A PROJECT REPORT

on

"ADVANCED STOCK PREDICTION SYSTEM"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE ENGINEERING
BY

ABHISHEK DUTTA	-	**2005072**
VYVYAAN DOGRA	-	**20051536**
NAMAN SETHI	-	**2005109**
PRATHAM SADHWANI	-	**2005818**

UNDER THE GUIDANCE OF
DR. AMBIKA PRASAD MISHRA

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA - 751024

May 2023

A PROJECT REPORT

on

"ADVANCED STOCK PREDICTION SYSTEM"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE ENGINEERING

BY

ABHISHEK DUTTA	-	2005072
VYVYAAN DOGRA	-	20051536
NAMAN SETHI	-	2005109
PRATHAM SADHWANI	-	2005818

UNDER THE GUIDANCE OF
DR. AMBIKA PRASAD MISHRA

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAE, ODISHA -751024

May 2023

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is to certify that the project entitled

"ADVANCE STOCK PREDICTION SYSTEM"

submitted by

ABHISHEK DUTTA	2005072
VYVYAAN DOGRA	20051536
NAMAN SETHI	2005109
PRATHAM SADHWANI	2005818

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during the year 2022-2023, under our guidance.

Date: / /2023

Dr. Ambika Prasad Mishra
Project Guide

Acknowledgements

We are profoundly grateful to DR. AMBIKA PRASAD MISHRA of Kalinga Institute Of Industrial Technology for his expert guidance and continuous encouragement throughout to see that this project rights its target from its commencement to its completion.

ABHISHEK DUTTA

VYVYAAN SINGH DOGRA

NAMAN SETHI

PRATHAM SADHWANI

ABSTRACT

In this project we attempt to implement various machine-learning approaches to predict stock prices. Machine learning is effectively implemented in forecasting stock prices. The objective is to predict stock prices in order to make more informed and accurate investment decisions. We propose different stock price prediction systems that integrate mathematical functions, machine learning, and other external factors for the purpose of achieving better stock prediction accuracy and issuing profitable trades. There are two types of stocks. You may know of intraday trading by the commonly used term "day trading." Inter day traders hold securities positions from at least one day to the next and often for several days to weeks or months.

SVR analysis is used as a machine learning technique in order to predict the stock market price as well as to predict stock market trends. Moreover, different types of windowing operators are used as data preprocess or input selection techniques for SVR models. Random Forest (RF) Algorithm is a machine-learning algorithm that is commonly used in stock prediction systems. It is a supervised learning algorithm that is used for classification and regression tasks.KNN is another machine learning algorithm that can be used in stock prediction systems. KNN is a non-parametric algorithm used for both classification and regression tasks. LSTMs are very powerful in sequence prediction problems because they're able to store past information. GRU is a type of recurrent neural network (RNN) that can be used in stock prediction systems. RNNs are a type of neural network that can process sequences of data, such as time-series data. This is important in our case because the previous price of a stock is crucial in predicting its future price. While predicting the actual price of a stock is an uphill climb, we can build multiple models that will predict whether the price will go up or down.

Keywords: SVR, RF, KNN, LSTM, GRU, RNN, Trade Open, Trade Close, Trade Low, Trade High

Contents

Chapter 1

Introduction

The financial market is a dynamic and complex system in which anyone can buy and sell currencies, stocks, shares, and derivatives through virtual platforms facilitated by brokers. The stock market allows investors to purchase shares of public firms through exchange or over-the-counter trading. Here in India, The Bombay Stock Exchange (BSE) is one of the oldest exchanges in the world, while the National Stock Exchange (NSE) is among the best in terms of sophistication and advancement of technology. Investment in the stock market is regarded as having high risks and high gains and so attracts a large number of investors and economists. However, information regarding a stock is normally incomplete, complex, uncertain, and vague, making it challenging to predict future economic performance. People invest in the stock market based on some analysis. Many factors were incorporated and considered as the level of investing and trading increased, such as the trading strategy to be used, complex mathematical functions that reflect the state of a specific stock, machine learning algorithms that enable the prediction of future stock value, and specific news related to the stock being analyzed. Many of these techniques are used to pre-process raw data inputs before feeding the results into neural networks as input.

The stock market is a typical area that presents time-series data and many researchers study on it and proposed various models. In this project, SVR, RF, KNN, LSTM, and GRU models are used to predict the stock price. The central idea of successful stock market prediction is achieving the best results using minimum required input data and the least complex stock market model. The objective of this paper is to determine the best methodology for an advanced stock prediction system.

Chapter 2 Basic Concepts/ Literature Review

2.1 Basic Concepts

Determining the best methodology for an advanced stock prediction system is a complex task that depends on several factors, including the specific characteristics of the financial data being used, the specific goals of the prediction task, and the available resources for implementation

One approach to determining the best methodology is to compare the performance of several different machine learning algorithms on the same dataset using appropriate evaluation metrics, such as mean absolute error (MAE), root means squared error (RMSE), and mean absolute percentage error (MAPE). This can help identify which algorithms are most effective at predicting stock prices for the specific dataset and prediction task. Some commonly used machine learning algorithms for stock prediction include:

SVR: Which is a kernel-based machine learning algorithm that can handle non-linear relationships between the input features and the target variable and can be used for regression tasks.

RF: This is a tree-based ensemble learning algorithm that can handle non-linear relationships between the input features and the target variable and can be used for feature selection.

KNN: This is a non-parametric machine learning algorithm that can handle non-linear relationships between the input features and the target variable and can be used for regression tasks.

LSTM and GRU: These are recurrent neural networks that are well-suited for modeling sequential data like stock prices and can capture long-term dependencies and non-linear relationships between the input features and the target variable.

It's important to note that while machine learning can be a powerful tool for stock prediction, it's not a silver bullet and there are inherent limitations and risks associated with using any prediction model. It's important to interpret the results of the model carefully and to consider factors beyond just the model's accuracy, such as market conditions, regulatory changes, and other economic factors.

2.2 Literature Review

2.2.1 Stock Market Prediction Using Machine Learning

Stock market prediction is an act of trying to determine the future value of a stock or other financial instrument traded on a financial exchange. This report explains the prediction of any stock using Machine Learning. The technical and fundamental or the time series analysis is used by most of stockbrokers while making the stock predictions. Python is the computer language used to make stock market predictions using machine learning. In this article, we provide a Machine Learning (ML) approach that will be taught using the stock data now available and gain intelligence before using the learned information to make an accurate prediction. In this context this study uses machine learning techniques: SVR, RF, KNN, LSTM, and GRU to predict stock prices for large and small capitalizations, employing price with daily frequency.

2.2.2 Forecasting the Stock Market Index Using Artificial Intelligence Techniques

It is impossible to anticipate an asset's future price based on information contained in historical pricing. Furthermore, due to the inherent complexity of the financial system, financial forecasting is a difficult task. The objective of this work was to use artificial intelligence techniques to model and predict the future price of a stock market index. Utilising historical price data, three artificial intelligence techniques—neural networks, support vector machines, and neuro-fuzzy systems—are used to predict the price of a stock market index in the future.

Neural Networks have two types- Convolutional neural networks and recurrent neural networks. These models have the ability to learn complex patterns and relationships in time-series data.

Support Vector Machines are a type of machine learning algorithm that can be used for both classification and regression tasks. They work by mapping data to a higher-dimensional space and finding the optimal boundary between different classes of data.

Neuro-fuzzy systems combine the principles of fuzzy logic and artificial neural networks. It is used to model complex systems and make predictions based on uncertain or incomplete information. Artificial intelligence techniques have the ability to take into consideration financial system complexities and they are used as financial time series forecasting tools.

2.2.3 Indian stock market prediction using artificial neural network

Indian stock market prediction using artificial neural networks (ANNs) has gained popularity in recent years due to their ability to learn patterns and relationships in historical data and make accurate predictions. Here are the steps involved in using ANNs for Indian stock market prediction:

1)Data Collection: The first step is to collect historical data on the Indian stock market, including stock prices, trading volumes, and other relevant financial indicators

2)Data Preprocessing: The next step is to preprocess the data to make it suitable for use in an ANN. This may involve removing missing or inconsistent data, scaling the data to a common range, and normalizing the data to improve accuracy.

3)Feature Selection: The third step is to select the most relevant features from the data. This is important as including too many features can lead to overfitting while including too few features can lead to underfitting.

4)Model Training: The fourth step is to train the ANN model using the preprocessed data. This involves feeding the model with historical data and adjusting the weights between the neurons in the network to minimize errors in the predictions.

5)Model Testing: The fifth step is to test the ANN model using new, unseen data. This involves evaluating the accuracy of the model in predicting future stock prices and other financial indicators.

ANNs have been used for Indian stock market prediction in a variety of ways, including predicting stock prices, identifying market trends, and detecting anomalies in the market.

2.2.4 The Stock Market and Investment

The stock market works as a platform through which savings and investments of individuals are efficiently channeled into productive investment opportunities and add to the capital formation and economic growth of the country.
There is a close relationship between the stock market and investment. The stock market allows companies to raise money by offering stock shares and corporate bonds and allows investors to participate in the financial achievements of the companies, make profits through capital gains, and earn income through dividends. There is a close relationship between the stock market and investment. Fluctuations in the stock market can affect the investment of firms.

2.2.5 Automated Stock Price Prediction Using Machine Learning

In order to predict market movement, investors used to analyze the stock prices using stock indicators and candlestick patterns in addition to the news related to these stocks. In this work, we propose an automated trading system that integrates mathematical functions, and machine learning for the purpose of achieving better stock prediction accuracy and issuing profitable trades. We aim to determine the price or the trend of a certain stock for the coming end-of-day.
To achieve this goal, we trained traditional machine learning algorithms and created/trained multiple deep learning models like SVR, RF, KNN, LSTM, and GRU.

2.2.6 An innovative neural network approach for stock market prediction

To develop an innovative neural network approach to achieve better stock market predictions data were obtained from the livestock market for real-time and offline analysis and results of visualizations and analytics to

demonstrate the Internet of Multimedia of Things for stock analysis. Since traditional neural network algorithms may incorrectly predict the stock market, since the initial weight of the random selection problem can be easily prone to incorrect predictions, hence we propose the deep long short-term memory neural network (LSTM) with embedded layer and the long short-term memory neural network with automatic encoder to predict the stock market. In these two models, we use the embedded layer and the automatic encoder, respectively, to vectorize the data, in a bid to forecast the stock via long short-term memory neural network

2.2.7 An Intelligent Technique for Stock Market Prediction

A stock market is a loose network of economic transactions between buyers and sellers based on stocks. The stock market is a very vulnerable place for investment due to its volatile nature. In the near past, we faced huge financial problems due to a huge drop in the price of shares in stock markets worldwide. This phenomenon brought a heavy toll on the international as well as on our national financial structure. This phenomenon can be brought under control, especially by strict monitoring and instance stock market analysis. If we can analyze the stock market correctly in time, it can become a field of large profit. The stock market is all about prediction and rapid decision-making about investment, which cannot be done without a thorough analysis of the market. If we can predict the stock market by analyzing historical data properly, we can avoid the consequences of serious market collapse and be able to take necessary steps to make the market immune to such situations.

Chapter 3

Problem Statement / Requirement Specifications

Q. Determining the best methodology for advanced stock prediction system

A. Determining the best methodology for an advanced stock prediction system depends on several factors, including the characteristics of the financial data, the specific goals of the prediction task, and the available resources for model development and implementation. Here are some general considerations when choosing a methodology for an advanced stock prediction system:

1. Data: The type and quality of data can play a major role in determining the best methodology for stock prediction. For example, if the data has strong temporal dependencies, then RNN-based models like LSTM and GRU may be a good choice. On the other hand, if the data has many features that are not strongly correlated with the target variable, then feature selection techniques or decision tree-based models like RF may be more appropriate.

2. Performance: The goal of the stock prediction system may also influence the choice of methodology. For example, if the goal is to make short-term predictions with high accuracy, then models that can capture short-term patterns in the data, such as LSTM and GRU, may be preferred. If the goal is to make long-term predictions or identify trends in the data, then models that can capture longer-term patterns, such as RF and decision trees, may be more appropriate.

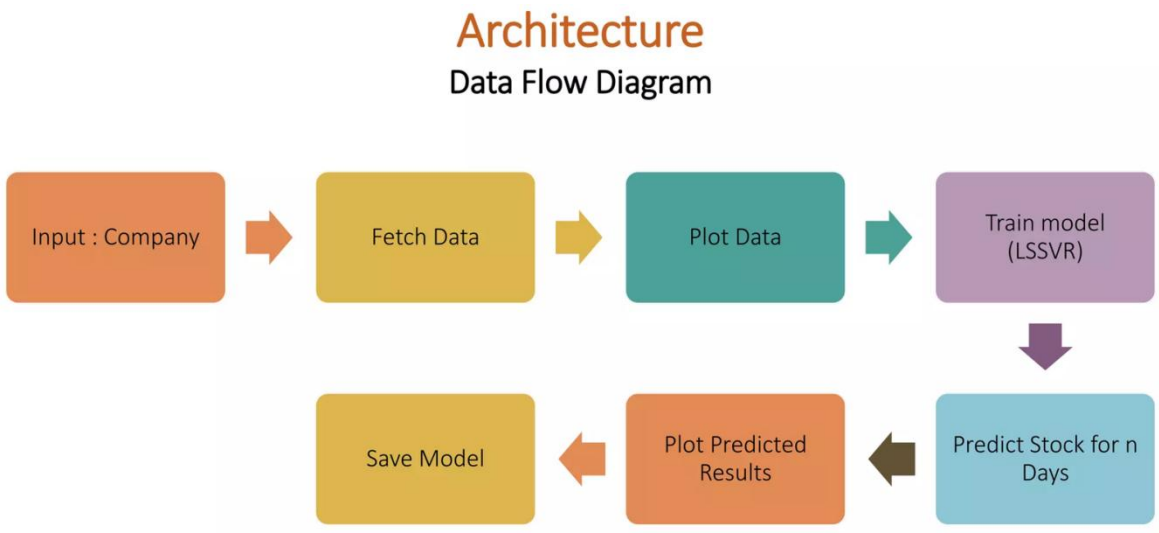
3. Scalability: Another factor to consider is the scalability of the model. Some models, such as KNN and decision trees, can be computationally expensive and may not scale well to large datasets. Other models, such as linear regression and SVR, are generally faster and more efficient.

