Real Time Stock Monitoring Dashboard and Forecasting Stock Prices

Sankha Patra
19BEC0864

Sona Susan Jacob
19BEC0479

Bollam Vamsi
19BEC0256

Abstract – In the past few years, more people of India have started investing in the stock market. Therefore, a platform is needed where we can track the stock prices and monitor them on a regular basis, as well as learn the market trends and take decisions based on these trends. Recently, people are getting interested in forecasting stock prices and gain profit through these predicted prices. In this project, we've developed a dashboard for monitoring stock prices in real time and an LSTM model to forecast stock prices and compared its performance with some machine learning algorithms.

Keyword – Indian Stock Market, National Stock Exchange, LSTM Neural Networks, Stock Market Dashboard, Web Scraping.

1. INTRODUCTION

Stocks of a company represent ownership equity in the firm. The people who hold these stocks or shares, called shareholders, get a claim on corporate earnings in the form of capital gains and dividends. A stock market is a platform for trading of a company's stocks and derivatives at an agreed price. [1] The supply and demand of these stocks drives the stock market and they're responsible for the increment or decrement of stock prices of companies.

In India, stock market is one of the most emerging sectors and many people in our country are directly or indirectly related to this sector. Therefore, it has become essential for people to know about the stock market trends. With the development in stock markets, traders are also getting interested in forecasting stock prices and gaining profits through these prices.

Forecasting stock prices of a company using historical stock prices is a challenging task as there are many variable factors affecting the actual stock price, but an intuition or insight can be taken from forecasting and use it to make decisions on whether to hold the stocks for longer or sell them to gain some profit.

In this project, we have built two interactive dashboards, one for analysing historical stock market trends and the other for monitoring real time stock price which is helpful for intraday traders. In addition to these two dashboards, we've worked on predicting future stock prices. There are two approaches to predict stock prices: technical and fundamental analysis. Technical analysis tries to identify some patterns from historical data while fundamental analysis focuses on overall economy, the company's financial condition and its management. [2]

For this project, we have taken the first approach i.e., technical analysis approach where we've built an LSTM model to predict future stock prices for the major oil and natural gas industry GAIL listed in the National Stock Exchange (NSE) of India and compared the performance of our LSTM model with some machine learning models like **Linear Regression**, **Ridge Regression** and **Lasso Regression** using the evaluation metric **R^2 Score**.

The R^2 Score is always between 0 and 1 for every model, the model whose value is closer to 1 performs better than those models whose R^2 Score is closer to 0.

2. LITERATURE SURVEY

Selvamuthu et al. [1] People are interested in stock price forecasting as the stock market develops. The stock market is a non-linear, noisy, deterministic chaotic system. A trader's main goal is to predict the stock price so he can sell it or buy it before it drops. In recent years, artificial neural networks have been used to solve many problems. These functions are used to predict future stock market prices using historical data and adaptive weights. Using ARIMA and MLP (Multilayer Perceptron), Mehdi Khashei and Zahra Haji Rahimi evaluated the performance of series and parallel strategies in 2017. (Mehdi & Zahra, 2017). Yodele et al. (2012) studied stock price index movement using ANN and SVM models (SVM). From 30 November 2017 to 11 January 2018, we used ANN to predict the stock price of Reliance Private Limited (excluding holidays). Such as LSTM and RNNs may provide better predictions than the ones used in this study (Long Short-Term Memory) Using tick data, all algorithms are 99.9% accurate. The accuracy changes completely over 15 minutes. However, the result obtained using 15-minute data is significantly inferior to that obtained using tick data.

Borovkova et al. [3] proposed an ensemble of Long-Short-Term-Memory (LSTM) neural networks for intraday stock predictions, using a variety of technical analysis indicators as network inputs. The proposed ensemble operates in a way that weighs the individual models proportionally to their recent performance, allowing us to deal with possible non-stationaries in an efficient manner. The performance of the models was measured by Area Under the Curve (AUC) and Receiver Operating Characteristic (ROC). The predictions obtained from the model using US stocks are compared with Lasso and Ridge Logistic classifiers.

Sharan et al. [4] This paper aimed at optimizing the hyperparameters of the LSTM neural network to predict stock's closing price obtained from National Stock Exchange (NSE), India containing companies from different sectors like banking, industry, etc. The authors converted the data obtained to time series format and used two different variations of LSTM neural network, Stateless LSTM and Stateful LSTM and compared their prediction results using the Root Mean Square Error (RMSE) metric.

Kimoto et al. [5] This paper proposed a buying and selling timing prediction system for stocks on the Tokyo Stock Exchange (TSE). The system is made up of multiple neural networks that learn the relationships between various technical and economical indexes and the timing for when to buy and sell the stock using a method – weighted sum of weekly returns. The weights are updated according to the sum of error signals after obtaining the whole learning data. The required learning rate is adjusted according to the amount of learning data present. When the amount of data increases, the learning rate reduces and vice-versa.

Nelson et al. [6] In this paper, the authors study the usage of LSTM neural networks to predict the future trends of stock price based on stock price history and technical analysis indicators. The data is obtained from IBovespa index of all stock for the period 2008 to 2015. Then it's normalized to stabilize the mean and variance of the data. The LSTM model was built with the help of TensorFlow library, and evaluated using the metrics – accuracy, precision, recall and F-score.

Shah et al. [7] This paper shows the comparison between the performance of an LSTM neural network and a Deep Neural Network (DNN) for predicting stock prices. The data for this purpose was obtained from Bombay Stock Exchange (BSE), the closing stock prices for the period 1997 to 2017. The LSTM model used the RMSPROP learning rate method whereas the DNN used ADADELTA for the same. The performance of the models was measured using two evaluation metrics – RMSE (Root Mean Square Error) and Forecast Bias.

Srinivasan et al. [8] These markets have attracted more investors than developed markets like America and Australia because of the high returns. Artificial Neural Networks (ANN) are a Deep Learning Algorithm model. This network's main uses are face recognition, image processing, natural language processing, and medical diagnosis. The following is a comprehensive literature review on the use of Artificial Neural Networks and ARIMA Models in forecasting stock movement and closing price. Gabriele D Acunto used Deep Learning to predict financial time series. A Deep Belief Network with three hidden layers and Logistic Regression were used to predict the S&P 500. The Deep Learning Network outperforms the others. Table 1 shows performance metrics for two dense units with 100 epochs for five stock indices. The accuracy of artificial neural networks in predicting stock index direction is found to be 51% for the Dow Jones, Nasdaq, S&P 500 and Nasdaq. The Shanghai Stock Exchange has the longest positive trend. NIFTY, India, and China (SSE) are less accurate due to dense population and pandemic. Based on the above results, ANN can predict the direction of the stock index with 53% accuracy when given 5 days closing price of the stock and processed on two hidden layers.

Duemig et al. [9] The model uses a long-term memory (LSTM) based recurrent neural network to predict whether the S&P 500 will increase (up) or decrease (down) over the next trading month. Machine Learning (ML) and Deep Learning (DL) tools allow for nonlinear predictor interactions, which can improve the stock return forecasts. All 7 models have a test AUC score close to 0.5, which suggests our features do not contain enough information to predict future returns. Specifically, we use an LSTM network, which is a special type of a recurring neural network that works much better than the standard version for many tasks. In theory, LSTM models are meant to consider the entire sequence for our prediction task.

