STOCK MARKET PRICE PREDICTION

Mini Project report submitted
in partial fulfillment of requirements
for the award of degree of

Bachelor of Technology
In
Information Technology

By

D.DIVYA	**(Reg No: 17131A1227)**
K.BHAVYA	**(Reg No: 17131A1248)**
K.HARSHA CHAITANYA	**(Reg No: 17131A1256)**
S.JAHNAVI SAI SINDHURA	**(Reg No: 17131A12A6)**

Under the esteemed guidance of
Mr.P.Praveen Kumar
(M.Tech, Assistant Professor)
Department of Information Technology
GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (AUTONOMOUS)
(Affiliated to JNTU-K, Kakinada)
VISAKHAPATNAM
2020 – 2021

Gayatri Vidya Parishad College of Engineering (Autonomous)

Visakhapatnam

CERTIFICATE

This report on **"STOCK MARKET PRICE PREDICTION"** is a bonafide record of the mini project work submitted

By

D.DIVYA	**(Reg No: 17131A1227)**
K.BHAVYA	**(Reg No: 17131A1248)**
K.HARSHA CHAITANYA	**(Reg No: 17131A1256)**
S.JAHNAVI SAI SINDHURA	**(Reg No: 17131A12A6)**

in their VI semester in partial fulfillment of the requirements for the Award of Degree of

Bachelor of Technology

In

Information Technology

During the academic year 2020-2021

Mr.P.Praveen Kumar

Project Guide

Dr.K.B.Madhuri

Head of the Department

Department of information technology

DECLARATION

we here by declare that this industry oriented mini project entitled **“STOCK MARKET PRICE PREDICTION”** is a bonafide work done by us and submitted to **Department of Information Technology G.V.P college of engineering (autonomous) Visakhapatnam,** in partial fulfilment for the award of the degree of B.Tech is of our own and it is not submitted to any other university or has been published any time before.

PLACE: VISAKHAPATNAM

DATE :

D.DIVYA(Reg No: 17131A1227)
K.BHAVYA(Reg No: 17131A1248)
K.HARSHA CHAITANYA(Reg No: 17131A1256)
S.JAHNAVI SAI SINDHURA(Reg No: 17131A12A6)

ACKNOWLEDEGEMENTS

We would like to take this opportunity to extend our hearty gratitude to our esteemed institute **“Gayatri Vidya Parishad College of Engineering (Autonomous)"** where we got the platform to fulfill our cherished desire.

We express our sincere thanks to **Dr. A.B. KOTESWARA RAO,** PRINCIPAL, Gayatri Vidya Parishad College of Engineering (Autonomous), for his support and encouragement during the course of this project.

We express our deep gratitude to **Dr. K.B.MADHURI,** Head of the Department of Information Technology, Gayatri Vidya Parishad College of Engineering (Autonomous) for her constant support and encouragement.

We are obliged to **Mr.P.PRAVEEN KUMAR,** Assistant Professor, Department of Information Technology, who has been our guide, whose valuable suggestions, guidance and comprehensive assistance helped us a lot in realizing the project.

We also thank **Mr. D. NAGATEJ,** Assistant Professor**,** Project Coordinator, Department of Information Technology, for guiding us throughout the project and helping us in completing the project efficiently.

We would like to thank all the members of teaching and non-teaching staff of Department of Information Technology, for all their support Lastly, we are grateful to all our friends, for their relentless support in augmenting the value of work, our family, for being considerate and appreciative throughout.

D.DIVYA(Reg No: 17131A1227)
K.BHAVYA(Reg No: 17131A1248)
K.HARSHA CHAITANYA(Reg No: 17131A1256)
S.JAHNAVI SAI SINDHURA(Reg No: 17131A12A6)

ABSTRACT

STOCK MARKET PRICE PREDICTION is a web application.It is used to determine the future value of a company stock or other financial instrument traded on a financial exchange.The successful prediction of a stock's future price will maximize investor's gains. There are so many factors involved in the prediction such as physical factors, rational and irrational behaviour.All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy. We will implement a mix of machine learning algorithms to predict the future stock price of this company, starting with simple algorithms like multiple linear regression,Decision tree,Random Forest and then move on to advanced techniques like XGBoost algorithm.In this project we used Nifty50 dataset from NSE(National Stock Exchange)India.At the end of this project we compare all the models and take the best fit model.