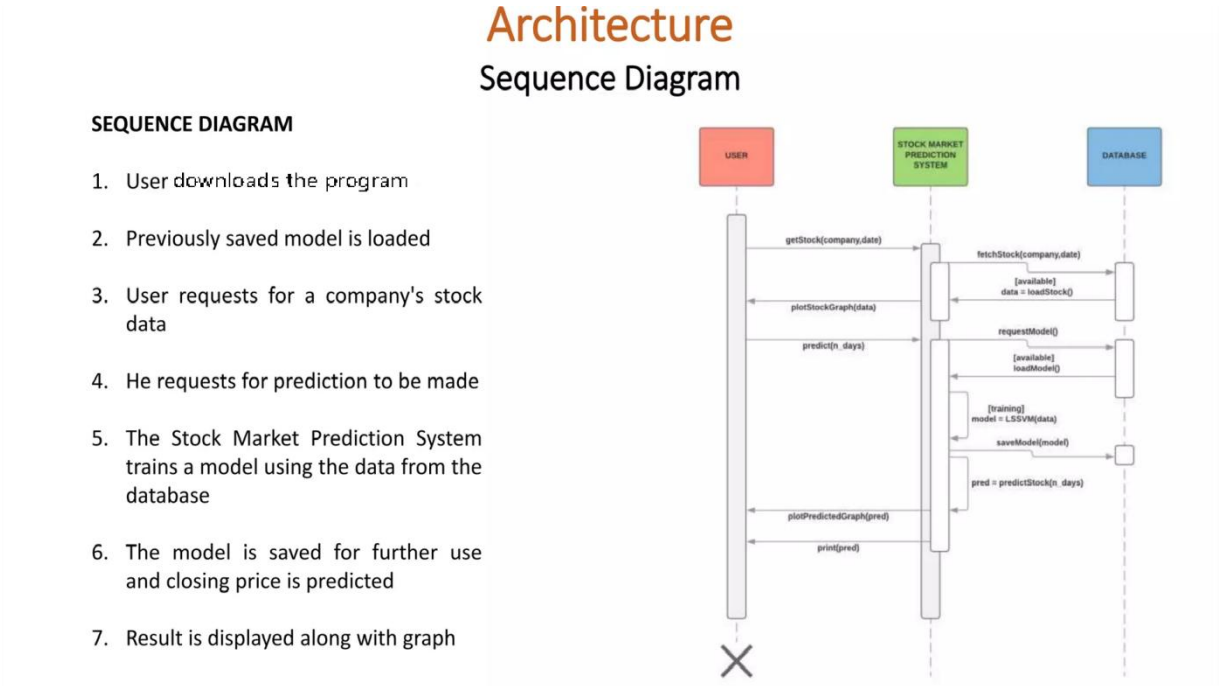
4. Interpretability: Finally, the interpretability of the model may also be important, particularly in domains such as finance where decisions can have significant consequences.

Linear regression and decision trees are generally more interpretable than complex models like neural networks, which can be difficult to interpret.

In summary, determining the best methodology for advanced stock prediction systems requires careful consideration of the specific characteristics of the financial data, the goals of the prediction task, and the available resources for model development and implementation. A combination of multiple models may also be appropriate, depending on the specific requirements of the prediction system.

3.1 Project Planning





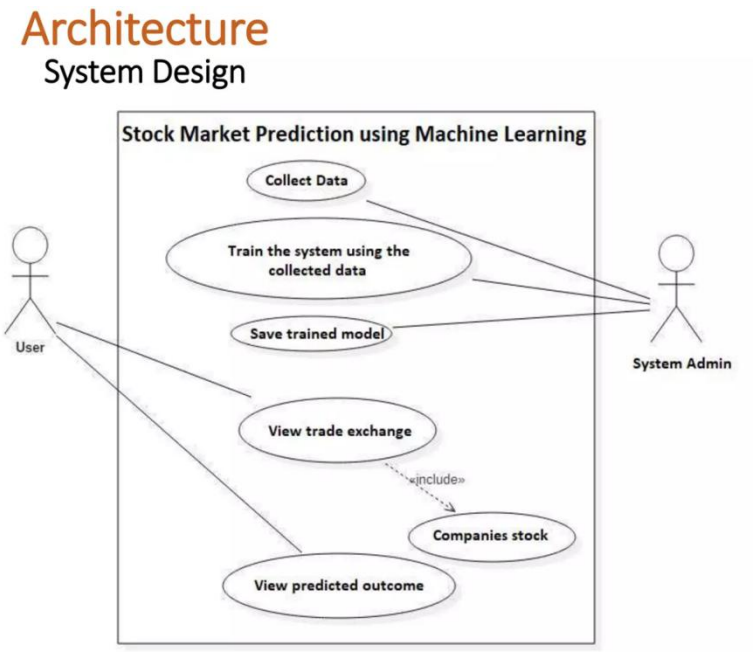
3.2 Project Analysis

After the requirements are collected or the problem statements are conceptualized, this needs to be analyzed for finding any sort of ambiguity, mistake, etc.

3.3 System Design

USE CASE DIAGRAM

1. Data is initially collected from online sources or the stock exchange
2. The data is then used to train the system
3. Trained model is saved
4. User views the trade exchange and stock of a company
5. Using the model, closing prices are predicted



3.3.1 Design Constraints

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

Hardware Requirements:

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

Software Requirements:

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

Environment: Since our project is a Python program executed in Project Jupyter Environment, we need a Python installation along with the libraries and packages used, to be installed in the Python environment system.

Import libraries and packages

```
import pandas as pd
import numpy as np
import math
import datetime as dt
from sklearn.metrics import mean_squared_error, mean_absolute_error, explained_variance_score, r2_score
from sklearn.metrics import mean_poisson_deviance, mean_gamma_deviance, accuracy_score
from sklearn.preprocessing import MinMaxScaler

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM, GRU

from itertools import cycle

import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots
```

0 ⚠ 3 Live Share Ln 1, Col 20 Spaces: 4 CRLF Cell 4 of 187 Prettier

Also, a dataset has to be imported to train and test the ML prediction algorithms so a (.csv) file format dataset has to be saved along with the program code and has to be inputted into the read path, mentioned in the program. (mentioned in the below fig.)

Import dataset

```
# Import dataset
bist100 = pd.read_csv(r'C:\Users\abhis\OneDrive\Desktop\Work\Projects\Minor-Stock Prediction System\RELIANCE.csv')
bist100.head()
```

[2] ✓ 0.0s Python

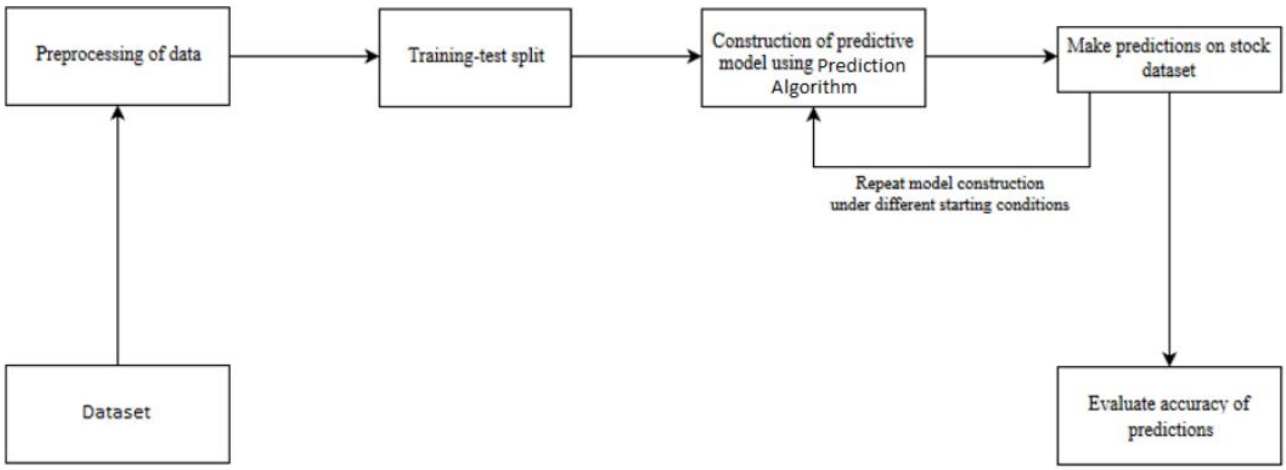
	Date	Open	High	Low	Close	Adj Close	Volume
0	2020-08-19	2141.000000	2154.000000	2121.350098	2131.550049	2124.715088	15731396.0
1	2020-08-20	2120.000000	2123.899902	2088.000000	2097.050049	2090.325684	10401212.0
2	2020-08-21	2118.000000	2122.000000	2077.000000	2081.850098	2075.174316	11667129.0
3	2020-08-24	2091.399902	2104.500000	2070.500000	2095.750000	2089.029785	15098991.0
4	2020-08-25	2106.000000	2111.300049	2078.000000	2082.100098	2075.423584	8947563.0

0 ⚠ 3 Live Share Ln 1, Col 20 Spaces: 4 CRLF Cell 4 of 187

3.3.2 System Architecture

1) Preprocessing of data

2) Overall Architecture



3.3.3 Block Diagram

Structure Chart

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

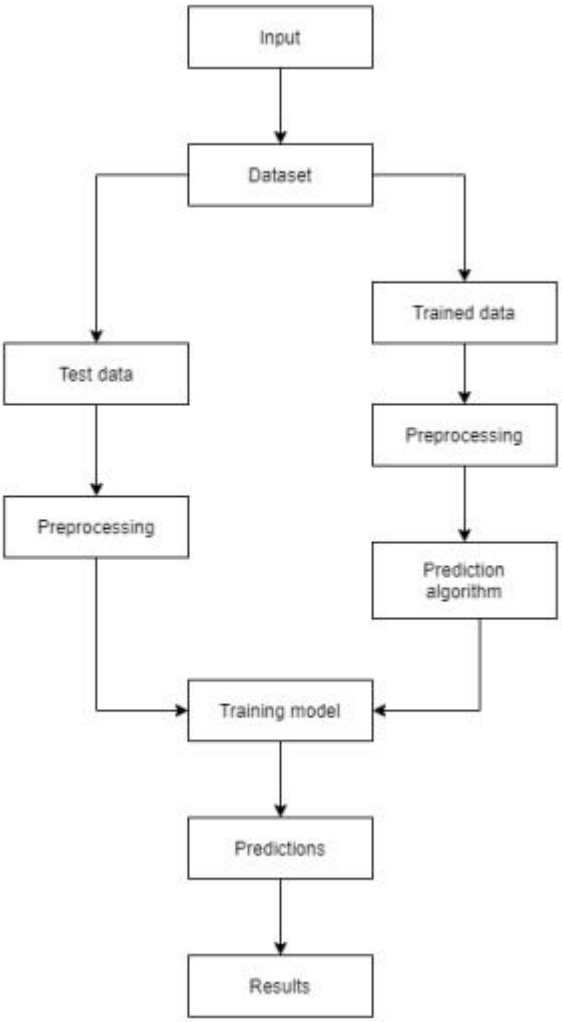


Fig.: Training and prediction

Collaboration Diagram

Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case or a part of a use case. Along with sequence diagrams, collaboration is used by designers to define and clarify the roles of the objects that perform a particular flow of events in a use case. They are the primary source of information used to determine class responsibilities and interfaces. The collaborations are used when it is essential to depict the relationship between the object. Both the sequence and collaboration diagrams represent the same information, but the way of portraying it is quite different. The collaboration diagrams are best suited for analyzing use cases.

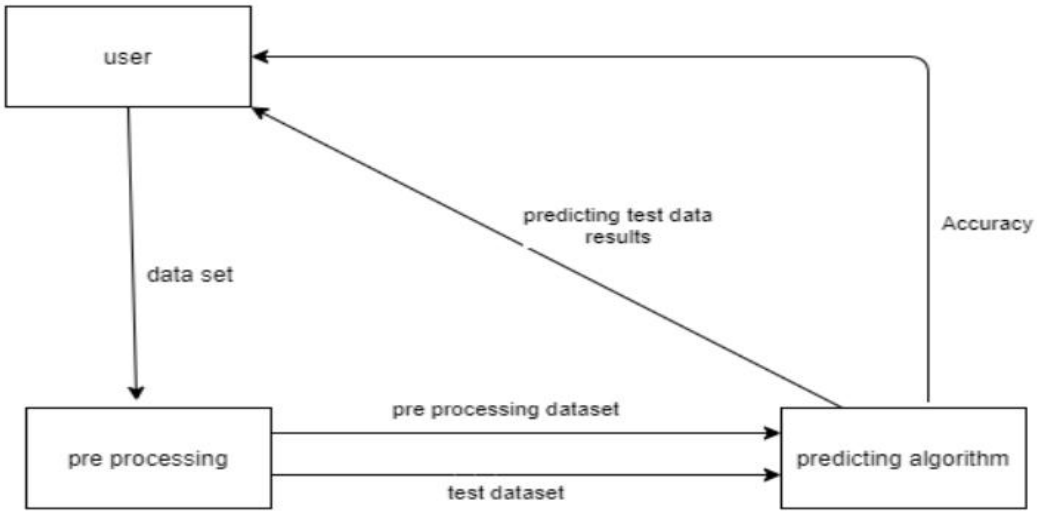


Fig.: Data transfer between modules

Component Diagram

Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that are often used to model the static implementation view of a system.