Pakdaman Naeini et al. [10] In the 1980s, Werbos claimed that neural networks were better than regression methods and Box-Jenkins models for predicting stock market changes. In this paper, we focus on feed forward multi-layer neural networks. Using this model, one can predict the value of a company's stock based on its stock trade history and without any information of the current stock market. In the suggested model two neural networks, a multilayer feed-forward and an Elman recurrent are used. The value of the stock market is normalized into a range of [0, 1, 1] according to the function of the neurons in neural networks. In the Ideal case, the ratio of correct forecast trend to the real stock changes should be equal to one. In the other hand, when the direction of stock changes is predicted incorrectly, the quantity of the (8) is expected to be closer to one as much as possible which shows that the prediction error is minimum in this case.

Lawrence et al. [11] Using neural networks to forecast stock market prices is a growing area of research. The Efficient Market Hypothesis (EMH) states that markets are efficient in that opportunities for profit are discovered so quickly that they cease to be opportunities. Some researchers claim the stock market and other complex systems exhibit chaos. No technique or combination of techniques has been successful enough to consistently "beat the market", says Ramon Lawrence, University of Manitoba. Chaos theory states that the apparent randomness of the market is just nonlinear dynamics too complex to be fully understood. The EMH states that a stock's direction cannot be determined from its past price. Wilson used self-organizing neural networks to construct a nonlinear chaotic model of stock prices. Other neural networks were developed to outperform current statistical and regression techniques. Overall, neural networks are able to partially predict share prices.

3. METHODOLOGY

The whole project is divided into three major parts:

- Historical Dashboard
- Real Time Dashboard
- Forecasting Stock Price

Historical Dashboard

For this part, we developed an interactive dashboard for analysing stock market trends using the stock prices since the company's Initial Public Offering (IPO) till the previous day, at any point of time in a day.

The data for the dashboard is obtained using the yfinance (Yahoo! Finance) API only for the companies listed on the National Stock Exchange (NSE) of India, which amounts to around 2000 companies and the dashboard itself is hosted locally as a web application using the Dash framework by Plotly. Dash is an open-source Python framework provided by Plotly for developing interactive web applications. Dash makes use of some other Python and JavaScript frameworks like Flask, Plotly.js and React.js.

The layout for the dashboard was designed using Dash framework requiring sufficient knowledge of HTML5 and CSS3. The layout contains the very essential graph that plots stock price on the Y-axis and date on the X-axis, and two dropdowns, one for the company whose stock price is to be plotted and the other one for choosing the year for plotting stock price for a certain year.

Real Time Dashboard

For the real time dashboard, the same set of tools i.e., Dash framework was used to design the layout of the dashboard and host it as a web application locally.

The data for this dashboard is obtained by scraping Google Finance website with the help of a Python script that uses the requests and bs4 (Beautiful Soup 4) libraries to fetch the current stock price of a company.

The Dash framework's inbuilt interval component is used to fetch the current stock price of the company every second. This is tied together with a monitoring system for the current stock price, if it goes above or below a certain threshold price, the intraday trader will be alerted of the current price, so that he/she can take the appropriate measure i.e., buy or sell the company's stocks.

Forecasting Stock Price

As mentioned before, there are two ways to predict future stock prices, technical analysis and fundamental analysis. In this project, we make use of the technical analysis method to forecast stock price of the leading oil and natural gas company GAIL using its historical stock price of the previous 30 days to predict the current day’s price.

The proposed predictive model uses Long Short-Term Memory (LSTM), a special type of Recurrent Neural Network (RNN) which is capable of learning long-term dependencies. All RNNs have repeating modules of basic neural networks. In standard RNN, these repeating modules contain only a single layer of hyperbolic tan whereas in LSTMs instead of a single tanh layer, there are four which interact in a special way with each other.

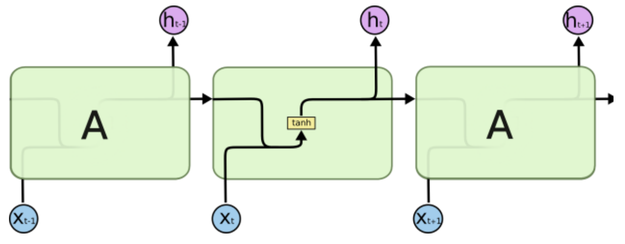


Fig 1 Simple RNN [12]

LSTMs are widely used for sequence and time-series data prediction problems and have proven to be extremely effective. The ability of LSTMs to store the past information that is deemed important and to forget the information that is not considered important is one feature to boast about.

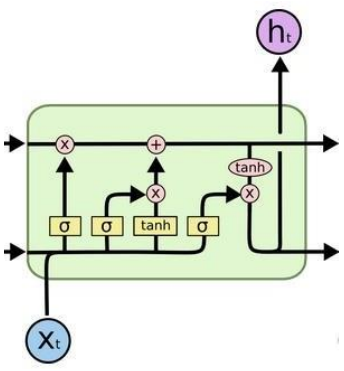


Fig 2 LSTM Neural Network [12]

The main focus of the LSTM is the horizontal line going through the block at the top, called the cell state. The LSTM can add or remove information from the cell state by using structures called as gates. A standard LSTM consists of three gates:

Input Gate: It adds information to the cell state.

Forget Gate: It removes information from cell state that is no longer required.

Output Gate: It selects the information to be shown as output.

The first step in our LSTM model is to decide what information is no longer required and should be thrown away from the cell state. This decision is made by the Forget Gate that consists of a sigmoid layer, which outputs a number between 0 and 1 where 1 represents the information is needed and should be kept in the cell state whereas 0 represents the information is no longer required.

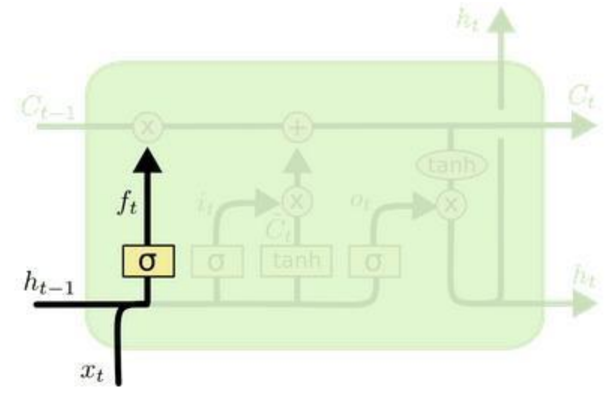


Fig 3 LSTM – Removing Information (Forget Gate) [12]

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$$

The next step is to add new information to the cell state. This happens in two parts, first it uses the Input Gate consisting of a sigmoid layer and then a tanh layer to create a vector of new informational values.

