
    background-color: rgba(0, 0, 0, 0.8);
    height: 100%;
    width: 100%;
    z-index: 3;
}

.trainingProgress .placeholder .layerName {
    color: white;
    font-size: 25px;
    margin-top: 110px;
}
```

```
.trainingProgress .placeholder .layerDesc {
    color: rgba(255, 255, 255, 0.7);
    font-size: 16px;
    margin-top: 10px;
}

.trainingProgress.inProgress {
    position: absolute;
    background-color: white;
    height: 100%;
    width: 100%;
    z-index: 2;
}

.trainingProgress .inProgress .layer {
    position: absolute;
    background-color: rgba(0, 0, 0, 0.8);
    height: 100%;
    width: 100%;
    z-index: 2;
}

.trainingProgress .inProgress .layerName {
    color: white;
    font-size: 25px;
    margin-top: 110px;
}

.trainingProgress .inProgress .layerDesc {
    color: rgba(255, 255, 255, 0.7);
    font-size: 16px;
    margin-top: 10px;
}

.trainingProgress.trainingCompleted {
    position: absolute;
    background-color: white;
    height: 100%;
    width: 100%;
    z-index: 2;
}

.trainingProgress .trainingCompleted .layer {
    position: absolute;
    background-color: rgba(0, 0, 0, 0.8);
    height: 100%;
    width: 100%;
    z-index: 2;
}

.trainingProgress .trainingCompleted .layerName {
    color: white;
    font-size: 25px;
    margin-top: 110px;
}

.trainingProgress .trainingCompleted .layerDesc {
    color: rgba(255, 255, 255, 0.7);
    font-size: 16px;
    margin-top: 10px;
}
```

Code for training.css

## Code of prediction.html file

```
<html>

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

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

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

    <script src="../static/chart.js"></script>
    <script src="../static/table2CSV.js"></script>

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

    <script>

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

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

    </script>
</head>

<body>
    <div id="header">
        <div class="container-fluid">
            <div class="row">
                <div class="col-sm-4 header-title">Stock Price 
Prediction</div>
                <div class="col-sm-8">
```

```
                    <div class="navbar">
                        <div onclick="changePage(this)"
page="home" class="nav-item">Home</div>
                        <div onclick="changePage(this)"
page="training" class="nav-item">Training</div>
                        <div onclick="changePage(this)"
page="predictions" class="nav-item active">Predictions</div>
                    </div>
                </div>
            </div>
        </div>
    </div>

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

        <div class="predictionsPage">
            <center>
                <div class="btn-group selectModel">
                    <button type="button" data-
toggle="dropdown">Select Model</button>
                    <div class="dropdown-menu dropdown-menu-
center"></div>
                </div>
            </center>

            <center>
                <div class="container-fluid predictionResults">
                    <div class="placeholder">
                        <div class="layer">
                            <div class="layerName">Model</div>
                            <div class="layerDesc">Select the
model by clicking "Select Model" button to show
                                predictions.</div>
                        </div>
                    </div>
                    <div class="inProgress">
                        <div class="layer">
                            <div class="layerName">Model</div>
                            <img src="../icons/loading.svg" />
                            <div class="layerDesc">Loading model
from server</div>
                        </div>
                    </div>
                    <div class="row">
                        <div class="modelName"></div>
                        <div class="col-sm-8
predictionGraphHolder"><canvas
id="predictionGraph"></canvas></div>
                        <div class="col-sm-4">
                            <div class="accuracyPercentage"></div>
                            <div class="realPredTable"></div>
                            <button
class="downloadButton">Download</button>
                        </div>
                    </div>
                </div>
            </center>
            <center>
                <button class="predictNextDayButton">Predict Next
Day</button>
                <div class="container-fluid predictNextDayDiv">
                    <div class="row">
                        <div class="col-sm-1"></div>
                        <div class="col-sm-2">
                            <div class="input-text">Open</div>
                            <input type="text" class="input-value
openValue" />
                        </div>
                        <div class="col-sm-2">
                            <div class="input-text">High</div>
                            <input type="text" class="input-value
highValue" />
                        </div>
                        <div class="col-sm-2">
                            <div class="input-text">Low</div>
                            <input type="text" class="input-value
lowValue" />
                        </div>
                        <div class="col-sm-2">
                            <div class="input-text">Volume</div>
                            <input type="text" class="input-value
volumeValue" />
                        </div>
                        <div class="col-sm-2">
                            <div class="input-text">Action</div>
                            <button
class="startManualPrediction">Predict Next Day</button>
                        </div>
                        <div class="col-sm-1"></div>
                    </div>
                    <div class="manualPrediction"></div>
                </div>
            </center>
        </div>

        <script>
            function plotGraph(chartId, trainingDates, 
trainingData, predictionDates, predictionData,realData, 
showAnimation) {
                var training = [];
                var predictions = [];
                var animation = false;



                var count=0;
                for (let i=0; i < trainingData.length; i++, 
count++) {
                    training.push(trainingData[i]);
                    predictions.push(NaN);
                }

                predictions[training.length  - 1] = 
training[training.length - 1];

                for (let i=0; i < predictionData.length; i++, 
count++) {
                    training.push(realData[i]);
                    predictions.push(predictionData[i]);
                }

                if (showAnimation) {
                    const totalDuration = 7600;
                    const delayBetweenPoints = totalDuration / 
(training.length + predictions.length);
                    const previousY = (ctx) => ctx.index === 0 ? 
ctx.chart.scales.y.getPixelForValue(100) : 
ctx.chart.getDatasetMeta(ctx.datasetIndex).data[ctx.index - 
1].getProps(['y'], true).y;
                    animation = {
                        x: {
                            type: 'number',
                            easing: 'linear',
                            duration: delayBetweenPoints,
                            from: NaN, // the point is initially 
skipped
                            delay(ctx) {
                                if (ctx.type !== 'data' ||
ctx.xStarted) {
                                    return 0;
                                }
                                ctx.xStarted = true;
                                return ctx.index *
delayBetweenPoints;
                            }
                        },
                        y: {
                            type: 'number',
                            easing: 'linear',
                            duration: delayBetweenPoints,
                            from: previousY,
                            delay(ctx) {
                                if (ctx.type !== 'data' ||
ctx.yStarted) {
                                    return 0;
                                }
                                ctx.yStarted = true;
                                return ctx.index *
delayBetweenPoints;
                            }
                        }
                    };
                }

                const config = {
                    type: 'line',
                    data: {
                        datasets: [
                        {
                            borderColor: "#ff7b00",
                            borderWidth: 1,
                            radius: 0,
                            data: predictions,
                        }
                        ,{
                            borderColor: "#3aa4eb",
                            borderWidth: 1,
                            radius: 0,
                            data: training,
                        }],