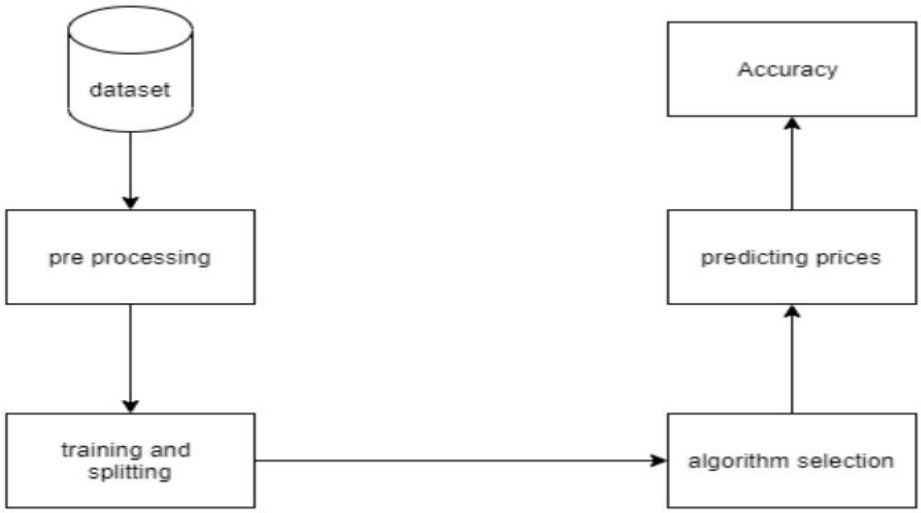


Fig.: Components present in the system

Chapter 4

Implementation

Import dataset

```
# Import dataset
bist100 = pd.read_csv(r'C:\Users\abhis\OneDrive\Desktop\Work\Projects\Minor-Stock Prediction System\RELIANCE.csv')
bist100.head()
```

[2] ✓ 0.0s Python

	Date	Open	High	Low	Close	Adj Close	Volume
0	2020-08-19	2141.000000	2154.000000	2121.350098	2131.550049	2124.715088	15731396.0
1	2020-08-20	2120.000000	2123.899902	2088.000000	2097.050049	2090.325684	10401212.0
2	2020-08-21	2118.000000	2122.000000	2077.000000	2081.850098	2075.174316	11667129.0
3	2020-08-24	2091.399902	2104.500000	2070.500000	2095.750000	2089.029785	15098991.0
4	2020-08-25	2106.000000	2111.300049	2078.000000	2082.100098	2075.423584	8947563.0

Rename columns

```
# Rename columns
bist100.rename(columns={"Date":"date","Open":"open","High":"high","Low":"low","Close":"close"}, inplace= True)
bist100.head()
```

[3] ✓ 0.1s Python

	date	open	high	low	close	Adj Close	Volume
0	2020-08-19	2141.000000	2154.000000	2121.350098	2131.550049	2124.715088	15731396.0
1	2020-08-20	2120.000000	2123.899902	2088.000000	2097.050049	2090.325684	10401212.0
2	2020-08-21	2118.000000	2122.000000	2077.000000	2081.850098	2075.174316	11667129.0
3	2020-08-24	2091.399902	2104.500000	2070.500000	2095.750000	2089.029785	15098991.0
4	2020-08-25	2106.000000	2111.300049	2078.000000	2082.100098	2075.423584	8947563.0

Preprocessing Data

Checking null and na value

```
# Checking null value
bist100.isnull().sum()
```

[4] ✓ 0.0s Python

```
date         0
open         1
high         1
low          1
close        1
Adj Close    1
Volume       1
dtype: int64
```

```
# Checking na value
bist100.isna().any()
```

[5] ✓ 0.1s Python

```
date          False
open           True
high           True
low            True
close          True
Adj Close      True
Volume         True
dtype: bool
```

```
bist100.dropna(inplace=True)
bist100.isna().any()
```

[6] ✓ 0.0s Python

```
date          False
open          False
high          False
low           False
close         False
Adj Close     False
Volume        False
dtype: bool
```

Checking datatype of each column

```
# Checking Data type of each column
print("Date column data type: ", type(bist100['date'][0]))
print("Open column data type: ", type(bist100['open'][0]))
print("Close column data type: ", type(bist100['close'][0]))
print("High column data type: ", type(bist100['high'][0]))
print("Low column data type: ", type(bist100['low'][0]))
```

[7] ✓ 0.0s Python

```
Date column data type:  <class 'str'>
Open column data type:  <class 'numpy.float64'>
Close column data type:  <class 'numpy.float64'>
High column data type:  <class 'numpy.float64'>
Low column data type:  <class 'numpy.float64'>
```

Convert date from string to date format

```
# convert date field from string to Date format and make it index
bist100['date'] = pd.to_datetime(bist100.date)
bist100.head()
```

[8] ✓ 0.0s Python

	date	open	high	low	close	Adj Close	Volume
0	2020-08-19	2141.000000	2154.000000	2121.350098	2131.550049	2124.715088	15731396.0
1	2020-08-20	2120.000000	2123.899902	2088.000000	2097.050049	2090.325684	10401212.0
2	2020-08-21	2118.000000	2122.000000	2077.000000	2081.850098	2075.174316	11667129.0
3	2020-08-24	2091.399902	2104.500000	2070.500000	2095.750000	2089.029785	15098991.0
4	2020-08-25	2106.000000	2111.300049	2078.000000	2082.100098	2075.423584	8947563.0

```
bist100.shape
```

[10] ✓ 0.0s Python

```
(249, 7)
```

EDA - Exploratory Data Analysis

Get the duration of dataset

```
print("Starting date: ",bist100.iloc[0][0])
print("Ending date: ", bist100.iloc[-1][0])
print("Duration: ", bist100.iloc[-1][0]-bist100.iloc[0][0])
```

[11] ✓ 0.0s Python

```
Starting date:  2020-08-19 00:00:00
Ending date:  2021-08-18 00:00:00
Duration:  364 days 00:00:00
```

Monthwise comparision between Stock actual, open and close price

```
monthvise= bist100.groupby(bist100['date'].dt.strftime('%B'))[['open','close']].mean().sort_values(by='close')
monthvise.head()
```

[12] ✓ 0.1s Python

date	open	close
January	1968.000006	1957.662494
April	1963.384207	1961.278956
November	1985.213160	1964.847367
May	1967.050006	1972.582513
December	1980.765897	1978.338645

```
fig = go.Figure()

fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparision between Stock actual, open and close price')
fig.show()
```

[31] ✓ 0.3s Python

Monthwise High and Low stock price

```
bist100.groupby(bist100['date'].dt.strftime('%B'))['low'].min()
```

[14] ✓ 0.0s Python

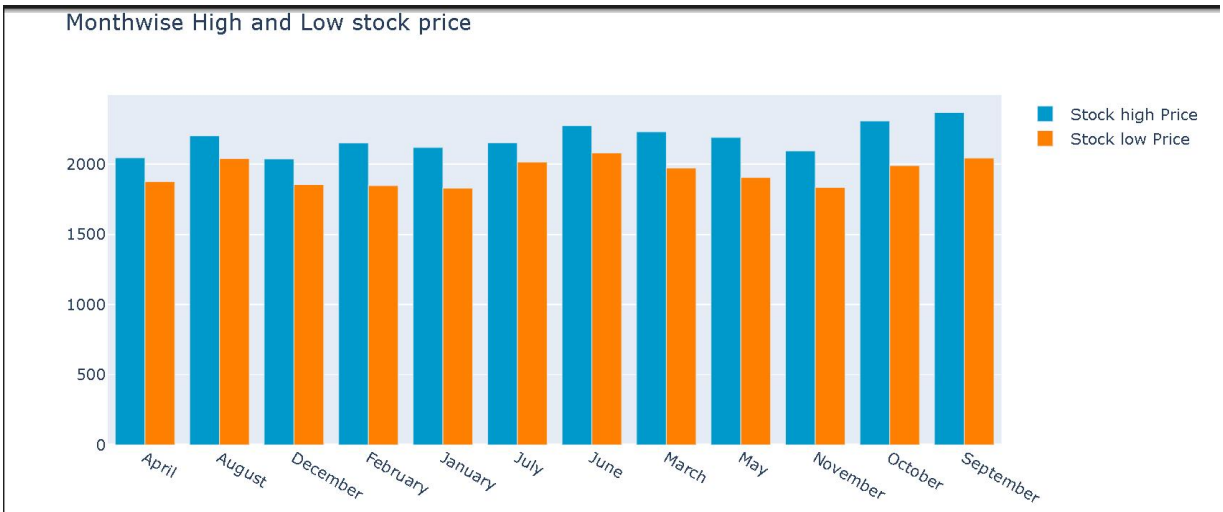
```
date
April        1876.699951
August       2041.150024
December     1855.250000
February     1848.000000
January      1830.000000
July         2016.250000
June         2081.000000
March        1973.699951
May          1906.000000
November     1835.099976
October      1991.000000
September    2044.250000
Name: low, dtype: float64
```

```
monthvise_high= bist100.groupby(bist100['date'].dt.strftime('%B'))['high'].max()
monthvise_low= bist100.groupby(bist100['date'].dt.strftime('%B'))['low'].min()
```

[15] ✓ 0.0s Python

```
fig = go.Figure()
fig.add_trace(go.Bar(
    x=monthvise_high.index,
    y=monthvise_high,
    name='Stock high Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=monthvise_low.index,
    y=monthvise_low,
    name='Stock low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title=' Monthwise High and Low stock price')
fig.show()
```

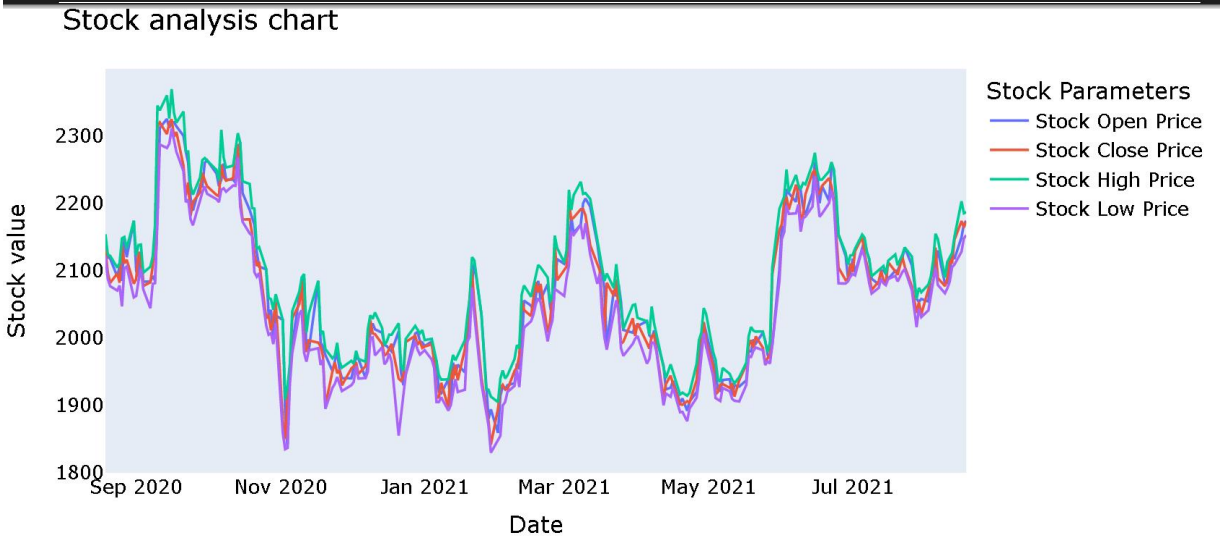
[16] ✓ 0.0s Python



Trend comparision between stock price, open price, close price, high price, low price

```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'])

fig = px.line(bist100, x=bist100.date, y=[bist100['open'], bist100['close'],
                                          bist100['high'], bist100['low']],
              labels={'date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',legend_title_text='Stock Paramete
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

[17] ✓ 1.1s Python



Close price prediction preparation and preprocessing

Make separate dataframe with close price

```
closedf = bist100[['date','close']]
print("Shape of close dataframe:", closedf.shape)
```

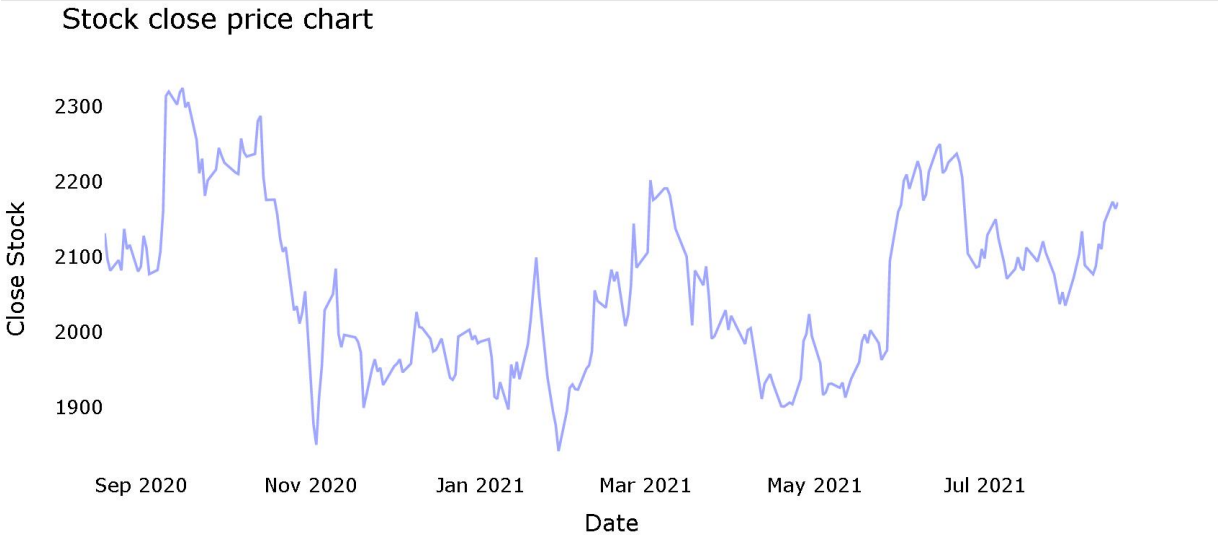
[18] ✓ 0.1s Python