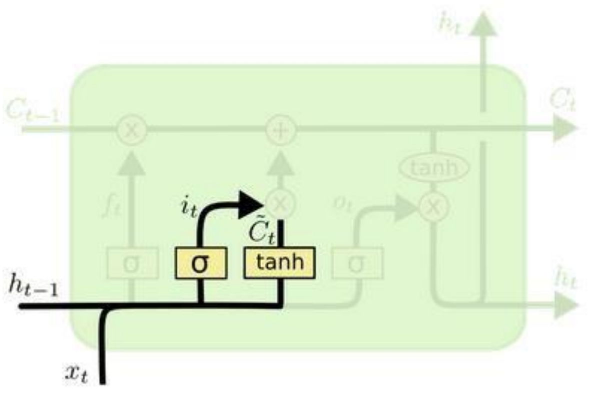


Fig 4 LSTM – Adding New Information (Input Gate) [12]

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_C.[h_{t-1}, x_t] + b_C)$$

Now, that we've already decided which part to remove and which part to add to the cell state, we can actually start doing it in this step of the LSTM.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Finally, in this last step we decide what to push as the output of our LSTM Neural Network. This is done in two parts, first we utilize the Output Gate consisting of another sigmoid layer and it is multiplied with the tanh of cell state C_t.

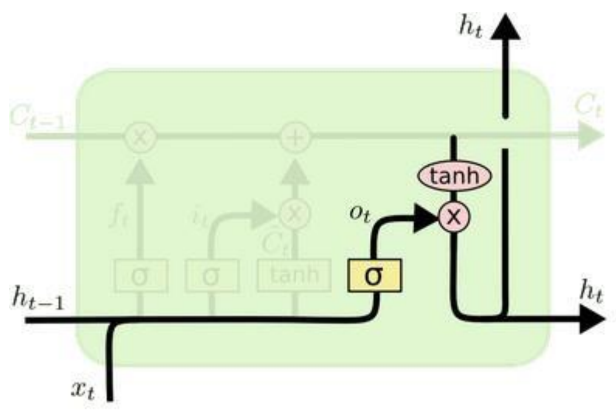


Fig 5 LSTM – Output (Output Gate) [12]

$$o_t = \sigma(W_o.[h_{t-1},x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

After predicting future stock prices of GAIL using LSTM Neural Network, we used three machine learning algorithms Linear, Ridge and Lasso Regression to do the same task, i.e., predict future stock prices of GAIL company.

Later, we use the R^2 Score metric to compare the four models and determine which is the best model among them for forecasting stock prices.

Linear Regression

Linear Regression is the most basic machine learning algorithm that estimates the relationship between the independent variables, often called features and the dependant variable called the target value or predicted value using a straight line. It does not penalize the model for its choice of weights.

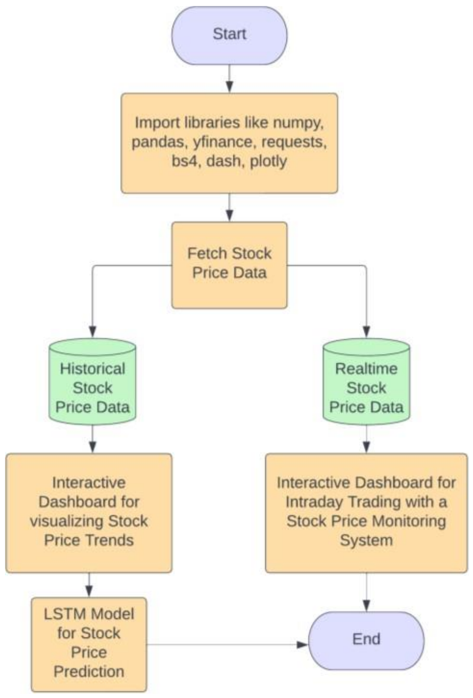
Ridge Regression

Ridge Regression is another algorithm for regression type problems that is widely used for data having high correlation between its features. This algorithm uses alpha whose value determines the penalty term that ultimately shrinks the parameters.

Lasso Regression

This is very similar to Ridge Regression but instead of shrinking the parameters by some values, it makes them complete zero while selecting some specific features to train the model. This property of making the coefficients zero is called feature selection and it is absent in Ridge Regression algorithm.

All four models are used to forecast the stock price and the same is plotted using a graph to compare their performance visually as well besides R^2 Score. The flowchart for our complete workflow for this project is given below in Figure 6.



Start
Import libraries like numpy, pandas, yfinance, requests, bs4, dash, plotly
Fetch Stock Price Data
Historical Stock Price Data
Realtime Stock Price Data
Interactive Dashboard for visualizing Stock Price Trends
Interactive Dashboard for Intraday Trading with a Stock Price Monitoring System
LSTM Model for Stock Price Prediction
End

Fig 6 Flowchart

4. RESULTS AND ANALYSIS

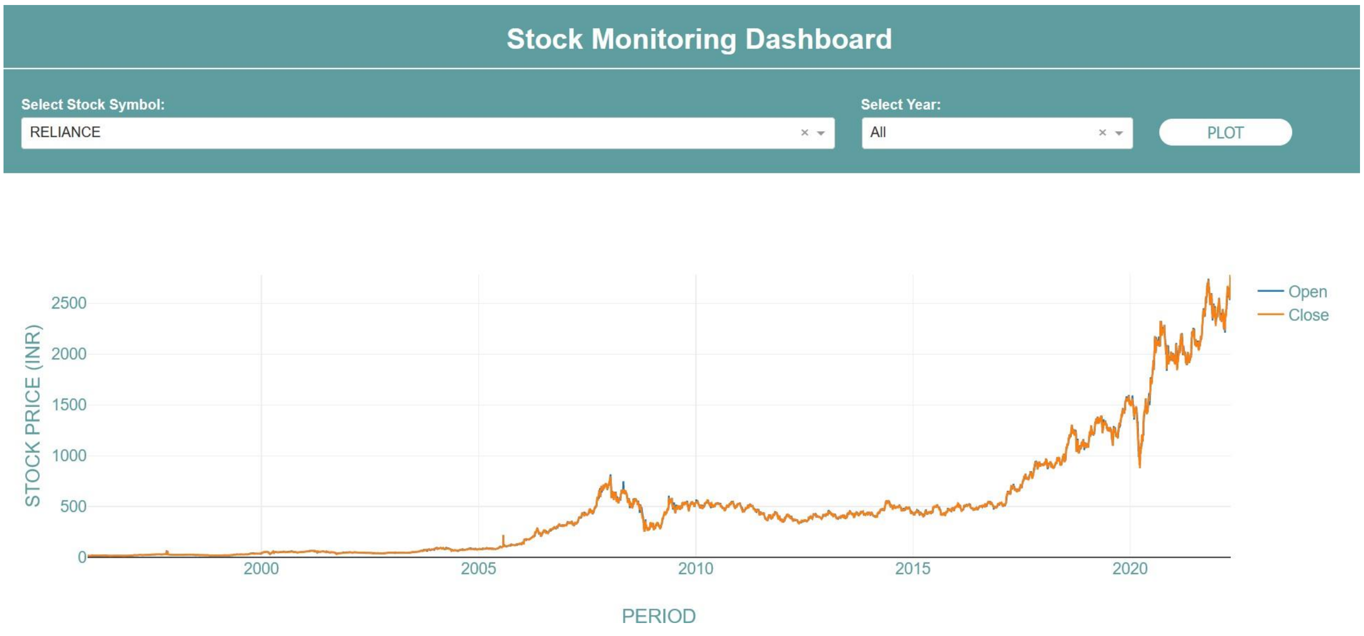


Fig 7 Historical Stock Price Dashboard

The historical stock price dashboard was completed and can be used by traders to learn stock market trends of all the companies listed on the National Stock Exchange (NSE) of India. This dashboard takes two inputs using two dropdowns, one for the stock ticker – the company whose stock prices you want to observe and the other input is for year, for traders to observe stock prices for a certain year as shown in Figure 7.