INDEX

1.INTRODUCTION

The stock (also capital stock) of a corporation constitutes the equity stake of its owners. It represents the residual assets of the company that would be due to stockholders after discharge of all senior claims such as secured and unsecured debt.Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit.

Basically,quantitative traders with a lot of money from stock markets buy stocks derivatives and equities at a cheap price and later on selling them at high price.There are two types to analyze stocks which investors perform before investing in a stock,first is the fundamental analysis,in this analysis investors look at the intrinsic value of stocks and performance of the industry,economy,political climate.To decide that whether invest or not.On the other hand,the technical analysis,it is an evolution of stocks by the means of studying generated by market activity,such as past prices and volumes.

1.1 Objective

There is an increasing interest in predicting markets among economists,policymakers, academics and market makers.The objective of the proposed work is to study andimprove the supervised learning algorithms to predict the stock price.It will be implemented in Regression.The system must be able to access a list of historical prices.It must also provide an instantaneous visualization of the market index.

1.2 About the project

In this project, we have developed our accurate model using various regression algorithms such as Multiple Linear Regression,Decision tree,Random Forest,XGBoost Regression to predict the stock market price.Our main goal is to predict the accurate close price of stocks.We developed our models in Jupyter Lab and deployed using a flask server.

1.3 Purpose

Before an investor invests in any stock,he needs to be aware how the stock market behaves.Investing in a good stock at a bad time can have disastrous results,while investment in a mediocre stock at the right time can bear profits.Financial investors of today are facing this problem of trading.So predicting long term value of the stock is relatively easy than predicting on day-to-day basis as the stock fluctuate rapidly every hour based on world events.

1.4 Scope

This project requires investigation in the following areas:
a)Investigating trends in stock market and factors affecting the stock prices.
b)Investigating the available tools and techniques for data mining and then selecting
those that are the best fit to solve the problem.

2. SRS DOCUMENT

2.1 Functional Requirements:

A functional requirement defines a function of a system or it’s component. A function can be described as set of inputs, the behavior, and outputs. It also depends upon the type of software, expected users and the type of system where the software is used.

The functional requirements in our project are:

- The values in the dataset (Open,High,Low) are given as input to our model.
- Based on the values we predict the Close Value in Stock Market.
- Our model is trained using Real values so that that it can be used in real time to predict the close price of the stock market.

2.1.1 Software Requirements:

- Operating system: Windows 10
- Programming Language: Python 3.6
- Anaconda navigator
- Dependencies
    - Numpy
    - Pandas
    - Sklearn

- Flask

2.1.2 Hardware Requirements:

- Processor: Intel processor
- RAM:8GB
- Disk space: 1 TB
- CPU:4GB

2.2 Non Functional Requirements:

- Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc. Apply to the system as whole.
- Non-functional requirements in this system are :

1.Reliability: Reliability based on this system defines the evaluation result of the system.

2.Ease of Use: The system is simple, user friendly, so any can use this system without any difficulties.

3.ALGORITHM ANALYSIS

3.1 Existing System:

Sales forecasting involves the determination of the expected levels of sales in the future, based on past and present sales data, the intentions of management and environmental influences upon the enterprise. The forecasting of sales can be regarded as a system having inputs, a process and an output. Stock market price prediction is previously implemented using **Linear Regression and Logistic Regression.**

3.2 Proposed System:

Various algorithms are used to predict highly accurate results. All the algorithms that are used in our system are described below.

Multiple Linear Regression:

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

Decision Tree Regression:

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

Random Forest Regression:

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

XGBoost Regression:

XGBoost(Extreme Gradient Boosting) is an algorithm used for structured or tabular data.It is an implementation of gradient boosted decision trees designed for speed and performance.It is a software library that you can download and install on your machine, then access from a variety of interfaces.XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework.

- We got the minimal error with multiple linear regression .