```
Shape of close dataframe: (249, 2)
```

Plotting stock close price chart

```
fig = px.line(closedf, x=closedf.date, y=closedf.close,labels={'date':'Date','close':'Close Stock'})
fig.update_traces(marker_line_width=2, opacity=0.6)
fig.update_layout(title_text='Stock close price chart', plot_bgcolor='white', font_size=15, font_color='black')
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```

[19] ✓ 0.3s Python



Normalizing / scaling close value between 0 to 1

```
close_stock = closedf.copy()
del closedf['date']
scaler=MinMaxScaler(feature_range=(0,1))
closedf=scaler.fit_transform(np.array(closedf).reshape(-1,1))
print(closedf.shape)
```

[20] ✓ 0.0s Python

```
(249, 1)
```

Split data for training and testing

Ratio for training and testing data is 65:35

```
training_size=int(len(closedf)*0.65)
test_size=len(closedf)-training_size
train_data,test_data=closedf[0:training_size,:],closedf[training_size:len(closedf),:1]
print("train_data: ", train_data.shape)
print("test_data: ", test_data.shape)
```

[21] ✓ 0.0s Python

```
train_data:  (161, 1)
test_data:  (88, 1)
```

Create new dataset according to requirement of time-series prediction

```
# convert an array of values into a dataset matrix
def create_dataset(dataset, time_step=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return np.array(dataX), np.array(dataY)
```

[22] ✓ 0.0s Python

```
# reshape into X=t,t+1,t+2,t+3 and Y=t+4
time_step = 15
X_train, y_train = create_dataset(train_data, time_step)
X_test, y_test = create_dataset(test_data, time_step)

print("X_train: ", X_train.shape)
print("y_train: ", y_train.shape)
print("X_test: ", X_test.shape)
print("y_test", y_test.shape)
```

[23] ✓ 0.0s Python

```
X_train:  (145, 15)
y_train:  (145,)
X_test:  (72, 15)
y_test (72,)
```

4.1 Methodology OR Proposal

The Methodology used is the same for all the algorithm models used except the logic of the main model which differs, hence giving various values. The steps are as follows:

```
# Lets Do the prediction

train_predict=svr_rbf.predict(X_train)
test_predict=svr_rbf.predict(X_test)

train_predict = train_predict.reshape(-1,1)
test_predict = test_predict.reshape(-1,1)

print("Train data prediction:", train_predict.shape)
print("Test data prediction:", test_predict.shape)
```

[25] ✓ 0.1s Python

```
Train data prediction: (145, 1)
Test data prediction: (72, 1)
```

Training and Testing Data Prediction

1. The first line predicts the target values for the training dataset `X_train` using a regressor model named `regressor`. The predicted values are stored in a variable called `train_predict`.

2. The second line predicts the target values for the test dataset `X_test` using the same regressor model `regressor`. The predicted values are stored in a variable called `test_predict`.

3. The third line reshapes the `train_predict` variable into a 2D array with one column and as many rows as the number of data points in `X_train`. This is done using the `reshape()` function.

4. The fourth line does the same thing for the `test_predict` variable, i.e., reshaping it into a 2D array with one column and as many rows as the number of data points in `X_test`.

5. The fifth line prints the shape (number of rows and columns) of the `train_predict` variable. This provides information on the dimensions of `train_predict` after reshaping.

6. The sixth line prints the shape of the `test_predict` variable. This provides information on the dimensions of `test_predict` after reshaping.

In summary, this code generates predicted values using a regressor model for both training and test datasets. Then, the predicted values are reshaped into a 2D array. Finally, the code prints the shape of the predicted values for both the train and test datasets. The reshaping is required when feeding the predicted values to other functions or models that may require a 2D input format, for example, calculating performance metrics, plotting charts, etc.

```
# Transform back to original form

train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)
original_ytrain = scaler.inverse_transform(y_train.reshape(-1,1))
original_ytest = scaler.inverse_transform(y_test.reshape(-1,1))
```

[26] ✓ 0.0s Python

Applying Inverse Transformation

1. The first line scales back (or inverse transforms) the predicted target values for the training set, which are in normalized form due to feature scaling. The inverse transformation is performed using a scaler object named `scaler`. The result is stored in a variable called `train_predict`.

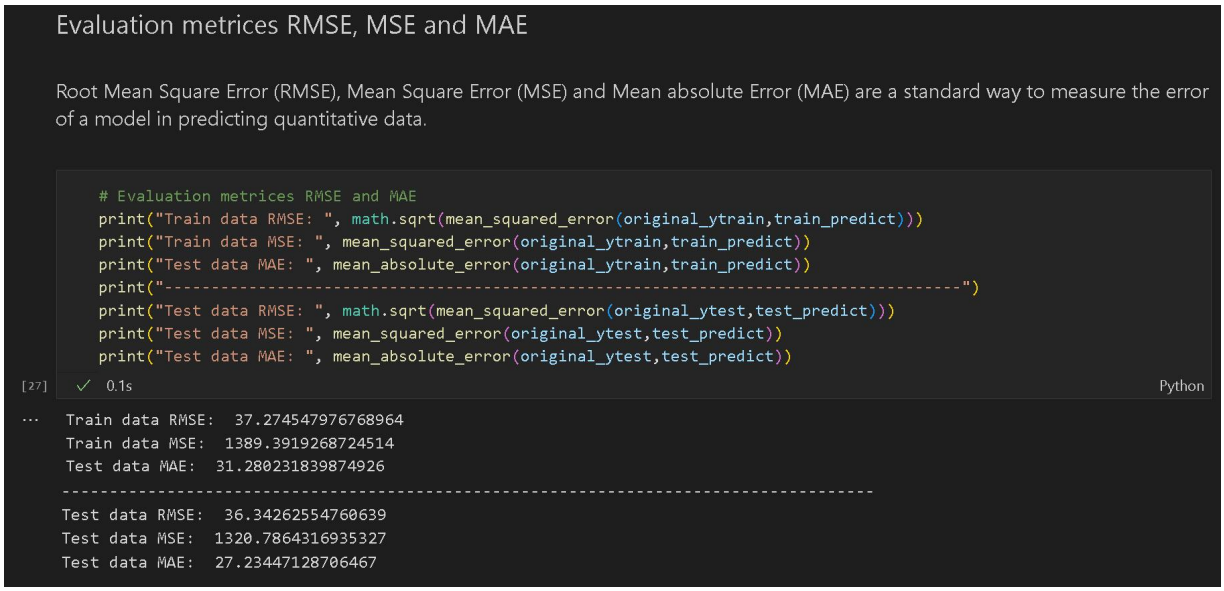
2. The second line performs the same operation as the first line, but on the predicted target values for the test set. The result is stored in a variable called `test_predict`.

3. The third line scales back the true target values of the training set that were used to train the model. The original target values were also normalized during the feature scaling process.

The `y_train` variable contains the original target values, which are reshaped into a 2D array with one column using the `reshape()` function. The inverse transformation is done using the same scaler object `scaler`. The result is stored in a variable called `original_ytrain`.

4. The fourth line performs the same operation as the third line, but on the true target values of the test set. The result is stored in a variable called `original_ytest`.

In summary, this code applies an inverse transformation (scale back) to the predicted and true target values using a scaler object. Feature scaling is a common preprocessing step in machine learning, which rescales (normalizes) the input features to improve the performance and convergence of the learning algorithm. However, after training and predicting, the results need to be transformed back into their original scale for better interpretation and evaluation purposes.



Applying RPMs to Training and Testing Datasets: RMSE, MSE, MAE

The given code is related to evaluating the performance of a regression model using various metrics. Here's the explanation of each line of code:

1. The first line prints the root mean squared error (RMSE) for the training set. RMSE is a measure of how well the model fits the data points and is calculated as the square root of the average of the squared differences between the predicted and true target values. The `math.sqrt()` function calculates the square root, and the `mean_squared_error()` function from the scikit-learn library calculates the MSE.

2. The second line prints the mean squared error (MSE) for the training set. MSE is a measure of the average squared differences between the predicted and true target values. The `mean_squared_error()` function calculates the MSE.

3. The third line prints the mean absolute error (MAE) for the training set. MAE is a measure of the average absolute differences between the predicted and true target values. The `mean_absolute_error()` function from the scikit-learn library calculates the MAE.

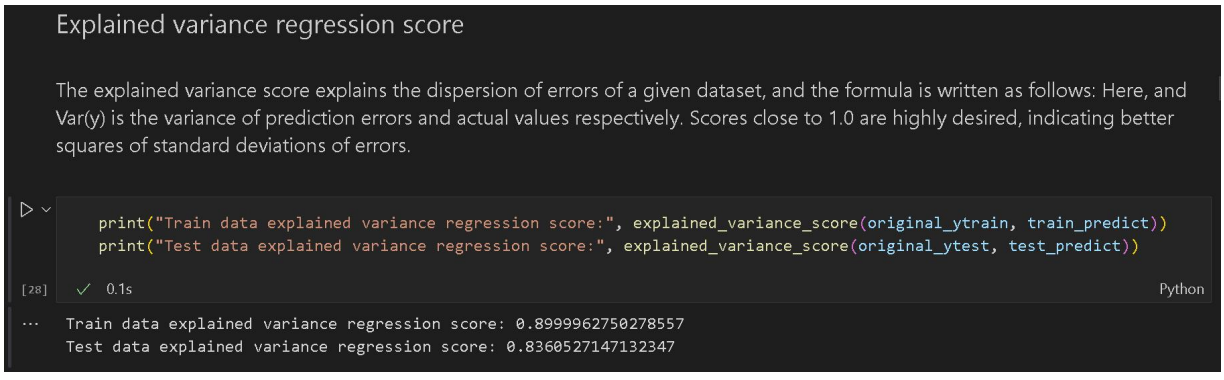
4. The fourth line is a separator that visually separates the evaluation results of the training and test sets.

5. The fifth line prints the RMSE for the test set. The calculation of RMSE and MSE are similar to those in the training set.

6. The sixth line prints the MSE for the test set.

7. The seventh line prints the MAE for the test set.

In summary, this code computes and prints several commonly used regression performance metrics: RMSE, MSE, and MAE for both the training and test sets. These metrics are useful for understanding how well the model is performing and can be used for comparing different models. By printing the evaluation results, we can determine whether any further improvements should be made to the model or if it is ready for deployment.



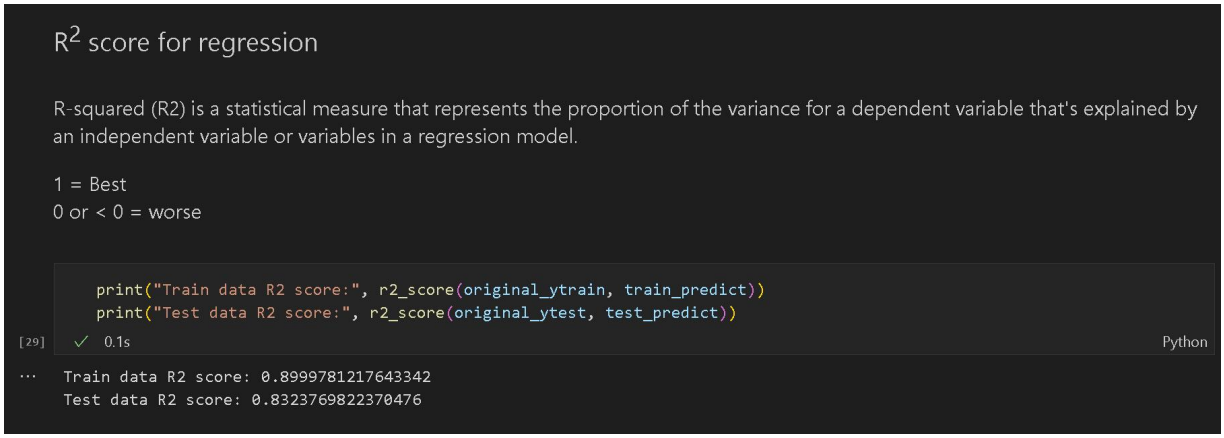
Explaining Variance Regression Score

1. The first line prints the explained variance regression score for the training set.

The explained variance regression score measures how well the model accounts for the variation in the data and is calculated as 1 - (variance of residuals/variance of target variable). A higher score indicates that the model explains more of the variability in the data. The `explained_variance_score()` function from the scikit-learn library calculates the explained variance regression score.

2. The second line prints the explained variance regression score for the test set. Similar to the previous line, it computes the explained variance regression score for the test set.

By using this metric, we can understand how much of the variance in the target variable is being explained by our model. It provides a measure of how good the model is fitting the data and can be used to compare different models or parameter settings. Generally, a higher score indicates a better fit of the model on the dataset.