After selecting the stock symbol and the year from the two dropdowns, click on PLOT button, to get the stock price plotted on the graph. It takes a few seconds to plot the stock prices on the graph as we're downloading the historical stock price data using yfinance (Yahoo! Finance) API in real time.

For the real time stock price monitoring dashboard, due to time constraints, we were only able to plot the stock prices of a single company, in our case, its RELIANCE, in real time i.e., that updates every second. The real time data of the stock price is being scraped directly from the Google Finance website by using a Python script that makes use of requests and bs4 (Beautiful Soup 4) libraries.

The requests library helps in fetching and downloading the source code of the website and bs4 helps in parsing the source code using its inbuilt HTML parser. The stock price of RELIANCE is plotted on the graph and it's updated every second and this dashboard is extremely useful for intraday traders as shown in Figure 7.

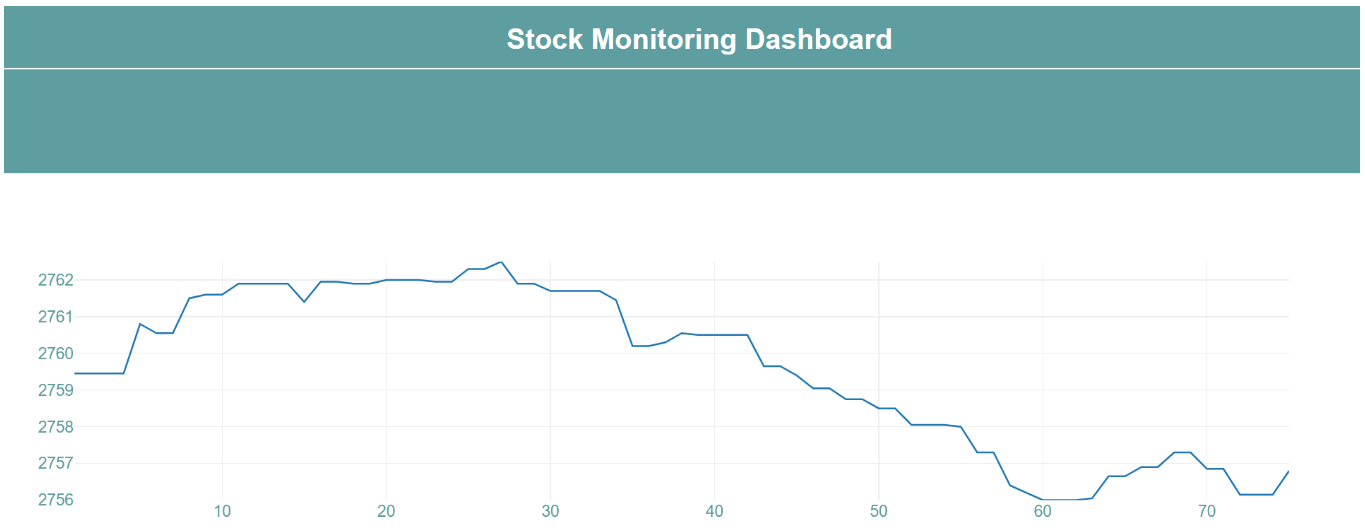


Fig 7 Real Time Stock Price Dashboard

For forecasting stock prices, we created a Sequential Neural Network with the help of TensorFlow and Keras framework provided by Google.

Proposed LSTM Model

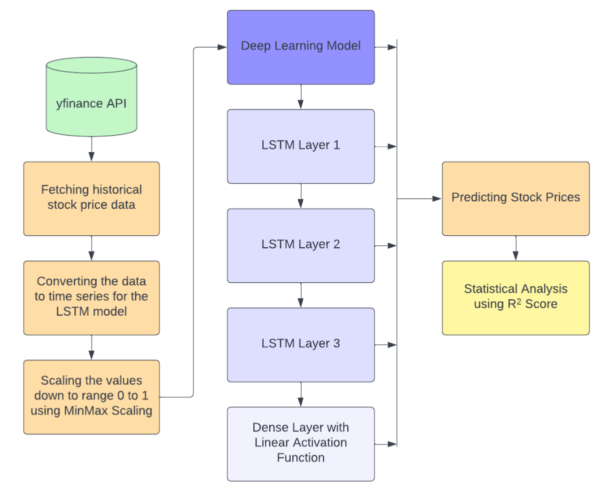

yfinance API
Fetching historical stock price data
Converting the data to time series for the LSTM model
Scaling the values down to range 0 to 1 using MinMax Scaling
Deep Learning Model
LSTM Layer 1
LSTM Layer 2
LSTM Layer 3
Dense Layer with Linear Activation Function
Predicting Stock Prices
Statistical Analysis using R² Score

Fig 8 Main Steps of Proposed Model

Our proposed LSTM model consists of total four layers of neurons, out of which three are LSTM layers and the last one is a simple Dense layer having a linear activation function.

First the whole stock price data day wise is fetched using yfinance API, afterwards its converted to time series data by taking previous 30 days of stock price values as the feature variables and the current stock price value as the target variable. 100 days stock price data is processed in the LSTM model as one batch. Along with our proposed LSTM model, we also predict the future stock prices using three machine learning algorithms – Linear, Ridge and Lasso Regression. These three machine learning algorithms are implemented very easily by using the scikit-learn library in Python. The stock prices of the next months i.e., 30 days is predicted using these three algorithms and visualized by plotting it on a graph in continuation with the actual historical stock prices of the company.