3.3 Feasibility Study

Feasibility analysis begins once the goals are defined. It starts by generating broad possible solutions, which are possible to give an indication of what the new system should look like. This is where creativity and imagination are used. Analysts must think up new ways of doing things- generate new ideas. There is no need to go into the detailed system operation yet. The solution should provide enough information to make reasonable estimates about project cost and give users an indication of how the new system will fit into the organization. It is important not to exert considerable effort at this stage only to find out that the project is not worthwhile or that there is a need significantly change the original goal. Feasibility of a new system means ensuring that the new system, which we are going to implement, is efficient and affordable.

There are various types of feasibility to be determined. They are

Economically Feasibility: Development of this application is highly economically feasible. The result obtained contains minimum errors and are highly accurate.

Technical feasibility: The technical requirement for the system is economic and it does not use any other additional Hardware and software.

Operational Feasibility: The system is quite easy to use and learn due to its simple but attractive interface. User requires no special training for operating the system.

3.4 Cost benefit analysis

The financial and the economic questions during the preliminary investigation are verified to estimate the following:

- The cost of the hardware and software for the class of application being considered.
- The benefits in the form of reduced cost.
- The proposed system will give the minute information, as a result.
- Performance is improved which in turn may be expected to provide increased profits.
- This feasibility checks whether the system can be developed with the available funds.
- This can be done economically if planned judicially, so it is economically feasible.
- The cost of project depends upon the number of man-hours required.

4.SOFTWARE DESCRIPTION

4.1 Anaconda-Navigator

The open-source Anaconda Individual Edition (formally Anaconda Distribution) is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 19 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling individual data scientists to:

- Quickly download 7,500+ Python/R data science packages.
- Manage libraries, dependencies, and environments with Conda.
- Develop and train machine learning and deep learning models with scikit-learn, TensorFlow.
- Analyze data with scalability and performance with Dask, NumPy, Pandas, and Numba.
- Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews.

4.2 Flask

Flask (source code) is a Python web framework built with a small core and easy-to-extend philosophy. Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks. Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy.

4.3 Sklearn

Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like Numpy and Scipy.

The sklearn library contains lot of efficient machine learning tools including classification, regression, clustering, and dimensionality reduction. It should not be used for reading the data manipulating the data and summarizing it. There are better libraries for that like Numpy , Pandas etc

4.4 Pickle

Python Pickle implements binary protocols for serializing ad deserializing a python object structure. Any object in python can be pickled so that it can be saved on disk. Pickling is the way to convert a python object into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

5.PROJECT DESCRIPTION

5.1 Problem Definition:

The Stock Market prediction task is interesting as well as divides researchers and academics into two groups those who believe that we can devise mechanisms to predict the market and those who believe that the market is efficient and whenever new information comes up the market absorbs it by correcting itself, thus there is no space for prediction.Develop a machine learning model to predict the stock prices of a company.

5.2 Project Overview:

Our project contains 3 modules, first module is inputting the dataset and splitting dataset and second one is choosing one of the regression algorithm and apply it to train, test and calculate Mean Square Error(MSE),Mean Absolute Error(MAE) and Root Mean Square Error(RMSE) of the model whereas third module is of deploying the model with the help of flask framework as a web application.

5.3 Mechanism Description

5.3.1 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine Learning focuses on the development of computer programs that can access data and use it themselves. The process of learning begins with observations or data, such as examples, direct experience or instruction in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim to allow the computers learn automatically without human intervention assistance and adjust accordingly. But, using the classic algorithms of machine learning, text is considered as a sequence of keywords ; instead and approach based on semantic analysis mimics the human ability to understand the meaning of a text.

5.3.2 Algorithm Description

Multiple Linear Regression:

Multiple Linear Regression attempts to model the relationship between two or more features and a response by fitting a linear equation to observed data. The steps to perform multiple linear Regression are almost similar to that of simple linear Regression. The Difference Lies in the evaluation. We can use it to find out which factor has the highest impact on the predicted output and now different variable relate to each other.

Decision Tree:

It is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs and utility.

Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

The branches/edges represent the result of the node and the nodes have either:

1. Conditions [Decision Nodes]
2. Result [End Nodes]

The branches/edges represent the truth/falsity of the statement and takes makes a decision based on that in the example below which shows a decision tree that evaluates the smallest of three numbers:

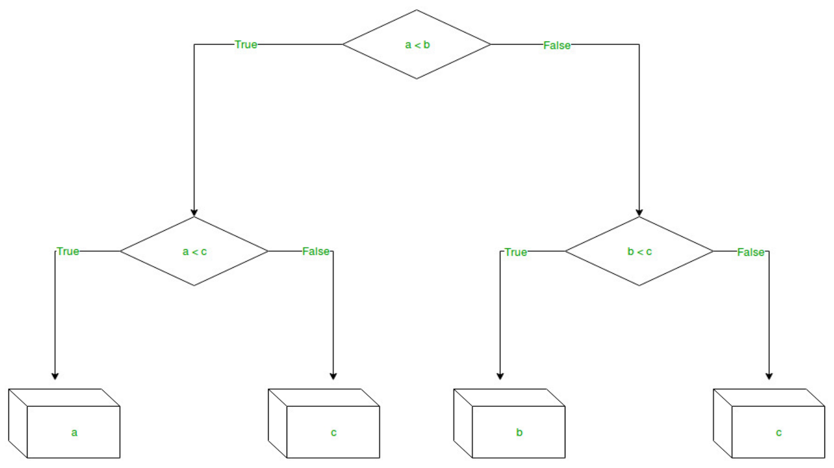


Fig:Decision Tree

Random Forest Algorithm

Stock Market Price Prediction is a Regression problem which is solved using Random Forest. Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

Working of Random Forest Algorithm:

- **Step 1** − First, start with the selection of random samples from a given data set.
- **Step 2** − Next, this algorithm will construct a decision tree for every sample.

  Then it will get the prediction result from every decision tree.
- **Step 3** − In this step, voting will be performed for every predicted result.
- **Step 4** − At last, select the most voted prediction result as the final prediction result.

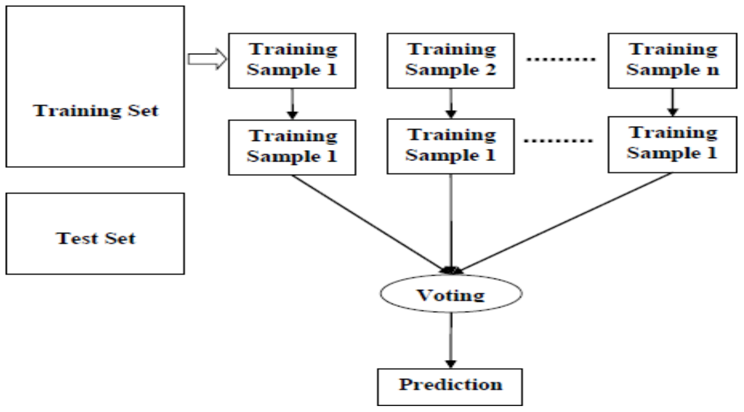


Fig:RANDOM FOREST

XGBoost Regression:

XGBoost is an implementation of Gradient Boosted decision trees. This library was written in C++. It is a type of Software library that was designed basically to improve speed and model performance. It has recently been dominating in applied machine learning.

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. Weight of variables predicted wrong by the tree is increased and these the variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

5.4 Flask Description:

Flask is a web framework. It means that flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks. Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy.

FLOWCHART

Our project contains 3 modules, first module is inputting the dataset and splitting dataset and second one is choosing one of the regression algorithm and apply it to train, test and calculate Mean Square Error(MSE),Mean Absolute Error(MAE) and Root Mean Square Error(RMSE) of the model whereas third module is of deploying the model with the help of flask framework as a web application