                    },
                    options: {
                        animation,
                        interaction: {
                            intersect: true
                        },
                        plugins: {
                            legend: false
                        },
                        scales: {
                            x: {
                                type: 'category',
                                labels: 
trainingDates.concat(predictionDates)
                            }
                        }
                    }
                };

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

                return myChart;

            }

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

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

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

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

```
                        $('.realPredTable').attr("table-name",
res.modelName);

                        $('.inProgress').hide();
                        $('.downloadButton').show();
                        $('.predictNextDayButton').show();

                        plotGraph(
                            'predictionGraph',
                            res.modelData.training.dates,
                            res.modelData.training.data,
                            res.modelData.predictions.dates,
                            res.modelData.predictions.predictedDat
a,
                            res.modelData.predictions.realData,
                            true
                        );
                        console.log(res);
                        var dates =
res.modelData.predictions.dates;
                        var realData =
res.modelData.predictions.realData;
                        var predictedData =
res.modelData.predictions.predictedData;

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

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

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

```
                                </tr>
                                `;
                        }
                        body += `</tbody>`;

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

                        $('.accuracyPercentage').html("Accuracy: "
+ res.modelData.predictions.accuracy.toFixed(2) + " %");
                    }
                });
            }

            $('.downloadButton').click(function () {
                var table = $('.realPredTable').first();
                var tableName = $('.realPredTable').attr("table-
name");

                var csv = $(table).table2CSV({
                    delivery: 'value'
                });

                var link = document.createElement('a');
                link.href = 'data:text/csv;charset=UTF-8,' +
encodeURIComponent(csv);
                link.download = tableName + " Results.csv";
                link.click();
            });

            $('.predictNextDayButton').click(() => {
                $('.predictNextDayDiv').show();
            });

            $('.startManualPrediction').click(() => {
                var fileName = $('.realPredTable').attr("table-
name");

                var openValue = $('.openValue').val();
                var highValue = $('.highValue').val();
                var lowValue = $('.lowValue').val();
                var volumeValue = $('.volumeValue').val();
                if ((openValue == "") || (highValue == "") ||
(lowValue == "") || (volumeValue == "")) {
                    alert("Please fill all the details");
                    return;
                }

                $.ajax({
                    url:
'http://localhost:7676/getManualPrediction',
                    method: 'post',
                    data: {
                        "fileName": fileName,
                        "openValue": openValue,
                        "highValue": highValue,
                        "lowValue": lowValue,
                        "volumeValue": volumeValue
                    },
                    success: (res) => {
                        console.log(res);

                        $('.manualPrediction').html('Predicted: '
+ res.manualPrediction);

                        var openValue = $('.openValue').val("");
                        var highValue = $('.highValue').val("");
                        var lowValue = $('.lowValue').val("");
                        var volumeValue =
$('.volumeValue').val("");
                    }
                });
            });

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

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

                        var preTrainedModelNames =
res.preTrainedModelNames;
```

```
                        var options = ``;

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

                        $('.selectModel .dropdown-
menu').html(options);
                    }
                });
            });

        </script>
    </div>
</body>

</html>
```

Code for prediction.html

## Code of prediction.css file

```
.selectModel {
    text-align: center;
}

.selectModel button {
    border: none;
    outline: none;
    border-radius: 7px;
    background-color: rgb(67, 142, 255);
    color: white;
    padding: 5px;
}

.selectModel .dropdown-item {
    display: block;
    text-decoration: none;
    padding: 5px;
}

.selectModel .dropdown-menu-center {
    right: auto;
    left: 50%;
    -webkit-transform: translate(-50%, 0);
    -o-transform: translate(-50%, 0);
    transform: translate(-50%, 0);
}

.selectModel .dropdown-menu {
    border: none;
    box-shadow: rgba(50, 50, 93, 0.25) 0px 6px 12px -2px, rgba(0,
0, 0, 0.3) 0px 3px 7px -3px;
}

.selectModel .dropdown-menu a {
    color: black;
}

.selectModel .dropdown-menu a:hover {
    background-color: rgb(228, 228, 228);
}

.predictionResults {
    width: 95%;
    position: relative;
    margin-top: 20px;
    min-height: 330px;
    box-shadow: rgba(50, 50, 93, 0.25) 0px 6px 12px -2px, rgba(0,
0, 0, 0.3) 0px 3px 7px -3px;
    padding: 0;
}

.predictionResults .modelName {
    font-size: 18px;
}

.predictionResults .placeholder {
    position: absolute;
    background-color: white;
    height: 100%;
    width: 100%;
    z-index: 2;
}

.predictionResults .placeholder .layer {
    position: absolute;
    background-color: rgba(0, 0, 0, 0.8);
    height: 100%;
    width: 100%;
    z-index: 2;
}

.predictionResults .placeholder .layerName {
    color: white;
    font-size: 25px;
    margin-top: 110px;
}

.predictionResults .placeholder .layerDesc {
    color: rgba(255, 255, 255, 0.7);
    font-size: 16px;
    margin-top: 10px;
}

.predictionResults.inProgress {
    position: absolute;
    background-color: white;
    height: 100%;
    width: 100%;
    z-index: 2;
```

```
}

.predictionResults .inProgress .layer {
    position: absolute;
    background-color: rgba(0, 0, 0, 0.8);
    height: 100%;
    width: 100%;
    z-index: 2;
}

.predictionResults .inProgress .layerName {
    color: white;
    font-size: 25px;
    margin-top: 110px;
}