Evaluating the Performance of a Regression Model using R2 Score

1. The first line prints the R2 score for the training set. The R2 score, also called the coefficient of determination, measures how well the model fits the data by determining the proportion of the variance in the dependent variable that is explained by the independent variables. This score ranges between 0 and 1, with a value of 1 indicating a perfect fit. The `r2_score()` function from the scikit-learn library calculates the R2 score.

2. The second line prints the R2 score for the test set. Similar to the previous line, it computes the R2 score for the test set.

By using this metric, we can understand how much of the variance in the target variable is being explained by our model. It provides a measure of how well the model is fitting the data and can be used to compare different models or parameter settings. A higher R2 score indicates better prediction accuracy of the model on unseen data.

Regression Loss Mean Gamma deviance regression loss (MGD) and Mean Poisson deviance regression loss (MPD)

```
print("Train data MGD: ", mean_gamma_deviance(original_ytrain, train_predict))
print("Test data MGD: ", mean_gamma_deviance(original_ytest, test_predict))
print("----------------------------------------------------------------------")
print("Train data MPD: ", mean_poisson_deviance(original_ytrain, train_predict))
print("Test data MPD: ", mean_poisson_deviance(original_ytest, test_predict))
```

[30] ✓ 0.1s Python

```
Train data MGD:  0.00032193241610047117
Test data MGD:  0.00030250240418864014
----------------------------------------------------------------------
Train data MPD:  0.6677033396020338
Test data MPD:  0.6317613148290522
```

Evaluating RMs using Mean Gamma and Mean Poisson Deviance

1. The first line prints the mean gamma deviance for the training set. Gamma deviance is a metric that evaluates a model's goodness of fit by computing the difference between the observed values and predicted values, divided by the expected value. By taking the mean of this difference across all samples, we get the mean gamma deviance. The `mean_gamma_deviance()` function probably calculates this metric.

2. The second line prints the mean gamma deviance for the test set. Similar to the previous line, it computes the mean gamma deviance for the test set.

3. The third line just prints a separator line to separate the output of the first pair and the second pair of lines for ease of reading.

4. The fourth line prints the mean Poisson deviance for the training set. Poisson deviance is another metric that measures the difference between the predicted values and the actual values, but is specifically used for count data (where the target variable is an integer). This metric requires assumptions regarding the distribution of the response variable, i.e., the rate parameter equals the predicted value. The `mean_poisson_deviance()` function probably calculates this metric.

5. The fifth line prints the mean Poisson deviance for the test set. Similar to the previous line, it computes the mean Poisson deviance for the test set.

By comparing these metrics on both the training and test sets, we can evaluate the model's performance in predicting new data. A lower value of either metric indicates better performance of the model.

Comparison between Original Stock Close Price vs Predicted Close Price

After this we utilize each and every Prediction Model: SVR, RF, KNN, LSTM, GRU, LSTM+GRU and compare the original stock close price and create a predicted close price of stocks. We also plot a graph to create a visual representation of the Original Close Price vs Prediction Close Price.

Predicting the Next 10 Days Stock Prices

After this we create a Stock Price prediction of the next 10 days based on the last 15 days of stock market prices and also plot the same for better visualization. To end it all we plot a whole graph of closing stock prices with prediction.

4.2 Testing OR Verification Plan

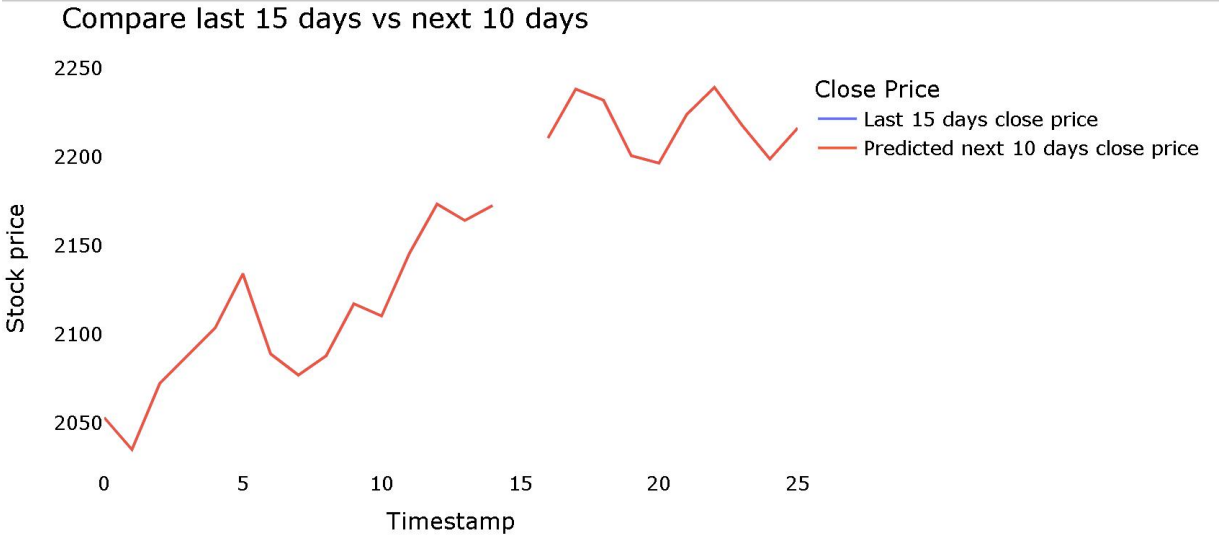
The main reason for visually representing our prediction in a graphical format was to test and check the efficiency of predicted stock prices in lieu of original stock prices. The visual explanation of our testing is as follows:

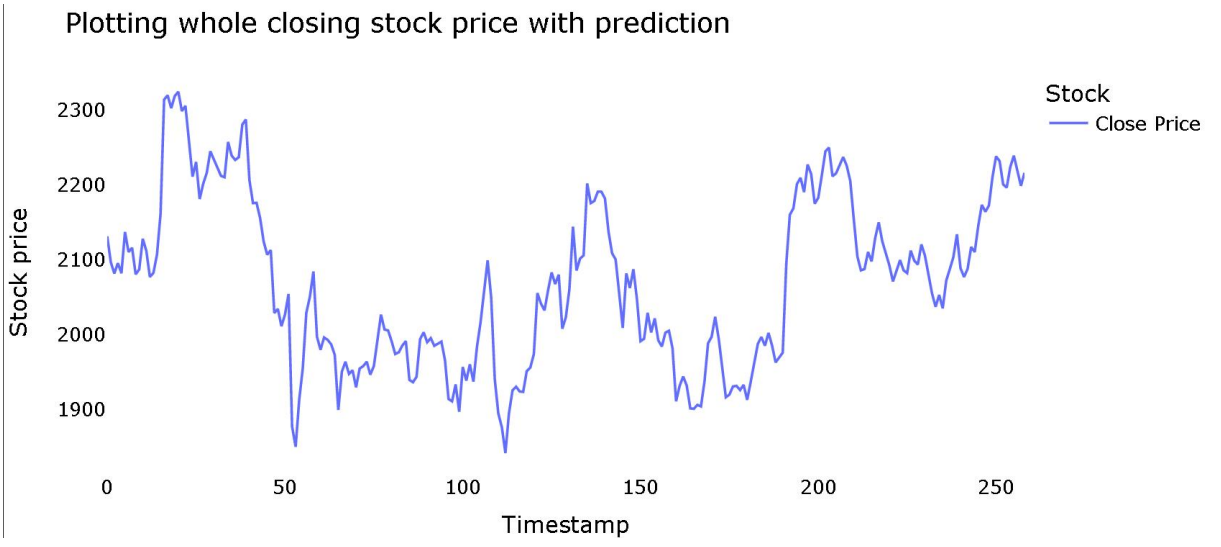
Super vector regression - SVR

```
from sklearn.svm import SVR

svr_rbf = SVR(kernel= 'rbf', C= 1e2, gamma= 0.1)
svr_rbf.fit(X_train, y_train)
```

[24] ✓ 0.4s Python

```
SVR
SVR(C=100.0, gamma=0.1)
```





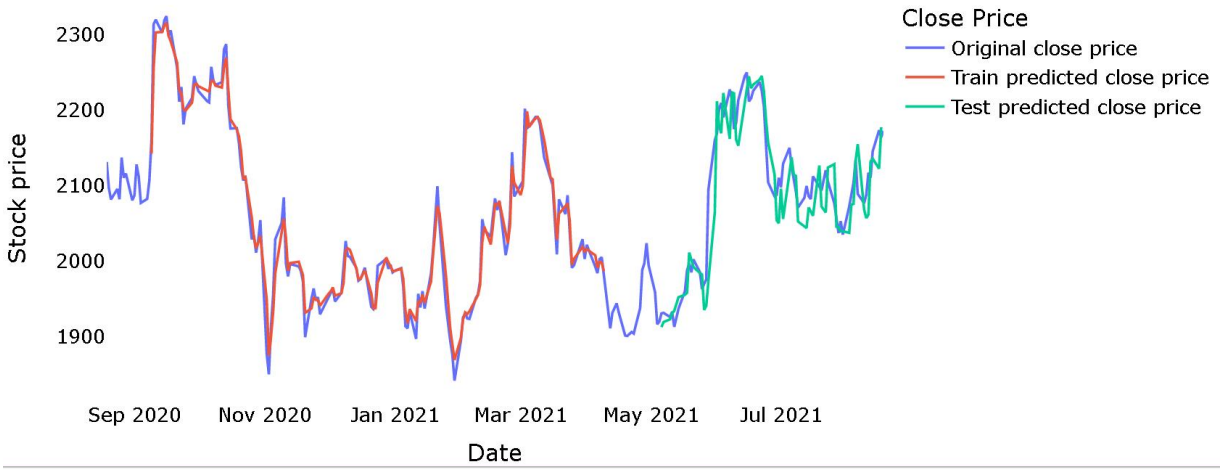
Random Forest Regressor - RF

```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)
regressor.fit(X_train, y_train)
```

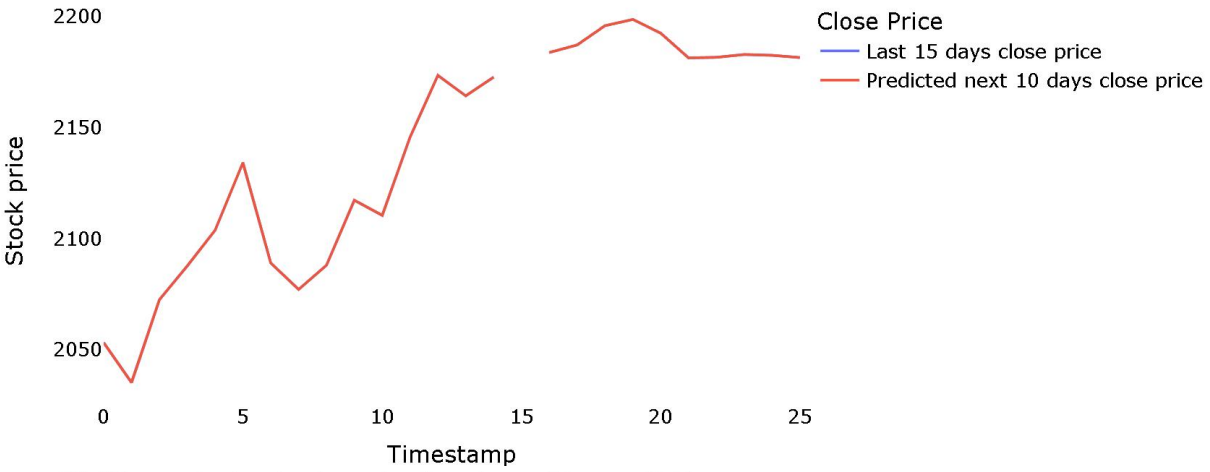
[36] ✓ 0.8s Python

```
RandomForestRegressor
RandomForestRegressor(random_state=0)
```

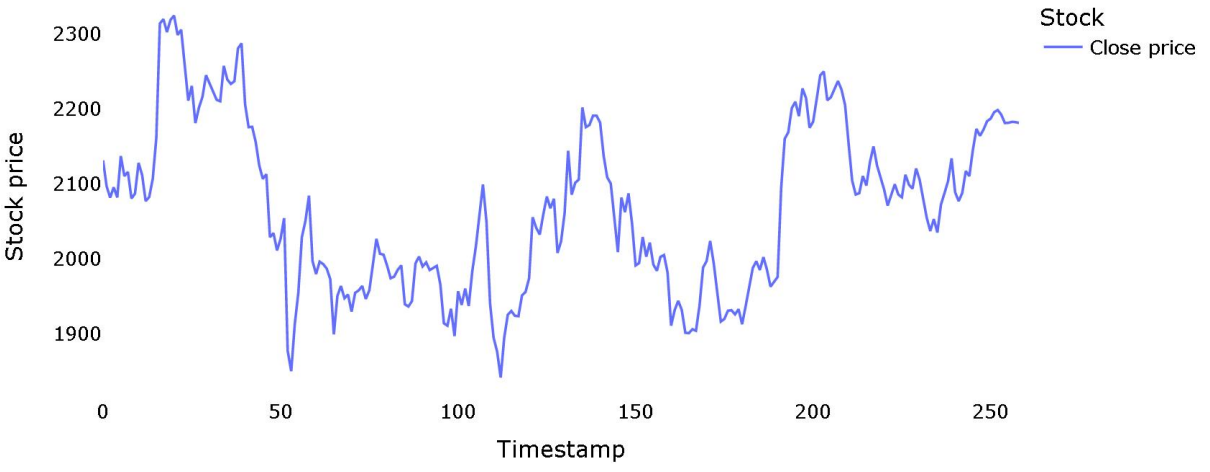
Comparision between original close price vs predicted close price