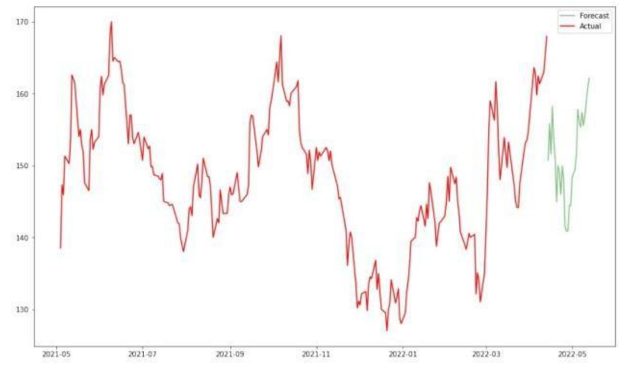


Fig 9 Linear Regression Forecasted Values

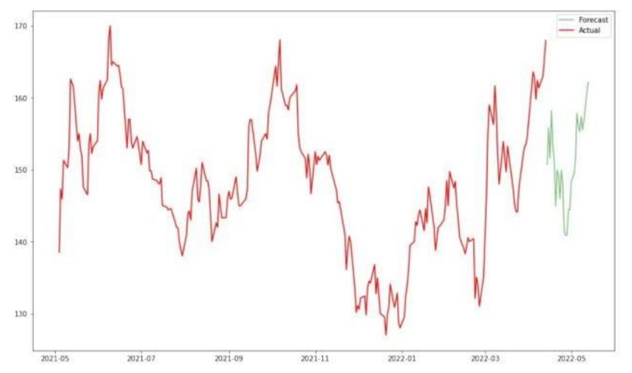


Fig 10 Ridge Regression Forecasted Values

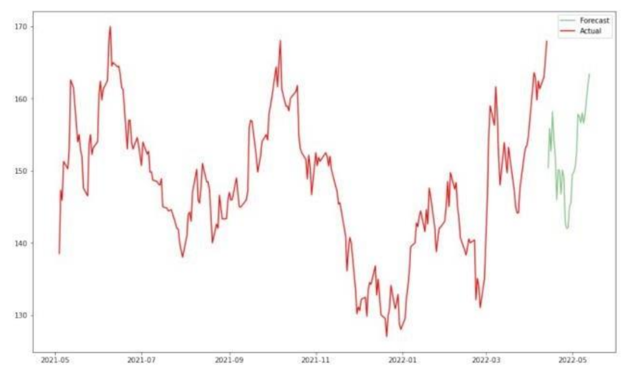


Fig 11 Lasso Regression Forecasted Values

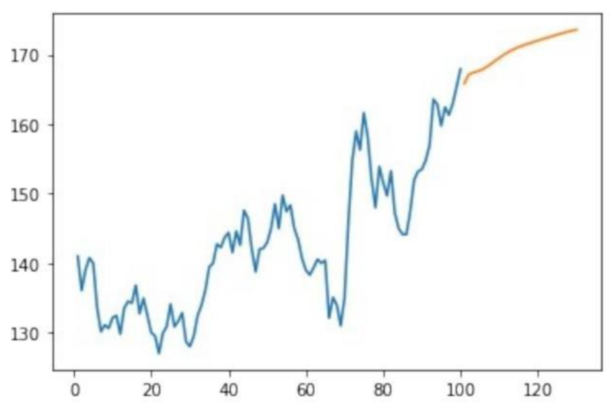


Fig 12 LSTM Model Forecasted Values

In Figures 9, 10 and 11, we can observe that the forecasted values though can match the ups and downs of prices as present in the actual historical stock prices, they are not really in continuation with the actual prices as they should've been.

As for our proposed LSTM Model, in Figure 12, we can observe that the forecasted stock prices seem a bit legit as compared to the stock prices predicted by the three machine learning algorithms.

To quantify the performance of these four models, we chose the R^2 Score evaluation metric. R^2 Score is a statistical measure of fit that tells how much variation of a dependant variable or a target variable is explained by the independent variables or features in a regression problem.

$$R^2 = 1 - \frac{\textit{Unexplained Variation}}{\textit{Total Variation}}$$

Unexplained Variation is the sum of squared errors i.e., the sum of the squares of all the differences of actual stock price value and predicted stock price value. Total Variation is the sum of the squares of all the differences of actual price value and the mean of actual stock price values. R^2 Score always lies between the range of 0 to 1, where 0 indicates that the Unexplained Variation and the Total Variation are equal i.e., no features explain the variation of the target variables at all and 1 represents that the Unexplained Variation is equal to zero i.e., the features fully explain the variation of the target variables.

The R^2 Score metric is a very good indicator for measuring and comparing the performance of various regression models. Table 1 shows the R^2 Score for all the four models.

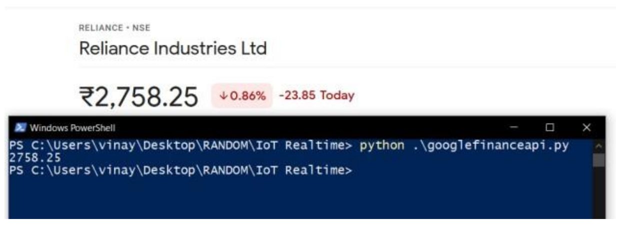
Model Name	**R^2 Score**
Linear Regression	0.751356
Ridge Regression	0.751360
Lasso Regression	0.752858
LSTM	0.825094

Table 1 Model Performan e Evaluation

5. DIFFICULTIES FACED

In India, there are two major stock exchanges, National Stock Exchange (NSE) and Bombay Stock Exchange (BSE). The companies listed in India are available in both of these exchanges but the stock prices of the same companies at one point of time may or may not be equal in the two stock exchanges. The stock price of the same company differs in both i.e., they're not in sync in real time. Due to this we were forced to fetch the stock price data from only one of these exchanges. In our project, we've chosen National Stock Exchange (NSE) prices for creating the dashboards and forecasting stock prices based on historical stock prices.

Another difficulty we faced was in creating the real time dashboard. There are several APIs that provide historical stock price data for free but we could not find any API that provides real time stock price for free. Only Premium APIs provide real time stock price data and they charge around 10$ per month for the same. So, instead of relying on these Premium APIs, we worked on a Python script that scrapes the real time stock price data of any company from Google Finance website. This Python script utilizes the requests and bs4 (Beautiful Soup) library and will not be used for commercial purposes.


RELIANCE • NSE
Reliance Industries Ltd
₹2,758.25 ↓0.86% -23.85 Today

```
PS C:\Users\vinay\Desktop\RANDOM\IoT Realtime> python .\googlefinanceapi.py
2758.25
PS C:\Users\vinay\Desktop\RANDOM\IoT Realtime>
```

Fig 13 Demonstration of Real Time Web Scraping Python Script

6. CONCLUSION

Hence, in this project as we successfully created two interactive dashboards that can be hosted locally for now for analysing stock market trends and for monitoring real time stock prices useful for intraday trading. Along with these two dashboards, we built an LSTM model for forecasting stock prices of GAIL company, and compared its performance with three machine learning algorithms – Linear Regression, Ridge Regression and Lasso Regression. The performance metric used was R^2 Score and the comparative study is summarized visually in Figure 14 below.

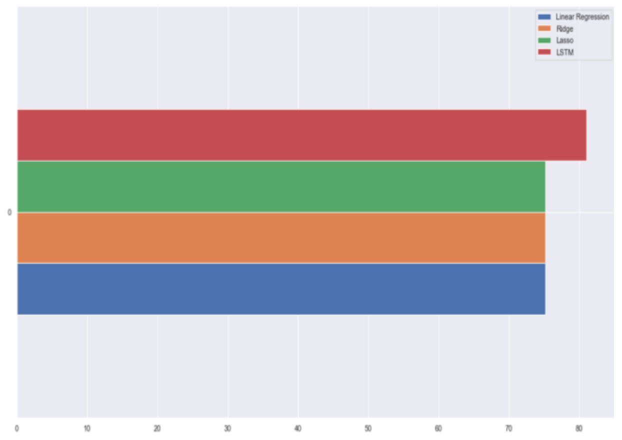


Fig 14 Model Performance Summary

7. FUTURE SCOPE

There still are a lot of parts in the project that can be worked upon. In the future, we’ll be adding the dropdowns for stock symbols along with a monitoring system provided with an alert system in the real time stock price dashboard which will prove to be useful for intraday traders. The User Interface (UI) for the dashboards can be improved in general to make it more visually pleasing as well as easier to use or operate.