.predictionResults .inProgress .layerDesc {
    color: rgba(255, 255, 255, 0.7);
    font-size: 16px;
    margin-top: 10px;
}

.predictionResults .realPredTable {
    text-align: center;
    margin-top: 20px;
    width: 95%;
    height: 330px;
    overflow-y: scroll;
}

.predictionResults .realPredTable .table {
    font-size: 12px;
    height: 100px;
}

.predictionResults .realPredTable .table thead th {
    position: sticky;
    top: 0;
    background-color: white;
}

.predictionResults .realPredTable::-webkit-scrollbar {
    display: none;
}

.downloadButton {
    border: none;
    outline: none;
    border-radius: 7px;
    background-color: rgb(67, 142, 255);
    color: white;
    padding: 5px;
    margin-top: 10px;
}

.predictNextDayButton {
    margin-top: 30px;
    margin-bottom: 20px;

    border: none;
    outline: none;
    border-radius: 7px;
    background-color: rgb(67, 142, 255);
    color: white;
    padding: 5px;
}

.predictNextDayDiv .startManualPrediction {
    border: none;
    outline: none;
    border-radius: 7px;
    background-color: rgb(67, 142, 255);
    color: white;
    padding: 5px;
}

.predictNextDayDiv .manualPrediction {
    margin-top: 20px;
    font-size: 18px;
}
```

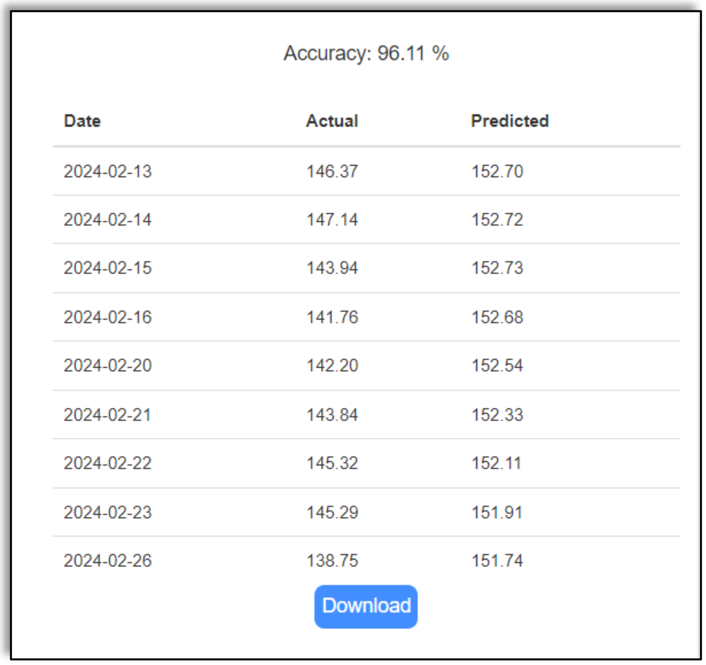
Code for prediction.css

# CHAPTER 6 – RESULT AND CONCLUSION

## 6.1 LSTM Performance

The performance of the Long Short-Term Memory (LSTM) model was evaluated using several key metrics, including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). The following results were obtained:

- **Mean Squared Error (MSE):** The MSE for the model was calculated to be 0.0005, indicating a small average squared difference between the predicted and actual stock prices.