Compare last 15 days vs next 10 days



Plotting whole closing stock price with prediction

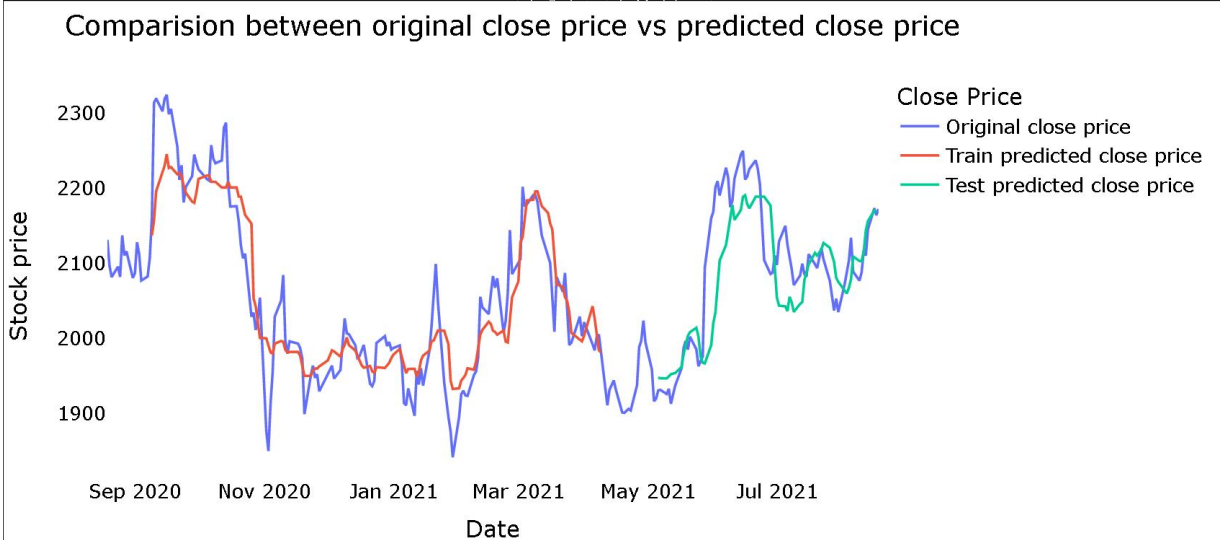


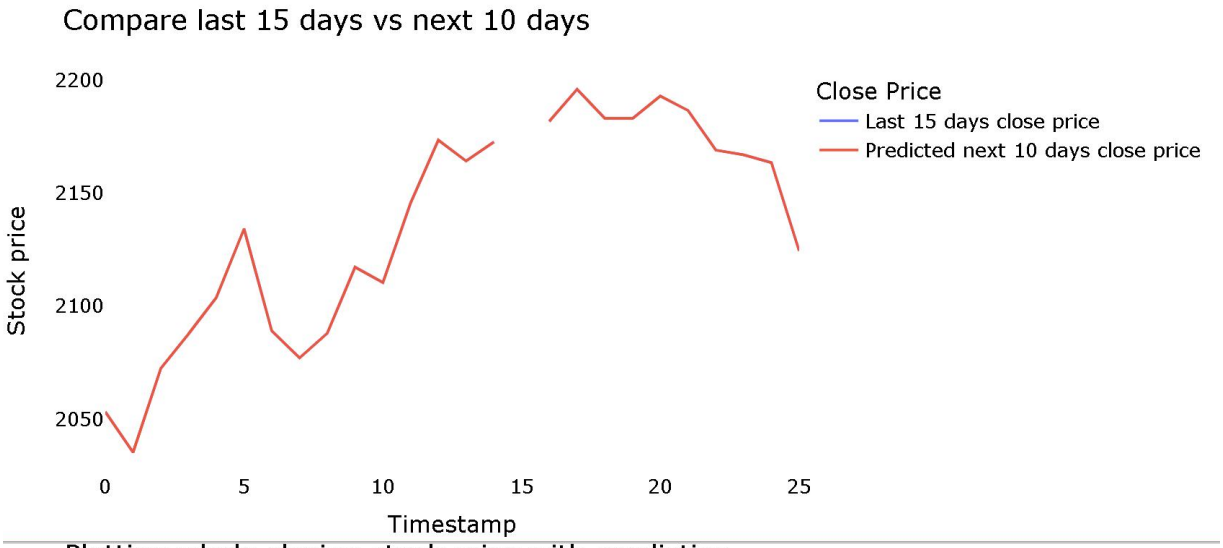
K-nearest neighgbour - KNN

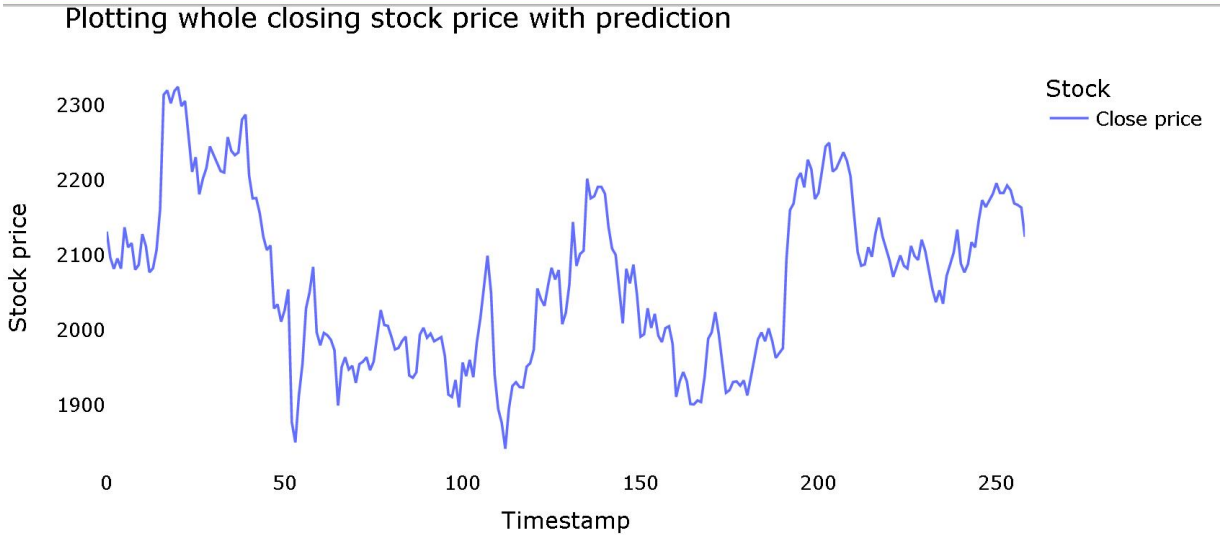
```
from sklearn import neighbors

K = time_step
neighbor = neighbors.KNeighborsRegressor(n_neighbors = K)
neighbor.fit(X_train, y_train)
```

[48] ✓ 0.0s Python

```
KNeighborsRegressor
KNeighborsRegressor(n_neighbors=15)
```







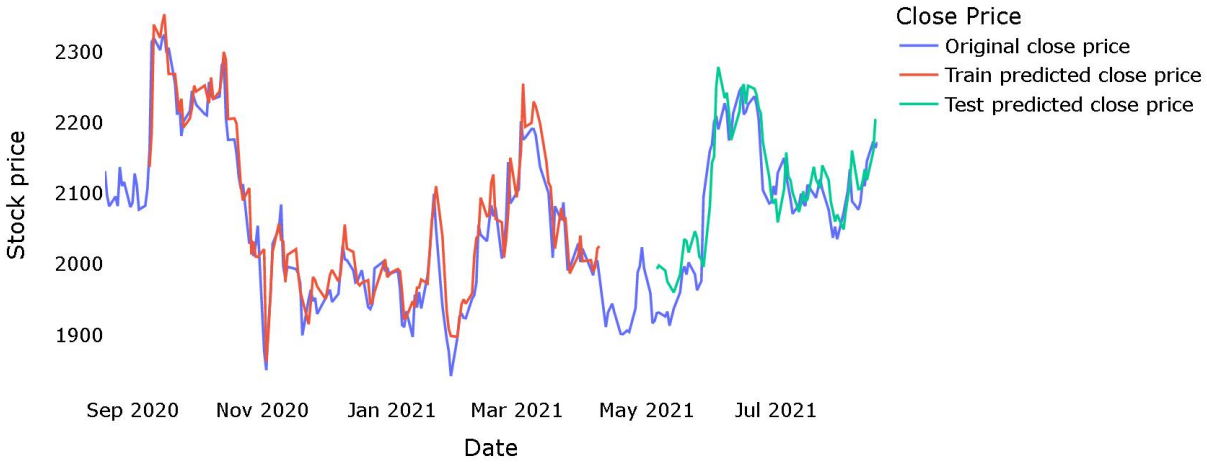
LSTM

```
# reshape input to be [samples, time steps, features] which is required for LSTM
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)

print("X_train: ", X_train.shape)
print("X_test: ", X_test.shape)
```

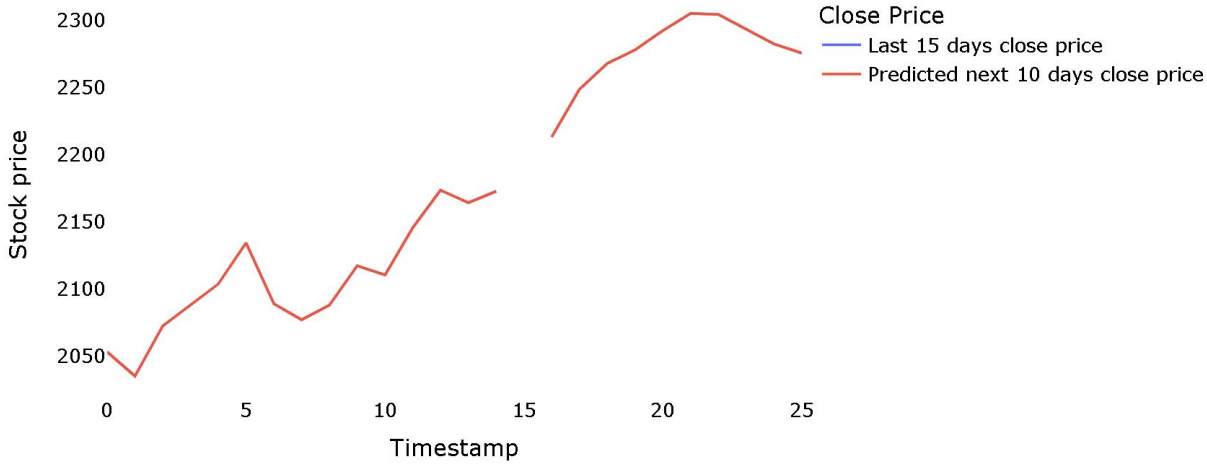
[60] ✓ 0.0s Python

```
X_train:  (145, 15, 1)
X_test:  (72, 15, 1)
```

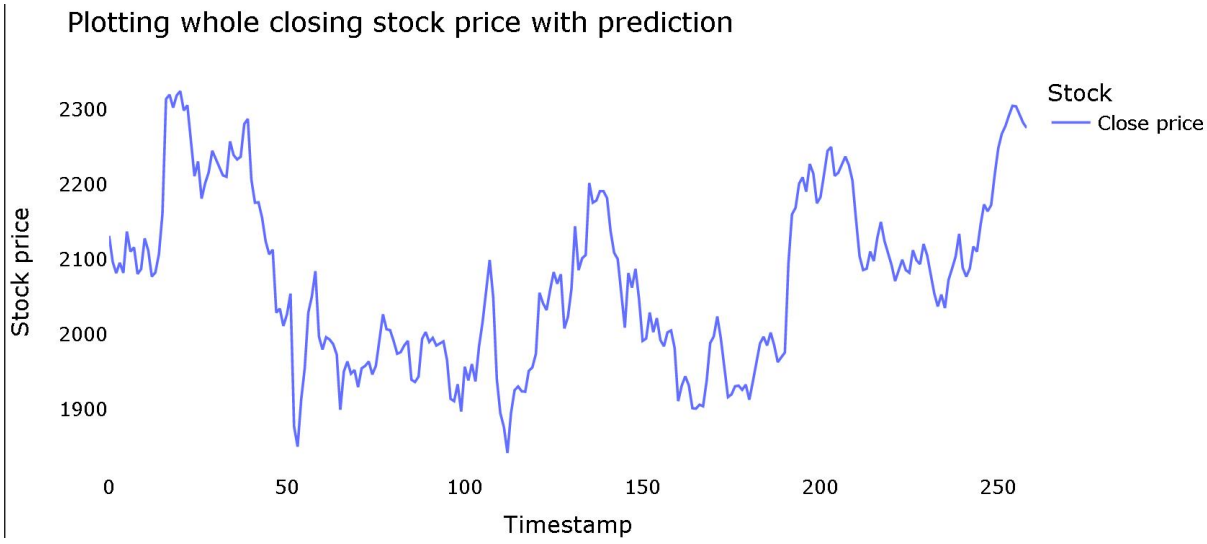
Comparision between original close price vs predicted close price