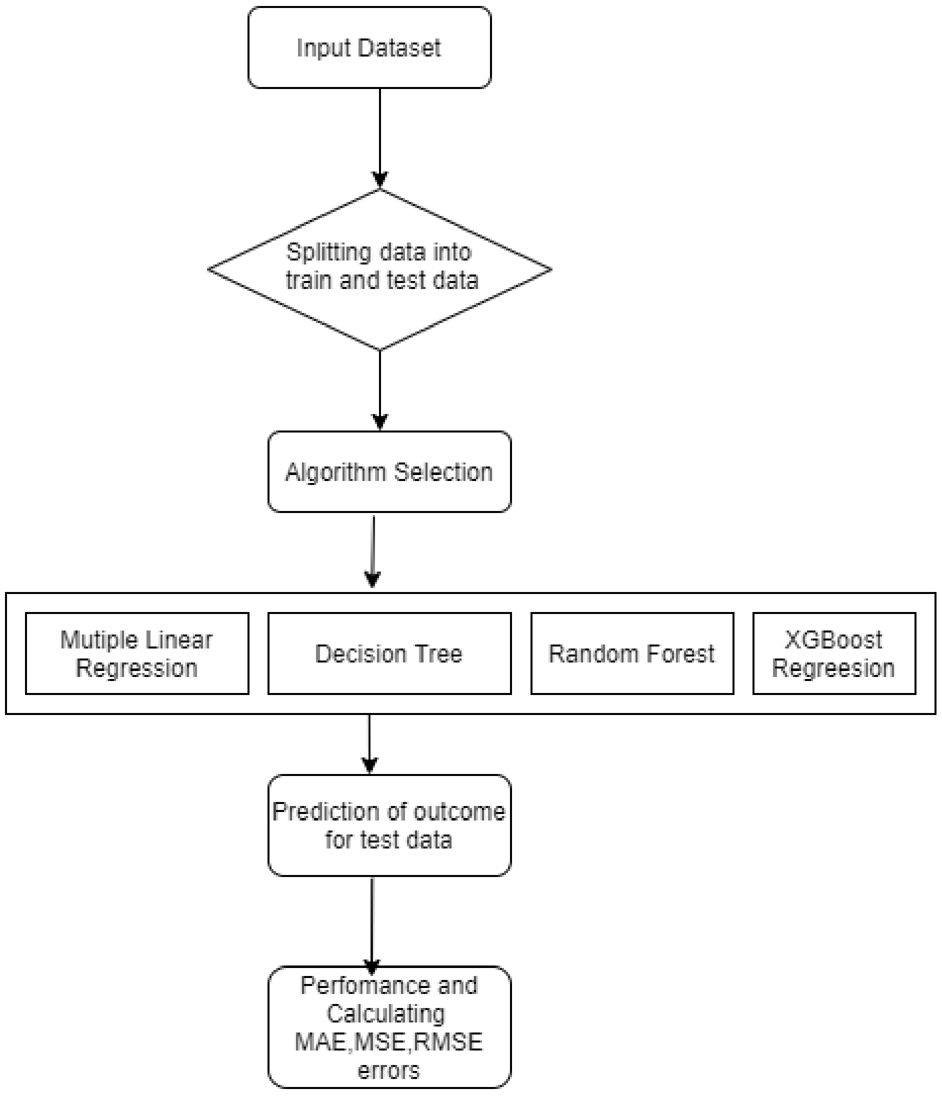

Input Dataset
Splitting data into train and test data
Algorithm Selection
Mutiple Linear Regression
Decision Tree
Random Forest
XGBoost Regreesion
Prediction of outcome for test data
Perfomance and Calculating MAE,MSE,RMSE errors

Fig: Flow Chart depicting the modules

6.SYSTEM DESIGN

6.1 Introduction to UML:

The system to be developed is best designed using UML i.e. Unified Modeling Language. The Unified Modeling Language includes a set of graphics notation techniques to create visual models of object-oriented intensive systems. UML is a visual language for specifying, constructing, and documenting the artifacts of software- intensive systems.

Complex software designs difficult for you to describe with text alone can readily be conveyed through diagrams using UML. You can use UML with all processes throughout the development life cycle and across different implementation.

A model is a simplification of reality. We build models so that we can better understand the system we are developing. Through modeling, we achieve four aims. They are:

1. Models help us to visualize a system as it is or as we want it to be.
2. Models permit us to specify the structure or behavior of a system.
3. Models give us a template that guides us in constructing a system.
4. Models document the decisions we have made. We build models of complex systems because we cannot comprehend such a System in its entirety.