- **Root Mean Squared Error (RMSE):** The RMSE was found to be 0.0224, providing an interpretable measure of the average prediction error. This low value suggests high accuracy in the model's predictions.
- **Mean Absolute Error (MAE):** The MAE was determined to be 0.0158, highlighting the average absolute difference between the predicted and actual values.



Accuracy: 96.11 %

| Date | Actual | Predicted |
|---|---|---|
| 2024-02-13 | 146.37 | 152.70 |
| 2024-02-14 | 147.14 | 152.72 |
| 2024-02-15 | 143.94 | 152.73 |
| 2024-02-16 | 141.76 | 152.68 |
| 2024-02-20 | 142.20 | 152.54 |
| 2024-02-21 | 143.84 | 152.33 |
| 2024-02-22 | 145.32 | 152.11 |
| 2024-02-23 | 145.29 | 151.91 |
| 2024-02-26 | 138.75 | 151.74 |

*Table 2 Accuracy of Google's stock*

Table below shows the actual and predicted price of Google's stock on specified date. It also shows the accuracy in terms of percentage as it is easy for the client to understand as compare to RMSE, MSE etc.

There is a download button which facilitates to save the table in csv file for further analysis.

A visual comparison between the actual and predicted stock prices was performed to qualitatively assess the model's performance. The plot below shows both the actual stock prices and the predicted prices over time:

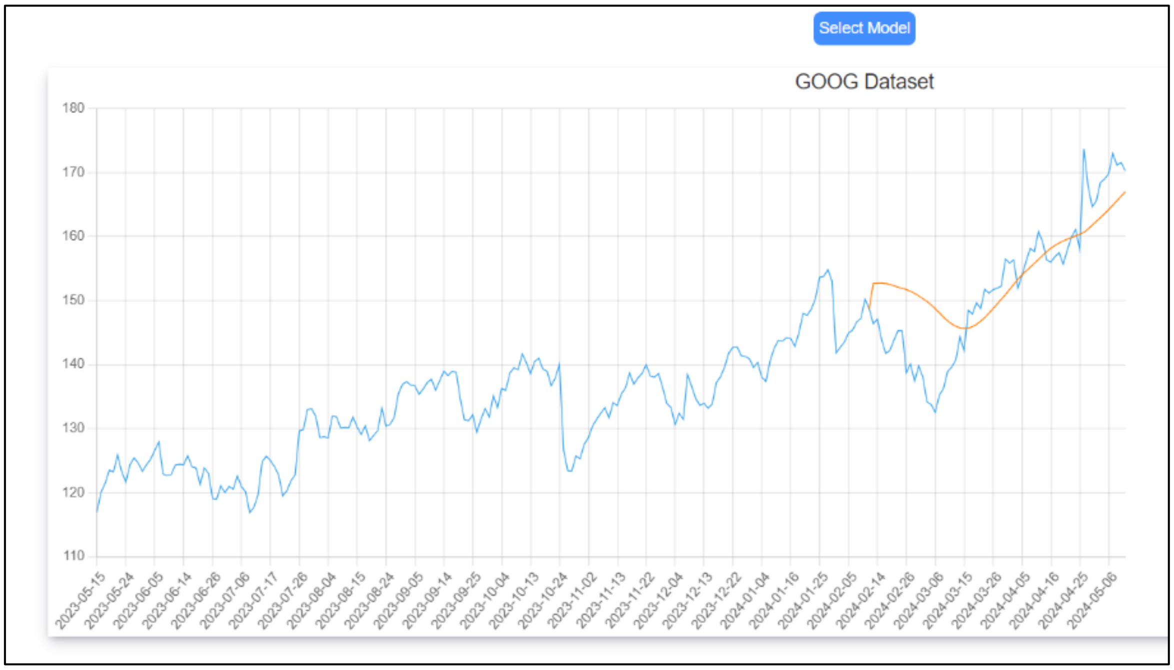


*Figure 16 Google's stock visual accuracy graph*

As seen from the plot, the model's predictions closely follow the trend of the actual stock prices, demonstrating its ability to capture the underlying patterns in the data effectively. An additional feature of predicting next day stock price is given in the model. User can input open, high, low and volume to predict the next day closing price.

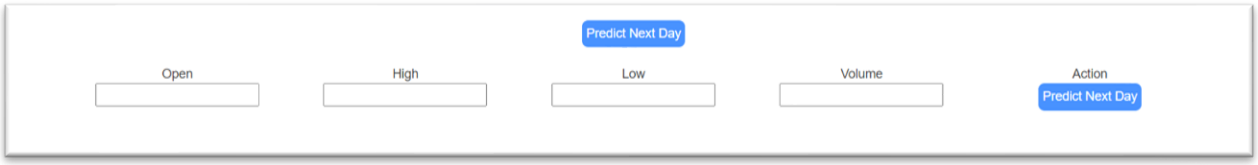


*Figure 17 interface for predicting next day closing price*