Compare last 15 days vs next 10 days



Plotting whole closing stock price with prediction

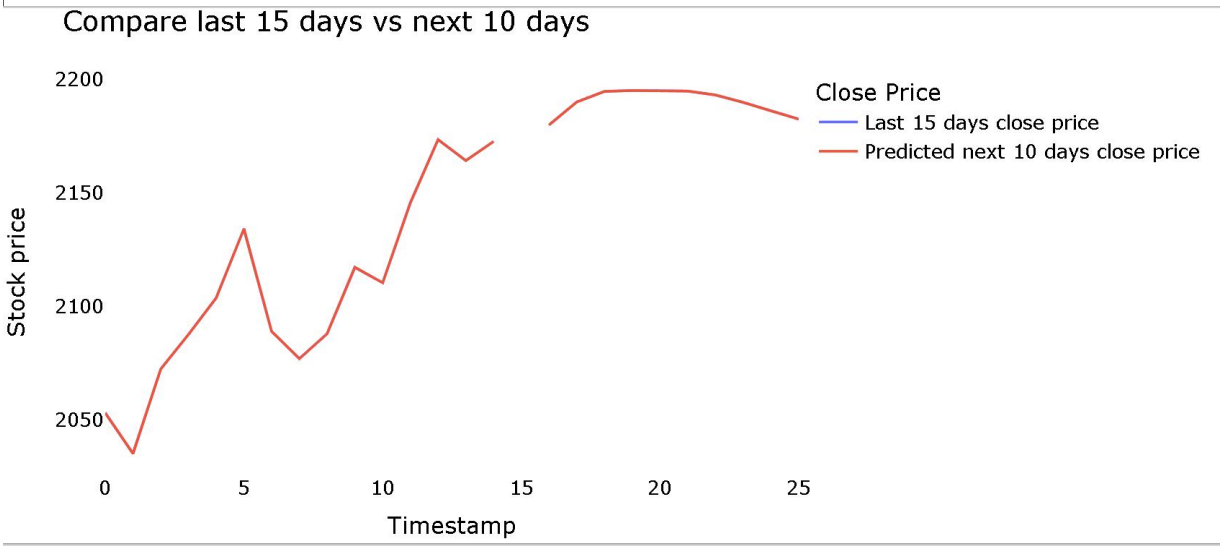


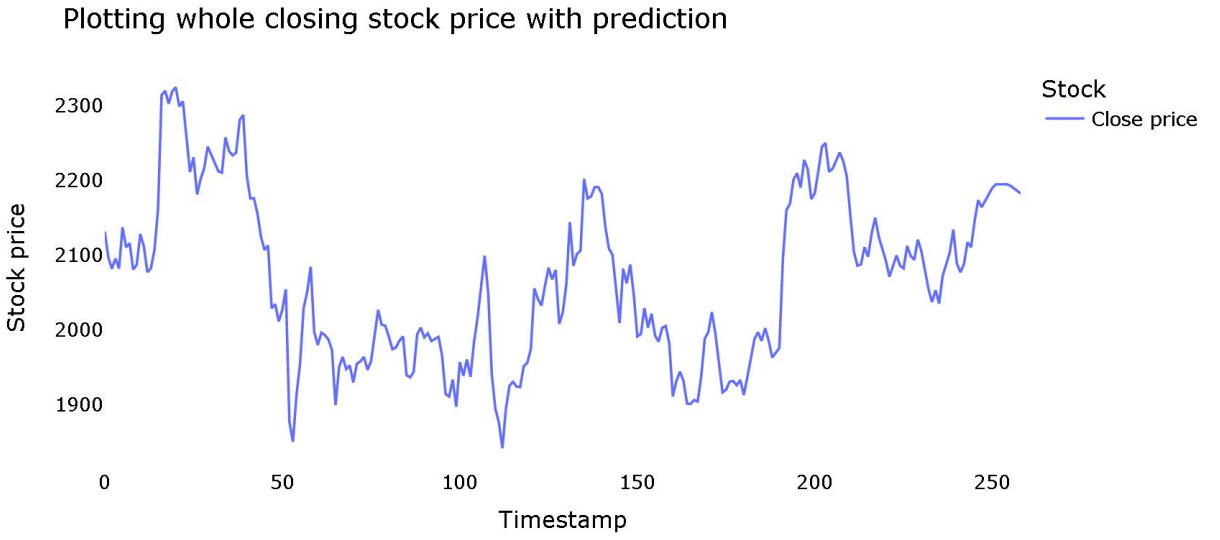
GRU

```
# reshape input to be [samples, time steps, features] which is required for LSTM
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)

print("X_train: ", X_train.shape)
print("X_test: ", X_test.shape)
```

[75] ✓ 0.0s Python

```
X_train:  (145, 15, 1)
X_test:  (72, 15, 1)
```



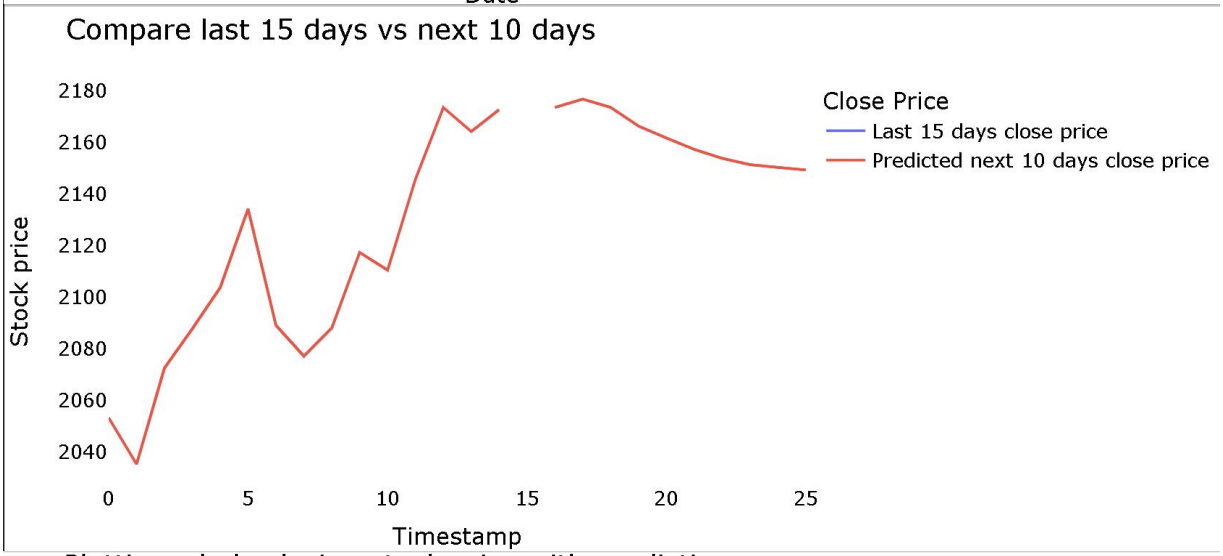


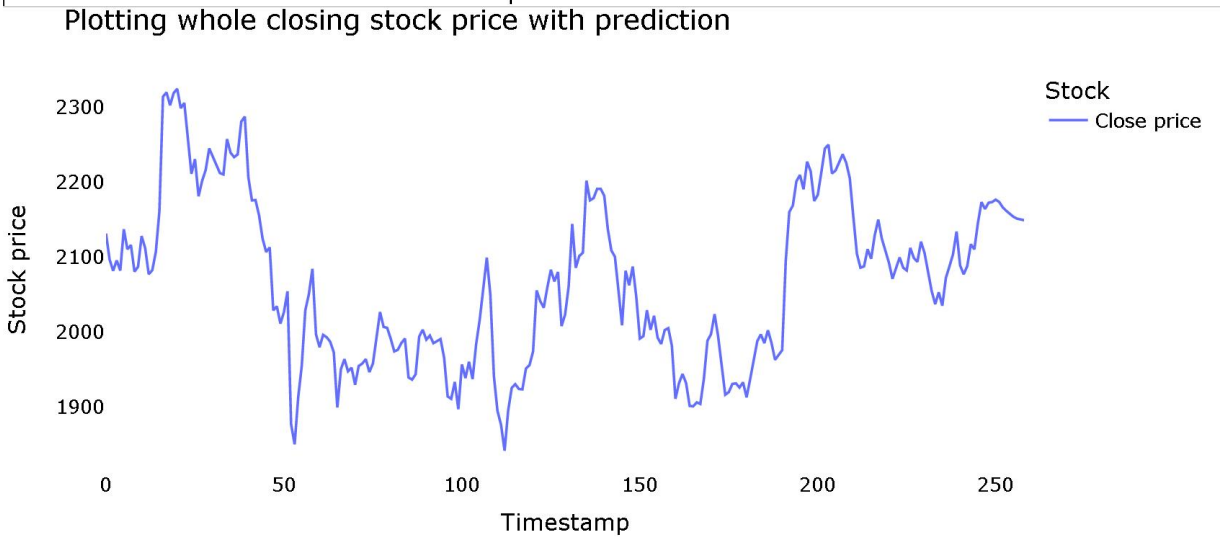
LSTM + GRU

```
# reshape input to be [samples, time steps, features] which is required for LSTM
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)

print("X_train: ", X_train.shape)
print("X_test: ", X_test.shape)
```

[90] ✓ 0.0s Python

```
X_train:  (145, 15, 1)
X_test:  (72, 15, 1)
```





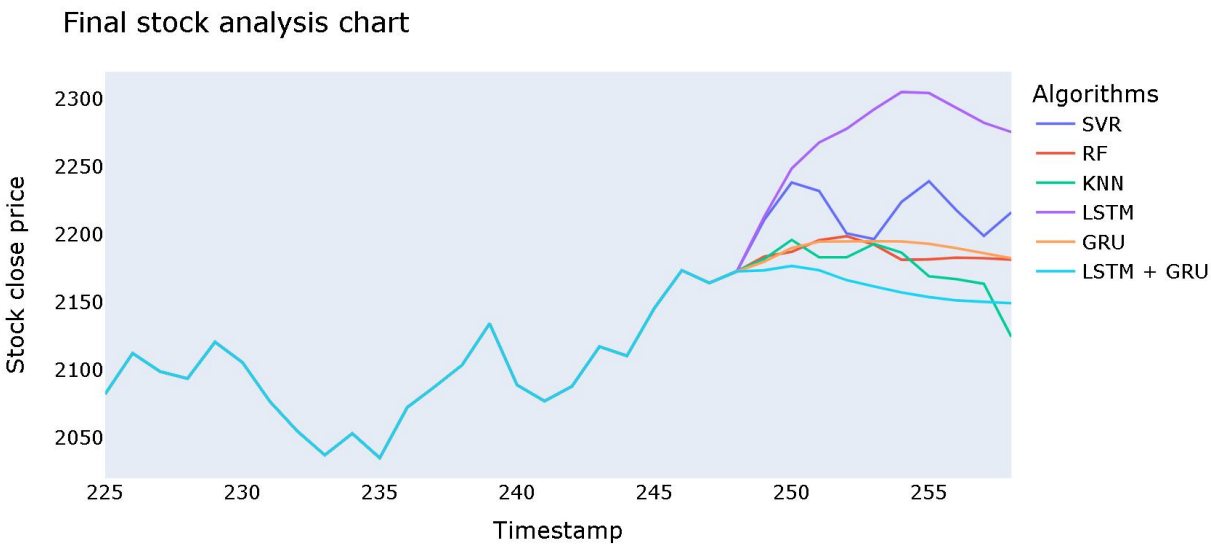
4.3 Result Analysis OR Screenshots

The resulting analysis of our project was that we were able to get a tabular and graphical representation of each and every prediction made via the different stock predictive models, we used:

```
finaldf = pd.DataFrame({
    'svr':svrdf,
    'rf':rfdf,
    'knn':knndf,
    'lstm':lstmdf,
    'gru':grudf,
    'lstm_gru':lstmgrudf,
})
finaldf.head()
```

[105] ✓ 0.0s Python

	svr	rf	knn	lstm	gru	lstm_gru
0	2131.550049	2131.550049	2131.550049	2131.550049	2131.550049	2131.550049
1	2097.050049	2097.050049	2097.050049	2097.050049	2097.050049	2097.050049
2	2081.850098	2081.850098	2081.850098	2081.850098	2081.850098	2081.850098
3	2095.750000	2095.750000	2095.750000	2095.750000	2095.750000	2095.750000
4	2082.100098	2082.100098	2082.100098	2082.100098	2082.100098	2082.100098



RELIANCE STOCK *CLOSE PRICE* PREDICTION

Total Rows = 249 and Columns = 7 Training data = 65% (161 rows) Testing Data = 35% (88 rows)

Algorithms	Main tuning Parameters (Hyper parameters)	RMSE		MSE		MAE		Variance Regression Score		R^2 score		Mean Gamma Deviance		Mean Poisson Deviance	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Super Vector Regressor	kernel = 'rbf' C = 0.001 gamma=0.1 degree= 3 epsilon=0.1	37.27	36.34	1389.39	1320.79	31.28	27.23	0.90	0.84	0.90	0.83	0.00032	0.00030	0.67	0.63
Random Forest	n_estimators= 100 random_state=O	15.94	36.39	254.20	1323.92	11.31	28.44	0.98	0.84	0.98	0.83	6.09874	0.00030	0.12	0.63
K-nearest Neighbor	n_neighbours = 15 metric='minkowski'	48.43	57.96	2345.29	3359.48	36.20	42.47	0.83	0.61	0.83	0.57	0.00054	0.00077	1.13	1.61
LSTM (epchs=200, Batch=5)	loss='mse' optimizer='adam' Three LSTM layers with 32 nodes	33.42	34.92	1116.76	1219.28	25.71	26.02	0.92	0.85	0.92	0.85	0.00026	0.00028	0.54	0.58
GRU (epchs=200, Batch=5)	loss='mse' optimizer='adam' Four GRU layers with 32 nodes	24.43	48.30	597.04	2332.68	18.72	36.05	0.96	0.71	0.96	0.70	0.00014	0.00053	0.59	1.12
LSTM + GRU (epchs=200, Batch= 5)	loss='mse' optimizer='adam' Two GRU layers with 32 nodes and two LSTM layers with 32 nodes	29.59	37.13	875.56	1378.43	21.97	27.93	0.94	0.83	0.94	0.83	0.00021	0.00031	0.42	0.66

4.4 Quality Assurance

In the end the prediction made by LSTM+GRU algorithm came out to be the closest for the next 10 days for the Reliance Stocks. However when it was tested again after a few days the system failed due to external factors, for instance, the start of IPL caused the Reliance stocks to fluctuate in a manner that was not predicted by any of the models we used.