In this project, we made use of only the historical stock prices to predict the future stock price values using our LSTM model. Later on, we would like to include some more impactful features like daily volume, volatility, fundamental ratios, etc. in the model which will improve the performance of our model. Analysing sentiments of stock market related news using Natural Language Processing (NLP) and feeding it to our LSTM model would be an interesting direction for our future research.

ACKNOWLEDGEMENT

In successfully completing this project, many people helped us and motivated us throughout the whole journey. We would like to thank all those people. We would like to thank Prof. Washima Tasnin under whose guidance we learned and completed this project. Your suggestions and directions were really essential for the completion of this project.

We would also like to thank our friends who constantly gave us suggestions regarding the project and our college, Vellore Institute of Technology, for giving us the opportunity to work on this wonderful project.

REFERENCES

[1] D. Selvamuthu, V. Kumar and A. Mishra. (2019) “Indian Stock Market Prediction using Artificial Neural Networks on Tick Data,” Financial Innovation 5.10.1186.

[2] Lam and Monica. (2004) “Neural Network Techniques for Financial Performance Prediction: Integrating Fundamental and Technical Analysis,” Decision Support Systems 37(4): 567-581.

[3] Borovkova and S. Tsiamas. (2019) “An Ensemble of LSTM Neural Networks for High-Frequency Stock Market Classification,” Journal of Forecasting 38: 600-619.

[4] A Yadav, C K Jha and A. Sharan. (2020) “Optimizing LSTM for Time-Series Prediction in Indian Stock Market,” Procedia Computer Science, vol. 167 pp. 2091-2100.

[5] Kimoto et al. (1990) “Stock Market Prediction System with Modular Neural Networks,” IJCNN International Joint Conference on Neural Networks pp. 1-6.

[6] Nelson, David & Pereira, Adriano & de Oliveira, Renato. (2017) “Stock market's price movement prediction with LSTM neural networks,” IJCNN International Joint Conference on Neural Networks pp. 1419-1426.

[7] D. Shah, W. Campbell and F. H. Zulkernine. (2018) “A Comparative Study of LSTM and DNN for Stock Market Forecasting,” IEEE International Conference on Big Data pp. 4148-4155.

[8] P. V. Chandrika, K. Sakthi Srinivasan. (2021) “Predicting Stock Market Movements Using Artificial Neural Networks,” Universal Journal of Accounting and Finance, 9(3), 405 - 410.

[9] David Duemig. (2019) “Predicting Stock Prices with LSTM Networks,” Stanford University CS230 Deep Learning.

[10] Mahdi Pakdaman Naeini, Hamidreza Taremian, and Homa Baradaran Hashemi. (2010) “Stock market value prediction using neural networks,” In International Conference on Computer Information Systems and Industrial Management Applications (CISIM), pp. 132–136.

[11] Ramon Lawrence. (1997) “Using Neural Networks to Forecast Stock Market Prices,” Department of Computer Science, University of Manitoba.

[12] Colah C. (2015) “Understanding LSTM Networks,” on Colah’s Personal Blog available at https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[13] Dash Documentation and User Guide by Plotly available at https://www.dash.plotly.com

[14] TensorFlow Core v2.8.0 Documentation available at https://www.tensorflow.org/api_docs/python/tf/keras

[15] Google Finance Website available at https://www.google.com/finance/

[16] Yahoo Finance API available at https://pypi.org/project/yfinance/

APPENDIX

1. Historical Stock Price Dashboard Code

Directory Structure

```
project/
├── assets/
│   └── styles.css
├── app.py
└── tickers.py
```

styles.css

```css
body {
    margin: 0;
    font-family: 'Helvetica';
    overflow: hidden;
}

#container {
    height: 150px;
    width: 100%;
    background-color: cadetblue;
    padding: 20px;
}

#header {
    margin: 0;
    margin-bottom: 30px;
    color: white;
    text-align: center;
}

#stock-symbol-dropdown {
    width: 60%;
    margin-top: 5px;
    display: inline-block;
}

#year-dropdown {
    width: 20%;
    display: inline-block;
    margin-left: 30px;
}

#plot-btn {
    position: absolute;
    display: inline-block;
    background-color: white;
    border: 0;
    color: cadetblue;
    width: 150px;
    height: 30px;
    font-size: 18px;
    border-radius: 20px;
    margin-top: 7px;
    margin-left: 30px;
}

#plot-btn:hover {
    background-color: rgb(193, 222, 223);
    color: white;
}

#historical-stock-price-graph {
    margin: 1%;
    height: 500px;
}

.text {
    color: white;
    font-size: 16px;
    font-weight: bold;
}

.whitespace {
    margin-right: 51.3%;
}

hr {
    border: 1px solid white;
    margin-top: -1%;
    margin-left: -5%;
    margin-bottom: 2%;
}
```

tickers.py

This file contains all the stock company tickers which is more than 2000.

app.py

```python
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input,
Output, State
import yfinance as yf
import pandas as pd
from datetime import date
import plotly
from plotly import graph_objs as go
from tickers import ticker_list
```

```
app = dash.Dash()

year_list = ['All']

app.layout = html.Div([
    html.Div(children=[
        html.H1('Stock Monitoring
Dashboard', id='header'),
        html.Hr(),
        html.Label('Select Stock Symbol:',
className='text'),
        html.Label('',
className='whitespace'),
        html.Label('Select Year:',
className='text'),
        html.Br(),
        dcc.Dropdown(id='stock-symbol-
dropdown',
        options=[
            {
                "label":
str(ticker_list[i]),
                "value":
str(ticker_list[i]),
            }
            for i in
range(len(ticker_list))
        ],
        value='RELIANCE',
        searchable=True
        ),
        dcc.Dropdown(id='year-dropdown',
        options=[
            {
                'label':
str(year_list[i]),
                'value':
str(year_list[i]),
            }
            for i in range(len(year_list))
        ],
        value='All',
        searchable=False),
        html.Button(
            'PLOT',
            id='plot-btn',
            n_clicks=1
            )
    ],
    id='container'),
    dcc.Graph(id='historical-stock-price-
graph', animate=True),
])
```

```
@app.callback(
    Output('year-dropdown', 'options'),
    [Input('stock-symbol-dropdown',
'value')]
)
def update_dropdown(stock_symbol):
    year_list = ['All']
    stock_ticker = "%s.NS"%stock_symbol
    df = yf.download(stock_ticker)

    for year in
list(df.index.year.unique()):
        year_list.append(year)

    options = [
            {
                'label':
str(year_list[i]),
                'value':
str(year_list[i]),
            }
            for i in range(len(year_list))
        ]

    return options


@app.callback(
        Output('historical-stock-price-
graph', 'figure'),
    [
        Input('plot-btn', 'n_clicks')
    ],
    [
        State('stock-symbol-dropdown',
'value'),
        State('year-dropdown', 'value')
    ]
)
def graph_update(n_clicks, stock_symbol,
year):
    if n_clicks >= 1:
        stock_ticker =
"%s.NS"%stock_symbol
        df = yf.download(stock_ticker)

        if year == 'All':
            df_filtered = df
        else:
            df_filtered =
df.loc[date(int(year), 1,
1):date(int(year), 12, 31)]
```

```
        data = [
            go.Scatter(
                x=df_filtered.index,
                y=df_filtered['Open'],
                name='Open',
                mode='lines',
            ),
            go.Scatter(
                x=df_filtered.index,
                y=df_filtered['Close'],
                name='Close',
                mode='lines',
            )
        ]


        max_val_y = 0
        min_val_y = 0

        if max(df_filtered['Open']) >
max(df_filtered['Close']):
            max_val_y =
max(df_filtered['Open'])
        else:
            max_val_y =
max(df_filtered['Close'])

        fig = {
            'data': data,
            'layout': go.Layout(
                xaxis=dict(range=[min(df_f
iltered.index), max(df_filtered.index)]),
                yaxis=dict(range=[min_val_
y, max_val_y]),
                xaxis_title='PERIOD',
                yaxis_title='STOCK PRICE
(INR)',
                font=dict(
                    family='Helvetica',
                    size=18,
                    color='cadetblue'
                ),
                )
            }

        return fig


if_name_== '_main_':
    app.run_server()
```