In Google’s stock price example, more than 96% accuracy was achieved, indicating that the model can reliably forecast future stock prices with a higher degree of precision.

## 6.2 Conclusion

The LSTM model developed for stock price prediction demonstrated strong performance, as evidenced by both quantitative metrics and visual inspection of the results. Key conclusions from the project are as follows:

1. **High Accuracy:** The model achieved a high accuracy of 96.11%, with low values of MSE, RMSE, and MAE, suggesting that it can effectively predict stock prices.
2. **Robust Performance:** The close alignment between the actual and predicted stock prices in the visual comparison indicates that the model captures the essential patterns in the stock price movements.
3. **Scalability:** The methodology used for data preprocessing, model training, and evaluation is scalable to different datasets and can be adapted for various stock prediction tasks.

## 6.3 Future Scope

The field of stock price prediction using machine learning and deep learning models is vast and constantly evolving. This project has demonstrated the application of LSTM (Long Short-Term Memory) networks for predicting stock prices, but there are several avenues for future research and improvement:

- **Enhanced Data Preprocessing**:
  - .1. **Feature Engineering**: Incorporating additional features such as sentiment analysis from news articles, social media feeds, and macroeconomic indicators could improve prediction accuracy.
  - .2. **Anomaly Detection**: Implementing techniques to detect and handle anomalies in the dataset could enhance model robustness.
- **Advanced Model Architectures**:
  - .1. **Hybrid Models**: Combining LSTM networks with other models like ARIMA (AutoRegressive Integrated Moving Average) or GARCH (Generalized

Autoregressive Conditional Heteroskedasticity) can leverage the strengths of both approaches.

.2. **Attention Mechanisms**: Incorporating attention mechanisms can help the model focus on the most relevant parts of the input data, potentially improving performance.

- **Transfer Learning**:

  .1. Utilizing pre-trained models from related domains and fine-tuning them on stock price data can reduce training time and improve prediction accuracy.

- **Hyperparameter Tuning**:

  .1. Systematic hyperparameter tuning using techniques like grid search, random search, or Bayesian optimization can help find the optimal settings for model training, leading to better performance.

- **Real-Time Prediction**:

  .1. Developing systems that can make real-time predictions based on live data streams would have significant practical applications in financial trading and investment strategies.

# CHAPTER 7 – SNAPSHOTS

## 7.1 Home Page:

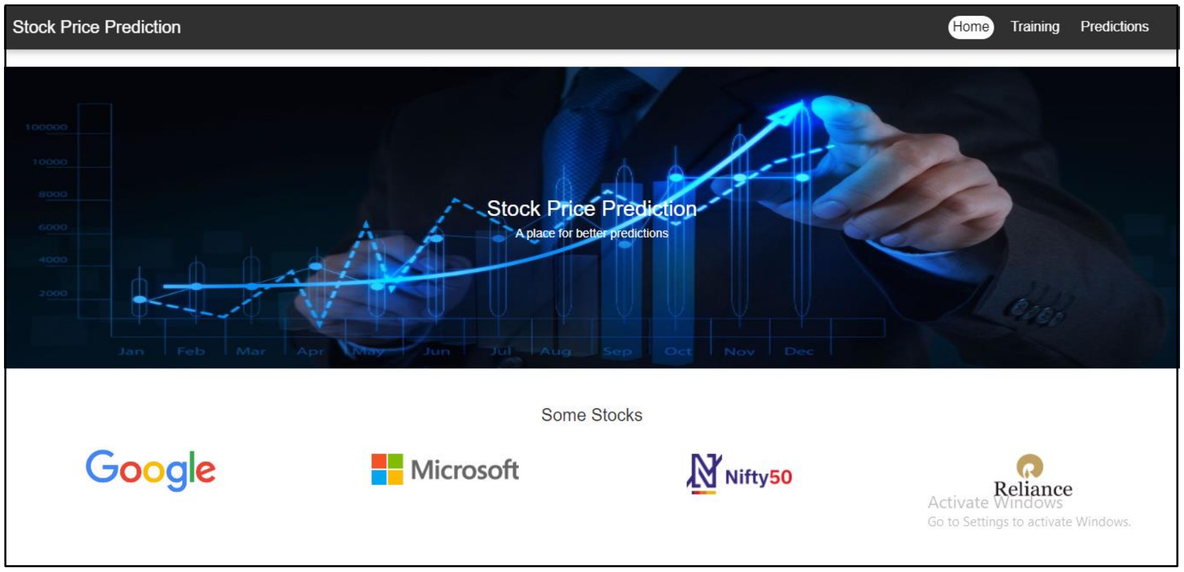
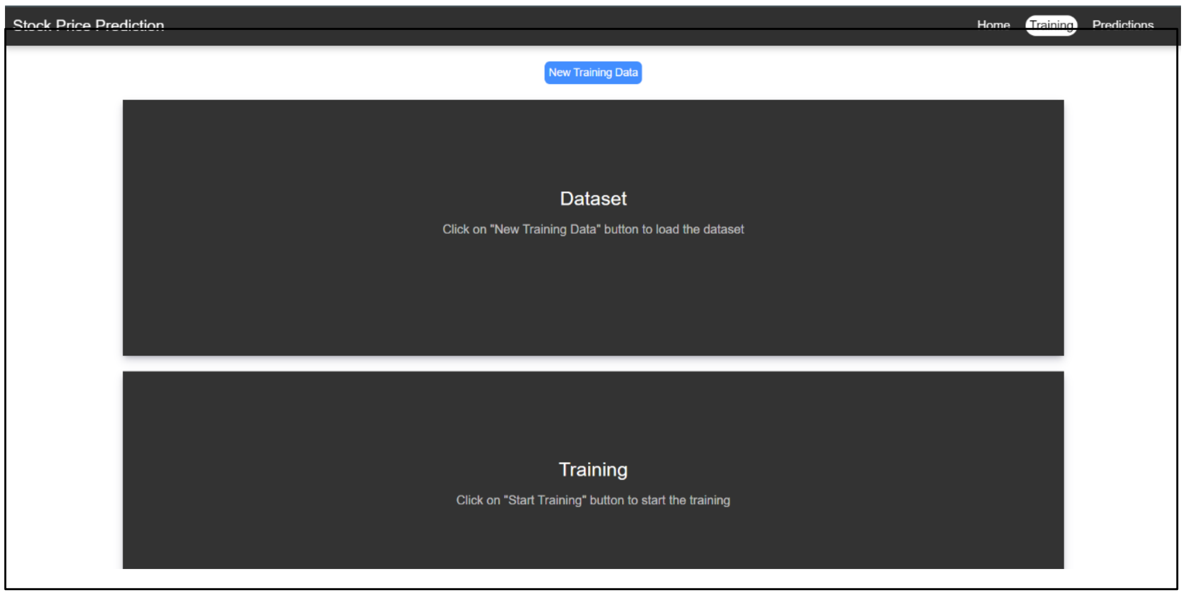


*Figure 18 Home Page*

## 7.2 Training Page:



Dataset

Click on "New Training Data" button to load the dataset

Training

Click on "Start Training" button to start the training

*Figure 19 Training page*

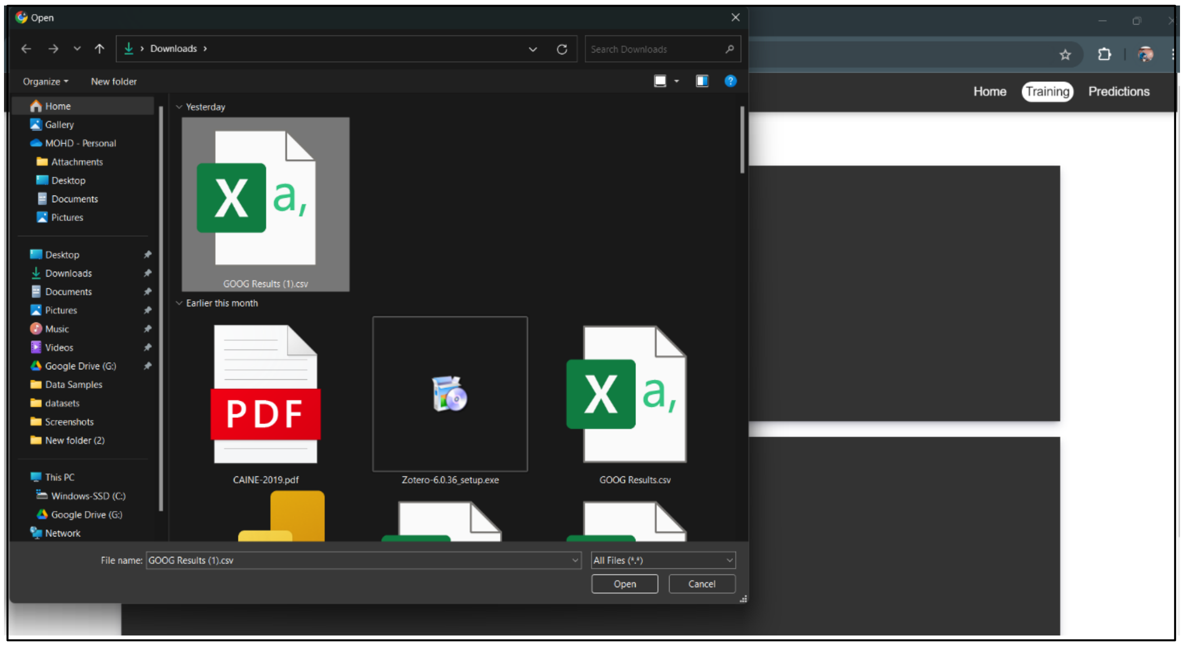
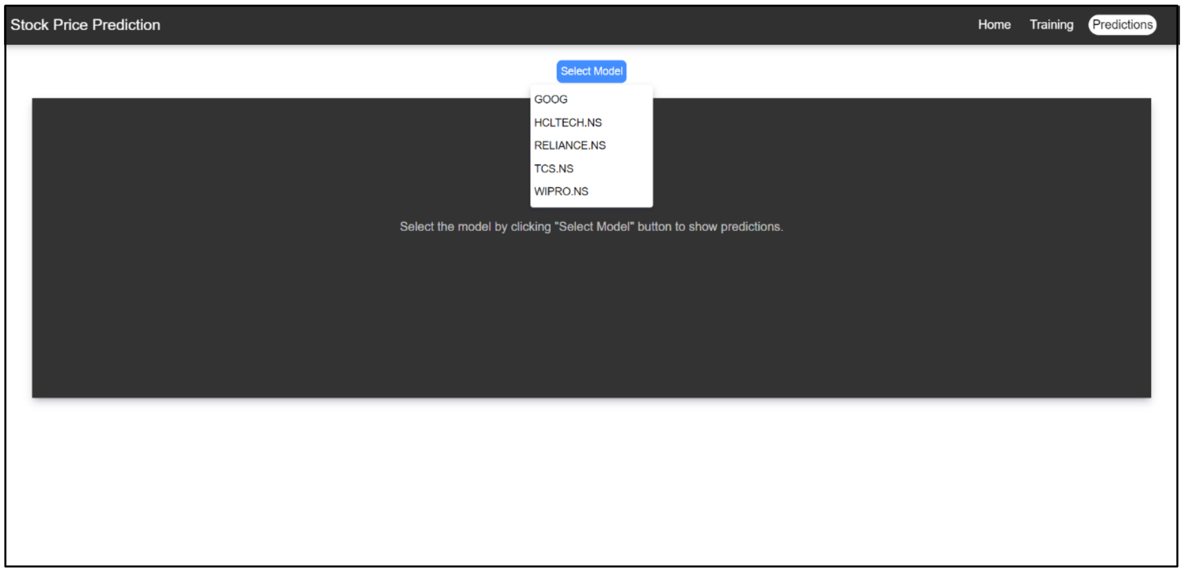


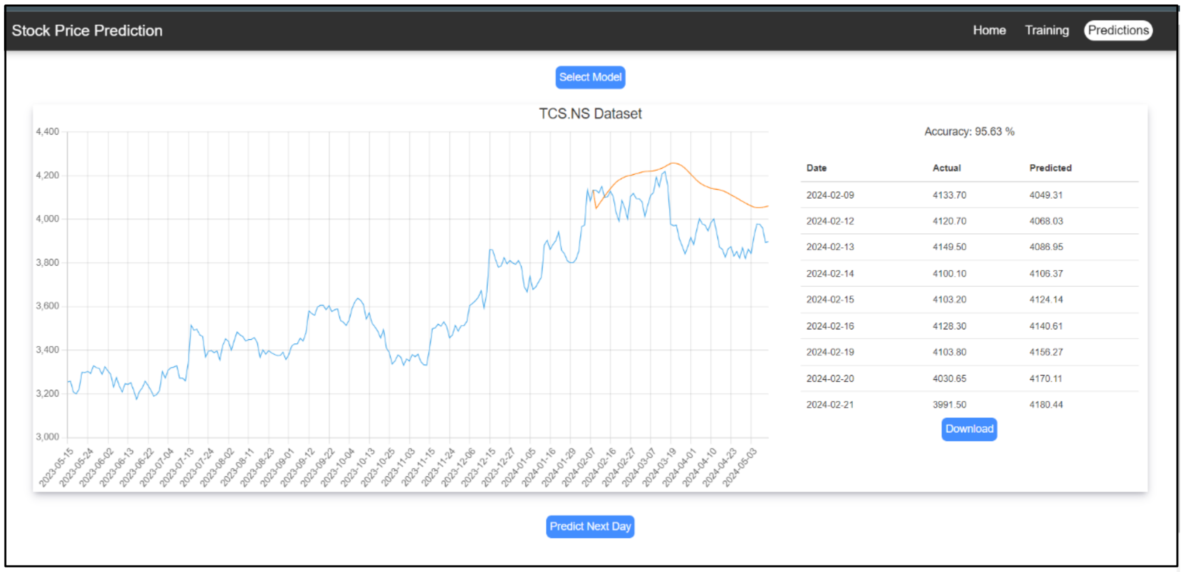
*Figure 20 Selecting dataset from local drive*

## 7.3 Prediction Page:



Select the model by clicking "Select Model" button to show predictions.

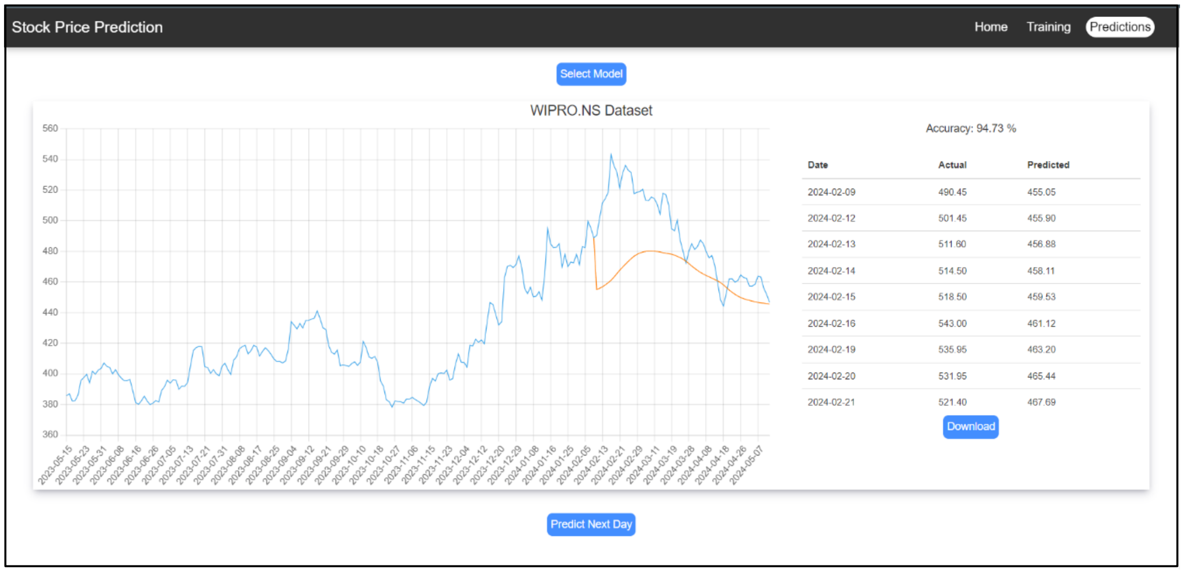
*Figure 21 Prediction page*



Accuracy: 95.63 %

| Date | Actual | Predicted |
|---|---|---|
| 2024-02-09 | 4133.70 | 4049.31 |