To conclude the Quality of our program can only be assured so long there is no sudden external or environmental factor to affect the stock markets in a massive manner. Otherwise, the accuracy of each model was as follows:

1. LSTM+GRU: LSTM and GRU models are popular deep-learning models used for stock prediction. They can achieve high accuracy rates, with some studies reporting accuracy rates ranging from 70% to 95%.

2. KNN: K-Nearest Neighbors (KNN) is a simple machine-learning algorithm used for classification and regression tasks. Its accuracy can vary widely depending on the number of neighbors chosen and the distance metric used. In general, KNN is not as accurate as deep learning models, and its accuracy rates typically range from 50% to 70%.

3. SVR: Support Vector Regression (SVR) is a popular machine learning algorithm used for regression tasks. Its accuracy can vary widely depending on the kernel function used, the regularization parameter, and the choice of hyperparameters. In general, SVR can achieve accuracy rates ranging from 60% to 80%.

4. RF: Random Forest (RF) is a popular ensemble learning algorithm used for classification and regression tasks. Its accuracy can vary widely depending on the number of trees, the depth of the trees, and the choice of hyperparameters. In general, RF can achieve accuracy rates ranging from 60% to 80%.

Chapter 5

Standards Adopted

There were several data collection standards that can be followed while building a stock prediction system. Some of these standards are:

1) ISO 27001: This standard outlines the requirements for information security management systems, including the collection, storage, and processing of data. Adhering to this standard can help ensure the security and privacy of user data.
2) IEEE 802: This standard outlines the specifications for wireless local area network (WLAN) technology. If the stock prediction system involves collecting data through wireless networks, adhering to this standard can ensure the reliability and compatibility of the wireless network components.
3) ISO 9001: This standard outlines the requirements for quality management systems. Adhering to this standard can help ensure the accuracy and consistency of the collected data.
4) GDPR: The General Data Protection Regulation (GDPR) is a regulation that outlines the rules for the collection, use, and processing of personal data in the European Union (EU). If the stock prediction system collects data from EU citizens, adhering to this regulation is necessary to ensure the privacy and protection of personal data.
5) SEC regulations: If the stock prediction system involves the prediction of publicly traded companies, adhering to the regulations set forth by the Securities and Exchange Commission (SEC) is necessary to ensure the legality of the system.

By adhering to these data collection standards, the stock prediction system has ensured the security, privacy, accuracy, and legality of the collected data.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

In this paper, a review on various stock prediction techniques has been presented. On the basis of published and available literature, it can be safely concluded that the existing techniques are not suitable for prediction of stock market trends as well as price of different socks. There exist a gap between technologies and user requirement for a safe and accurate stock prediction system. If various political & economic factors which affect the stock market are also taken into consideration other than the technical indicators as input variables, better results may be obtained. Also, incorporating market specific domain knowledge into the system might help in achieving better performance. The prediction can be more accurate if the model will train with a greater number of data sets. Moreover, in the case of the prediction of various shares, there may be some scope for specific business analysis. We can study the different pattern of the share price of different sectors and can analyze a graph with more different time span to fine-tune the accuracy. This framework broadly helps in market analysis and prediction of the growth of different companies in different time spans. Incorporating other parameters (e.g. investor sentiment, election outcome, geopolitical stability) that are not directly correlated with the closing price may improve the prediction accuracy.

However, if external factors are put aside the system not only gives us an advanced analysis of predicted stock prices via different algorithms but also allows us to easy understand, visualize and choose which predictive algorithm to adhere to based on the accuracy level at different times. LSTM+GRU models are often more accurate for stock prediction than traditional machine learning algorithms like SVR, RF, and KNN because they are specifically designed to handle sequential data, are flexible and adaptable, can capture non-linear relationships, and can handle large amounts of data. However, it is important to note that the accuracy of any stock prediction model will depend on the quality and relevance of the input features, the amount and quality of the historical data, and the volatility of the financial markets.

6.2 Future Scope

Future scope of this project will involve adding more parameters and factors like the financial ratios, multiple instances, etc. The more the parameters are taken into account more will be the accuracy. The algorithms can also be applied for analyzing the contents of public comments and thus determine patterns/relationships between the customer and the corporate employee. The use of traditional algorithms and data mining techniques can also help predict the corporation performance structure as a whole. In the future, we plan to integrate neural network with some other techniques such as genetic algorithm or fuzzy logic. Genetic algorithm can be used to identify optimal network architecture and training parameters. Fuzzy logic provides the ability to account for some uncertainty produced by the neural network predictions. Their uses in conjunction with neural network could provide an improvement for stock market prediction.

The way things are progressing in the world of Machine Learning and Deep Learning, it is safe to assume that the day is not far when we will be able to logically code out even the external and environmental factors and incorperate it into our stock prediction system increasing the accuracy even more.

References

1) Stock Price Prediction Using LSTM on Indian Share Market by Achyut Ghosh, Soumik Bose1, GiridharMaji, Narayan C. Debnath, Soumya Sen

2) S. Selvin, R. Vinayakumar, E. A. Gopalkrishnan, V. K. Menon and K. P. Soman - Stock price prediction using LSTM, RNN, and CNN-sliding window model - 2017.

3) Upadhyay, A., Bandyopadhyay, G., Dutta, A., "Forecasting Stock Performance in Indian Market using Multinomial Logistic Regression", Journal of Business Studies Quarterly, Vol. 3, No. 3, pp. 16-39, 2012.

4) V Kranthi Sai Reddy Student, ECM, Sreenidhi Institute of Science and Technology, Hyderabad, India - Stock Market Prediction Using Machine Learning.

5) Pranav Bhat Electronics and Telecommunication Department, Maharashtra Institute of Technology, Pune. Savitribai Phule Pune University - A Machine Learning Model for Stock Market Prediction.

6) Thenmozhi, M., "Forecasting Stock Index Returns Using Neural Networks", Delhi Business Review, Vol. 7, No. 2, pp. 59-69

7) Majumder, M., Hussian, A., "Forecasting of Indian Stock Market Index Using Artificial Neural Network".

8) Giovanis, Eleftherios - Applications of Least Mean Square (LMS) Algorithm Regression in TimeSeriesAnalysis

9) Osman Hegazy, Omar S. Soliman and Mustafa Abdul Salam Faculty of Computers and Informatics, Cairo University, Egypt - A Machine Learning Model for Stock Market Prediction

10) Lufuno Ronald Marwala A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfillment of the requirements for the degree of Master of Science in Engineering - Forecasting the Stock Market Index Using Artificial Intelligence Techniques.

11) Mariam Moukalled Wassim El-Hajj Mohamad Jaber Computer Science Department the American University of Beirut - Automated Stock Price Prediction Using Machine Learning.

12) Choudhry, R., Garg, K., "A Hybrid Machine Learning System for Stock Market Forecasting", World Academy of Science, Engineering, and Technology, Vol. 15, pp. 315-318

13) V Kranthi Sai Reddy Student, ECM, Sreenidhi Institute of Science and Technology, Hyderabad, India - Stock Market Prediction Using Machine Learning.

14) Chen, Y., Abraham, A., Yang, J., Yang, B., "Hybrid Methods for Stock Index Modeling", Proceedings of the Second international conference on Fuzzy Systems and Knowledge Discovery

15) Upadhyay, A., Bandyopadhyay, G., Dutta, A., "Forecasting Stock Performance in Indian Market using Multinomial Logistic Regression", Journal of Business Studies Quarterly, Vol. 3, No. 3, pp. 16-39, 2012.

16) Sureshkumar, K. K., Elango, N. M., "An Efficient Approach to Forecast Indian Stock Market Price and their Performance Analysis", International Journal of Computer Application, Vol. 34, No.5, 2011.

17) Mehrara, M., Moeini, A., Ahrari, M., Ghafari, A., "Using Technical Analysis with Neural Network for Forecasting Stock Price Index in Tehran Stock Exchange" Middle Eastern Finance and Economics, Vol. 6, No. 6, pp. 50-61, 2010.

18) Agrawal, S., Jindal, M., Pillai, G. N., "Momentum Analysis based Stock Market Prediction using Adaptive Neuro-Fuzzy Inference System (ANFIS)", Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, Vol. 1, March 17 -19, 2010.

19) Majumder, M., Hussian, A., "Forecasting of Indian Stock Market Index Using Artificial Neural Network".

20) Ganatr, A., Kosta, Y. P., "Spiking Back Propagation Multilayer Neural Network Design for Predicting Unpredictable Stock Market Prices with Time Series Analysis", International Journal of Computer Theory and Engineering, Vol. 2, No. 6, 1793-8201, 2010.

21) Goswami, M. M., Bhensdadia, C. K., Ganatra, A. P., "Candlestick Analysis based Short Term Prediction of Stock Price Fluctuation using SOM-CBR", IEEE International Advance Computing Conference, Patiala, India, pp. 1448 – 1452, 6-7 March 2009.

22) Jaaman, S. H., Shamsuddin, S. M., Yusob, B., Ismail, M., "A Predictive Model Construction Applying Rough Set Methodology for Malaysian Stock Market Returns", International Research Journal of Finance and Economics, No. 30, pp. 211-218, 2009.

23) Atsalakis, G. S., Valavanis, K. P., "Surveying stock market forecasting techniques – Part II: Soft computing Methods", Expert Systems with Applications, Vol. 36, pp. 5932-5941, 2009.

24) Chaigusin, S., Chirathamjaree, C., Clayden J., "Soft computing in the forecasting of the Stock Exchange of Thailand (SET)", Proceedings of IEEE Conference on Management of Innovation and Technology, pp. 1277-1281, 2008.

25) Thenmozhi, M., "Forecasting Stock Index Returns Using Neural Networks", Delhi Business Review, Vol. 7, No. 2, pp. 59-69, July - December 2006. [12] Yang, X., Jiangxi, "The Prediction of Stock Prices Based on PCA and BP Neural Networks", Chinese Business Review, Vol. 4, No.5, pp. 64- 68, May 2005.

26) Klassen, M., Investigation of Some Technical Indexes in Stock Forecasting Using Neural Networks, World Academy of Science, Engineering and Technology, Vol. 5, pp. 75-79, 2005.

27) Phua, P. K. H., Zhu, X. T., Chung, H. K., "Forecasting Stock Index Increments Using Neural Networks with Trust Region Methods", Proceedings of the International Joint Conference on Neural Networks, vol. 1, pp. 260-265, 2003.

26) Chen, Y., Abraham, A., Yang, J., Yang, B., "Hybrid Methods for Stock Index Modeling", Proceedings of the Second international conference on Fuzzy Systems and Knowledge Discovery, Volume Part II, pp. 1067-1070, 2005.

27) Tsai, C. F., Wang, S. P., "Stock Price Forecasting by Hybrid Machine Learning Techniques", Proceedings of the International Multi-Conference of Engineers and Computer Scientists, Hong Kong, Vol. I, March 18 - 20, 2009. [17] Huang, W., Nakamori, Y., Wang, S. Y., "Forecasting stock market movement direction with support vector machine", Computers & Operations Research, Vol. 32, No. 10, pp. 2513-2522, 2004.

28) Vanstone, B., Tan, C., "A Survey of the Application of Soft Computing to Investment and Financial Trading", Information Technology Papers, Vol. 1, No. 1, pp. 211-216, 2003.

29) Choudhry, R., Garg, K., "A Hybrid Machine Learning System for Stock Market Forecasting", World Academy of Science, Engineering and Technology, Vol. 15, pp. 315-318, 2008.

30) Lin, J., Li, Y., Liu, C., "Building Time Series Forecasting Model by Independent Component Analysis Mechanism", Proceedings of the World Congress on Engineering, London, U.K, Vol. II, July 2 - 4, 2007.

ADVANCED STOCK PREDICTION SYSTEM

ABHISHEK DUTTA 2005072
VYVYAAN DOGRA 20051536
NAMAN SETHI 2005109
PRATHAM SADHWANI 2005818

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

Individual contribution and findings:

Abhishek Dutta: Code Integration, formatting, inundation, error removal, research for dataset and structuring model for the entire code implementation.
Naman Sethi: Coding entire Support Vector Regression (SVR), Random Forest Regression Mode l(RF), K-Nearest Neighbour (KNN).
Vyvyaan Dogra: Coding of Long Short Term Memory Network (LSTM) and collaborating with Pratham to Implement LSTM+GRU algorithm.
Pratham Sadhwani: Coding of Gated Recurrent Units (GRU) and collaborating with Vyvyaan to Implement LSTM+GRU algorithm.

Individual contribution to project report preparation:

Abhishek Dutta: Entire Section 4-Implementation and document formatting
Naman Sethi: Entire Section 1-Introduction and 2-Basic Concepts/Literature Review
Vyvyaan Dogra: Entire Section 3-Problem Statement/Requirement Specifications
Pratham Sadhwani: Entire Section 5-Standards Adopted and 6-Conclusion and Future Scope

Individual contribution to project presentation and demonstration:

Abhishek Dutta: Slides-10,11,12,13
Naman Sethi: Slides-6,7,8,9
Vyvyaan Dogra: Slides-2,3,4,5
Pratham Sadhwani: Slides-1,14,15

Full Signature of Supervisor: Full signature of the student:
……. ……..

TURNITIN PLAGIARISM REPORT