2. Real Time Stock Price Dashboard

Directory Structure

```
project/
├── assets/
│   └── styles.css
├── app.py
├── tickers.py
└── googlefinanceapi.py
```

styles.css

This file is the same for both Historical and Real Time Stock Price Dashboard.

tickers.py

For the same reason as we previously mentioned, we won’t be providing the source code for this particular file.

googlefinanceapi.py

```
import requests
from bs4 import BeautifulSoup

def getHTML(url):
    return requests.get(url).text

class GFA:
    def__init__(self):
        self.url_prefix =
"https://www.google.com/finance/quote/"

    def get(self, company, exchange):
        url = self.url_prefix +
"%s:%s"%(company, exchange)
        html_document = getHTML(url)
        soup =
BeautifulSoup(html_document,
'html.parser')
        div =
soup.select('div.YMlKec.fxKbKc')
        price =
float(div[0].get_text()[1:].replace(',',
''))
        return price
```

app.py

```
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input,
Output, State
from googlefinanceapi import GFA
import pandas as pd
from datetime import date
import plotly
from plotly import graph_objs as go
from tickers import ticker_list

app = dash.Dash()
gfa = GFA()

X = []
Y = []

app.layout = html.Div([
    html.Div(children=[
        html.H1('Stock Monitoring
Dashboard', id='header'),
        html.Hr(),
    ],
    id='container'),
    dcc.Graph(id='realtime-stock-price-
graph', animate=True),
    dcc.Interval(
            id='interval-component',
            interval=5*1000,
            n_intervals=0
        )
])


@app.callback(
        Output('realtime-stock-price-
graph', 'figure'),
    [
        Input('interval-component',
'n_intervals',)
    ],
)
def graph_update(interval):

    # stock_tracker = "RELIANCE"
    # if stock_symbol != stock_tracker:
    #         X.clear()
    #         Y.clear()
    #         stock_tracker = stock_symbol

    Y.append(gfa.get('RELIANCE', 'NSE'))
    X.append(len(Y))
    print(X, Y)

    data = [
        go.Scatter(
            x=list(X),
            y=list(Y),
            mode='lines',
        ),
    ]

    fig = {
        'data': data,
        'layout': go.Layout(
            xaxis_title='',
            yaxis_title='',
            xaxis=dict(range=[min(X),max(X
)]),
            yaxis = dict(range =
[min(Y),max(Y)]),
            font=dict(
                family='Helvetica',
                size=18,
                color='cadetblue'
            ),
            )
        }


    return fig


if_name_== '_main_':
    app.run_server()
```