| 2024-02-12 | 4120.70 | 4068.03 |
| 2024-02-13 | 4149.50 | 4086.95 |
| 2024-02-14 | 4100.10 | 4106.37 |
| 2024-02-15 | 4103.20 | 4124.14 |
| 2024-02-16 | 4128.30 | 4140.61 |
| 2024-02-19 | 4103.80 | 4156.27 |
| 2024-02-20 | 4030.65 | 4170.11 |
| 2024-02-21 | 3991.50 | 4180.44 |

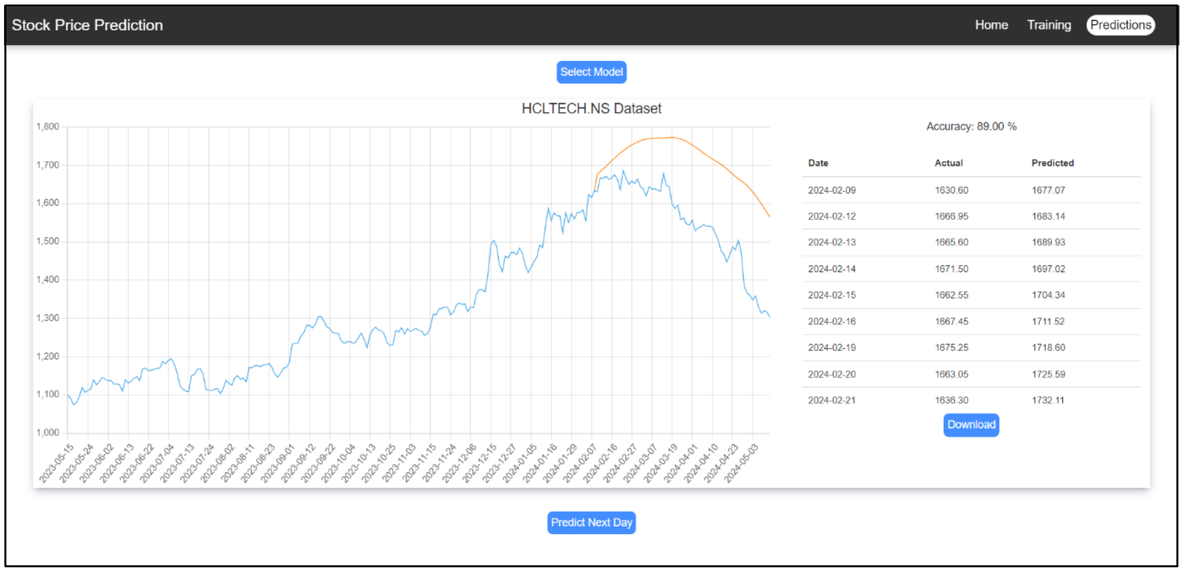
*Figure 22 Prediction of TCS.NS stock*



Accuracy: 94.73 %

| Date | Actual | Predicted |
|---|---|---|
| 2024-02-09 | 490.45 | 455.05 |
| 2024-02-12 | 501.45 | 455.90 |
| 2024-02-13 | 511.60 | 456.88 |
| 2024-02-14 | 514.50 | 458.11 |
| 2024-02-15 | 518.50 | 459.53 |
| 2024-02-16 | 543.00 | 461.12 |
| 2024-02-19 | 535.95 | 463.20 |
| 2024-02-20 | 531.95 | 465.44 |
| 2024-02-21 | 521.40 | 467.69 |

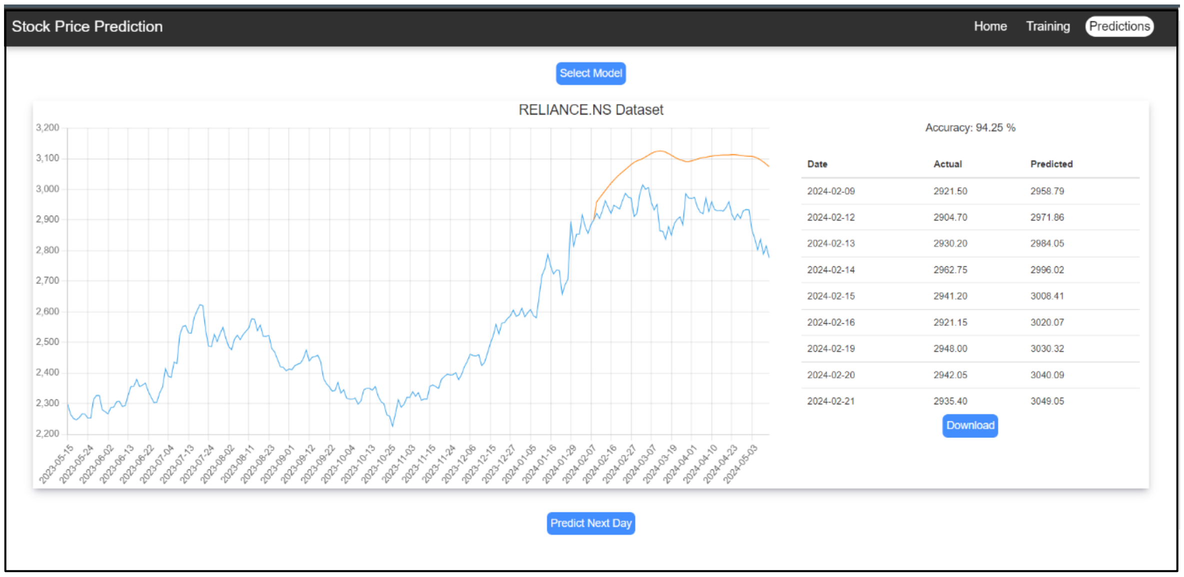
*Figure 23 Prediction of Wipro.NS stock*



Accuracy: 89.00 %

| Date | Actual | Predicted |
|---|---|---|
| 2024-02-09 | 1630.60 | 1677.07 |
| 2024-02-12 | 1666.95 | 1683.14 |
| 2024-02-13 | 1665.60 | 1689.93 |
| 2024-02-14 | 1671.50 | 1697.02 |
| 2024-02-15 | 1662.55 | 1704.34 |
| 2024-02-16 | 1667.45 | 1711.52 |
| 2024-02-19 | 1675.25 | 1718.60 |
| 2024-02-20 | 1663.05 | 1725.59 |
| 2024-02-21 | 1636.30 | 1732.11 |

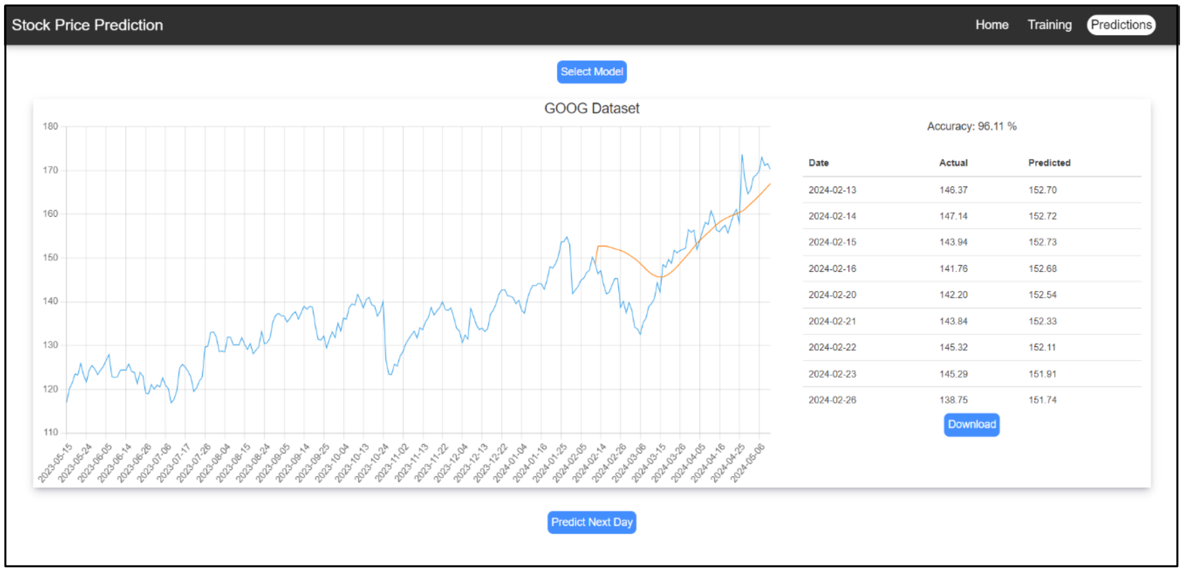
*Figure 24 Prediction of HCLTECH.NS*



Accuracy: 94.25 %

| Date | Actual | Predicted |
|---|---|---|
| 2024-02-09 | 2921.50 | 2958.79 |
| 2024-02-12 | 2904.70 | 2971.86 |
| 2024-02-13 | 2930.20 | 2984.05 |
| 2024-02-14 | 2962.75 | 2996.02 |
| 2024-02-15 | 2941.20 | 3008.41 |
| 2024-02-16 | 2921.15 | 3020.07 |
| 2024-02-19 | 2948.00 | 3030.32 |
| 2024-02-20 | 2942.05 | 3040.09 |
| 2024-02-21 | 2935.40 | 3049.05 |

*Figure 25 Prediction of Reliance.NS stock*



Accuracy: 96.11 %

| Date | Actual | Predicted |
|---|---|---|
| 2024-02-13 | 146.37 | 152.70 |
| 2024-02-14 | 147.14 | 152.72 |
| 2024-02-15 | 143.94 | 152.73 |
| 2024-02-16 | 141.76 | 152.68 |
| 2024-02-20 | 142.20 | 152.54 |
| 2024-02-21 | 143.84 | 152.33 |
| 2024-02-22 | 145.32 | 152.11 |
| 2024-02-23 | 145.29 | 151.91 |
| 2024-02-26 | 138.75 | 151.74 |

*Figure 26 Prediction of Google stock*

# CHAPTER 8 - REFERENCES

1. Chouksey, S. & Hoare, T. STOCK PRICE PREDICTION USING TIME SERIES MODELS. (2018).

2. Chen, C., Xue, L. & Xing, W. Research on Improved GRU-Based Stock Price Prediction Method. *Applied Sciences* **13**, 8813 (2023).

3. Moghar, A. & Hamiche, M. Stock Market Prediction Using LSTM Recurrent Neural Network. *Procedia Computer Science* **170**, 1168–1173 (2020).

4. Leung, C. K.-S., MacKinnon, R. K. & Wang, Y. A machine learning approach for stock price prediction. in *Proceedings of the 18th International Database Engineering & Applications Symposium on - IDEAS '14* 274–277 (ACM Press, Porto, Portugal, 2014). doi:10.1145/2628194.2628211.

5. Shen, J. & Shafiq, M. O. Short-term stock market price trend prediction using a comprehensive deep learning system. *J Big Data* **7**, 66 (2020).

6. Surayagari, H. K. S. Stock Market Predictions Using Machine Learning. (2021).

7. Roondiwala, M., Patel, H. & Varma, S. Predicting Stock Prices Using LSTM. **6**, (2017).

8. Siami-Namini, S. & Namin, A. S. Forecasting Economics and Financial Time Series: ARIMA vs. LSTM. Preprint at http://arxiv.org/abs/1803.06386 (2018).

9. Giusti et al., ". M.-6. (2015).
10. Dematos, G. B.-7. (2007).
11. Ding, X. Z.-d.-f. (2015).
12. Eyben, F. W.-u.-3. (2009).
13. Nowrouz Kohzadi, M. S. (1996).
14. Roondiwala, M. P. (2017).
15. S. Hochreiter and J. Schmidhuber, ". S.-T.-1. (1997).
16. Tarwani, K. M.-3. (2017).