3. LSTM Model Notebook

```
from datetime import timedelta
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import style
from sklearn.preprocessing import
MinMaxScaler
from sklearn.linear_model import
LinearRegression, Ridge, Lasso
from sklearn.model_selection import
train_test_split
from sklearn.metrics import r2_score
import yfinance as yf
%matplotlib inline
stock_symbol = 'GAIL.NS'
```

```
df =
yf.download(tickers=stock_symbol,period='5
y',interval='1d')
df.head()
df['Open'].plot(label='GAIL', figsize=(15,
9), title='Adjusted Opening Price',
color='red', linewidth=1.0, grid=True)
plt.legend()
open_col = df['Open']
mvag = open_col.rolling(window=100).mean()
df['Open'].plot(label='Gail',
figsize=(15,10), title='Opening Price vs
Moving Average', color='red',
linewidth=1.0, grid=True)
mvag.plot(label='MVAG', color='blue')
plt.legend()
close_col = df['Close']
rd = open_col / close_col.shift(1) - 1
rd.plot(label='Return', figsize=(15, 10),
title='Return Deviation', color='red',
linewidth=1.0, grid=True)
plt.legend()
predict_days = 30
df['Prediction'] = df['Open'].shift(-
predict_days)
X = np.array(df.drop(['Prediction'], axis
= 1))
X = X[:-predict_days]
print(X.shape)
y = np.array(df['Prediction'])
y = y[:-predict_days]
print(y.shape)
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
y_pred =linear_model.predict(X_test)
print('R2 Score for Linear Regression : ',
r2_score(y_test, y_pred))
Linear_score=r2_score(y_test, y_pred)
X_predict =
np.array(df.drop(['Prediction'], 1))[-
predict_days:]
linear_model_predict_prediction =
linear_model.predict(X_predict)
linear_model_real_prediction =
linear_model.predict(np.array(df.drop(['Pr
ediction'], 1)))
predicted_dates = []
recent_date = df.index.max()
```

```
display_at = 1000
alpha = 0.5
for i in range(predict_days):
    recent_date += (timedelta(days=1))
    predicted_dates.append(recent_date)
plt.figure(figsize=(15, 9))
plt.plot(df.index[display_at:],
linear_model_real_prediction[display_at:],
label='Linear Prediction', color='blue',
alpha=alpha)
plt.plot(df.index[display_at:],
df['Open'][display_at:], label='Actual',
color='red')
plt.legend()
plt.figure(figsize=(15, 9))
plt.plot(predicted_dates,
linear_model_predict_prediction,
label='Forecast', color='green',
alpha=alpha)
plt.plot(df.index[display_at:],
df['Open'][display_at:], label='Actual',
color='red')
plt.legend()
ridge_model = Ridge()
ridge_model.fit(X_train, y_train)
y_pred =ridge_model.predict(X_test)
print('R2 Score for Ridge Regression: ',
r2_score(y_test, y_pred))
Ridge_score=r2_score(y_test, y_pred)
ridge_model_predict_prediction =
ridge_model.predict(X_predict)
ridge_model_real_prediction =
ridge_model.predict(np.array(df.drop(['Pre
diction'], 1)))
plt.figure(figsize=(15, 9))
plt.plot(df.index[display_at:],
ridge_model_real_prediction[display_at:],
label='Ridge Prediction', color='blue',
alpha=alpha)
plt.plot(df.index[display_at:],
df['Open'][display_at:], label='Actual',
color='red')
plt.legend()
plt.figure(figsize=(15, 9))
plt.plot(predicted_dates,
ridge_model_predict_prediction,
label='Forecast', color='green',
alpha=alpha)
plt.plot(df.index[display_at:],
df['Open'][display_at:], label='Actual',
color='red')
plt.legend()
lasso_model = Lasso()
lasso_model.fit(X_train, y_train)
```

```
y_pred =lasso_model.predict(X_test)
print('R2 Score for Lasso Regression: ',
r2_score(y_test, y_pred))
Lasso_score=r2_score(y_test, y_pred)
lasso_model_predict_prediction =
lasso_model.predict(X_predict)
lasso_model_real_prediction =
lasso_model.predict(np.array(df.drop(['Pre
diction'], 1)))
plt.figure(figsize=(15, 9))
plt.plot(df.index[display_at:],
lasso_model_real_prediction[display_at:],
label='Lasso Prediction', c='blue',
alpha=alpha)
plt.plot(df.index[display_at:],
df['Open'][display_at:], label='Actual',
color='red')
plt.legend()
plt.figure(figsize=(15, 9))
plt.plot(predicted_dates,
lasso_model_predict_prediction,
label='Forecast', color='green',
alpha=alpha)
plt.plot(df.index[display_at:],
df['Open'][display_at:], label='Actual',
color='red')
plt.legend()
print(Linear_score*100,Ridge_score*100,Las
so_score*100)
best_score = max(Linear_score*100,
Ridge_score*100,Lasso_score*100 )
index = np.argmax([Linear_score,
Ridge_score,Lasso_score])
best_regressor = {0:'Linear Regression
Model',
                  1:'Ridge Model',
                  2:'Lasso Model'}
print("The Best Performer is {0} with the
score of
{1}%.".format(best_regressor[index],
best_score))
data =
yf.download(tickers=stock_symbol,period='5
y',interval='1d')
type(data)
data.head()
len(data)
data.tail()
opn = data[['Open']]
opn.plot()
ds = opn.values
plt.plot(ds)
normalizer =
MinMaxScaler(feature_range=(0,1))
```

```
ds_scaled =
normalizer.fit_transform(np.array(ds).resh
ape(-1,1))
normalizer =
MinMaxScaler(feature_range=(0,1))
ds_scaled =
normalizer.fit_transform(np.array(ds).resh
ape(-1,1))
len(ds_scaled), len(ds)
train_size = int(len(ds_scaled)*0.70)
test_size = len(ds_scaled) - train_size
train_size,test_size
ds_train, ds_test =
ds_scaled[0:train_size,:],
ds_scaled[train_size:len(ds_scaled),:1]
len(ds_train),len(ds_test)
def create_ds(dataset,step):
    Xtrain, Ytrain = [], []
    for i in range(len(dataset)-step-1):
        a = dataset[i:(i+step), 0]
        Xtrain.append(a)
        Ytrain.append(dataset[i + step,
0])
    return np.array(Xtrain),
np.array(Ytrain)
time_stamp = 100
X_train, y_train =
create_ds(ds_train,time_stamp)
X_test, y_test =
create_ds(ds_test,time_stamp)
X_train.shape,y_train.shape
X_test.shape, y_test.shape
X_train =
X_train.reshape(X_train.shape[0],X_train.s
hape[1] , 1)
X_test =
X_test.reshape(X_test.shape[0],X_test.shap
e[1] , 1)
from keras.models import Sequential
from keras.layers import Dense, LSTM
model = Sequential()
model.add(LSTM(units=50,return_sequences=T
rue,input_shape=(X_train.shape[1],1)))
model.add(LSTM(units=50,return_sequences=T
rue))
model.add(LSTM(units=50))
model.add(Dense(units=1,activation='linear
'))
model.compile(loss='mean_squared_error',op
timizer='adam')
model.fit(X_train,y_train,validation_data=
(X_test,y_test),epochs=100,batch_size=64)
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)
```

```python
train_predict =
normalizer.inverse_transform(train_predict
)
test_predict =
normalizer.inverse_transform(test_predict)
plt.plot(normalizer.inverse_transform(ds_s
caled))
plt.plot(train_predict)
plt.plot(test_predict)
type(train_predict)
test =
np.vstack((train_predict,test_predict))
plt.plot(normalizer.inverse_transform(ds_s
caled))
plt.plot(test)
len(ds_test)
fut_inp =  ds_test[272:]
fut_inp = fut_inp.reshape(1,-1)
tmp_inp = list(fut_inp)
fut_inp.shape
tmp_inp = tmp_inp[0].tolist()
lst_output=[]
n_steps=100
i=0
while(i<30):

    if(len(tmp_inp)>100):
        fut_inp = np.array(tmp_inp[1:])
        fut_inp=fut_inp.reshape(1,-1)
        fut_inp = fut_inp.reshape((1,
n_steps, 1))
        yhat = model.predict(fut_inp,
verbose=0)
        tmp_inp.extend(yhat[0].tolist())
        tmp_inp = tmp_inp[1:]
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        fut_inp = fut_inp.reshape((1,
n_steps,1))
        yhat = model.predict(fut_inp,
verbose=0)
        tmp_inp.extend(yhat[0].tolist())
        lst_output.extend(yhat.tolist())
        i=i+1
print(lst_output)
len(ds_scaled)
plot_new=np.arange(1,101)
plot_pred=np.arange(101,131)
plt.plot(plot_new,
normalizer.inverse_transform(ds_scaled[113
7:]))
plt.plot(plot_pred,
normalizer.inverse_transform(lst_output))
```

```python
ds_new = ds_scaled.tolist()
len(ds_new)
from sklearn.metrics import
mean_squared_error, r2_score,
mean_absolute_error
y_pred = model.predict(X_test)
model = Sequential()
print('R2 Score: ', r2_score(y_test,
y_pred))
print('MAE: ', mean_absolute_error(y_test,
y_pred))
Lstm_score=r2_score(y_test, y_pred)
ds_new.extend(lst_output)
plt.plot(ds_new[1200:])
final_graph =
normalizer.inverse_transform(ds_new).tolis
t()
plt.plot(final_graph,)
plt.ylabel("Price")
plt.xlabel("Time")
plt.title("{0} prediction of next month
open".format(stock_symbol))
plt.axhline(y=final_graph[len(final_graph)
-1], color = 'red', linestyle = ':', label
= 'NEXT 30D:
{0}'.format(round(float(*final_graph[len(f
inal_graph)-1]),2)))
plt.legend()
print('Linear_score:',Linear_score*100)
print('Ridge_score:',Ridge_score*100)
print('Lasso_score:',Lasso_score*100)
print('Lstm_score:',Lstm_score*100)
best_score = max(Linear_score*100,
Ridge_score*100,Lasso_score*100
,Lstm_score*100)
index = np.argmax([Linear_score,
Ridge_score,Lasso_score,Lstm_score])
best_regressor = {0:'Linear Regression
Model',
                  1:'Ridge Model',
                  2:'Lasso Model',
                  3:'Lstm Model'}
print("The Best Performer is {0} with the
score of
{1}%.".format(best_regressor[index],
best_score))
```