6.2 Building Blocks of UML:

The vocabulary of the UML encompasses three kinds of building blocks:

1. Things

2. Relationships
3. Diagrams

- Things are the abstractions that are first-class citizens in a model;
- Relationships tie these things together;
- Diagrams group interesting collections of things;

❖ Things in the UML:

There are four kinds of things in the UML:

1. Structural things
2. Behavioral things
3. Grouping things
4. Annotational things

6.2.1 Structural Things:

Structural things are the nouns of UML models. These are the mostly static parts of a model, representing elements that are either conceptual or physical. In all, there are seven kinds of structural things.

- A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics. A class implements one or more interfaces. Graphically, a class is rendered as a rectangle, usually including its name, attributes, and operations.
- An interface is a collection of operations that specify a service of a class or component. An interface therefore describes the externally visible behavior of that element. An interface might represent the complete behavior of a class or component or only a part of that behavior.
- Collaboration defines an interaction and is a society of roles and other elements that work together to provide some cooperative behavior that's bigger than the sum of all the elements. Therefore, collaborations have structural, as well as behavioral, dimensions. Graphically, a

collaboration is rendered as an ellipse with dashed lines, usually including only its name.

- Use case is a description of set of sequence of actions that a system performs that yields an observable result of value to a particular actor. A use case is used to structure the behavioral things in a model. Graphically, a use case is rendered as an ellipse with solid lines, usually including only its name.
- An active class is a class whose objects own one or more processes or threads and therefore can initiate control activity. An active class is just like a class except that its objects represent elements whose behavior is concurrent with other elements. Graphically, an active

class is rendered just like a class, but with heavy lines, usually including its name, attributes, and operations.

- A component is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces. Graphically, a component is rendered as a rectangle with tabs, usually including only its name.
- A node is a physical element that exists at run time and represents a computational resource, generally having at least some memory and, often, processing capability. A set of components may reside on a node and may also migrate from node to node. Graphically, a node is rendered as a cube, usually including only its name.

6.2.2. Behavioral Things:

- Behavioral things are the dynamic parts of UML models. These are the verbs of a model, representing behavior over time and space. In all, there are two primary kinds of behavioral things.
- An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose.
- A state machine is a behavior that specifies the sequences of states an object or an interaction

goes through during its lifetime in response to events, together with its responses to those events. A state machine involves a number of other elements, including states, transitions (the flow from state to state), events (things that trigger a transition), and activities (the responseto a transition). Graphically, a state is rendered as a rounded rectangle, usually including its name and its substrates, if any.

6.2.3. Grouping Things:

- Grouping things are the organizational parts of UML models. These are the boxes into which a model can be decomposed. In all, there is one primary kind of grouping thing, namely, packages.
- A package is a general-purpose mechanism for organizing elements into groups.
- Structural things, behavioral things, and even other grouping things may be placed in a package. Unlike components, a package is purely conceptual. Graphically, a package is rendered as a tabbed folder, usually including only its name and, sometimes, its contents.

6.2.4. Annotational Things:

- Annotational things are the explanatory parts of UML models. These are the comments you may apply to describe, illuminate, and remark about any element in a model. There is one primary kind of annotational thing, called a note.
- A note is simply a symbol for rendering constraints and comments attached to an element or a collection of elements. Graphically, a note is rendered as a rectangle with a dog-eared corner, together with a textual or graphical comment.

Relationships in the UML:

There are four kinds of relationships in the UML:

1. Dependency

2. Association

3. Generalization

4. Realization

These relationships are the basic relational building blocks of the UML. We use them to write well-formed models.

- A dependency is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).
- An association is a structural relationship that describes a set of links, a link being a connection among objects.
- A generalization is a specialization/generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element (the parent).
- A realization is a semantic relationship between classifiers, wherein one classifier specifies a contract that another classifier guarantees to carry out.

Diagrams in the UML:

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships). You draw diagrams to visualize a system from different perspectives, so a diagram is a projection into a system. UML includes nine diagrams.

- A class diagram shows a set of classes, interfaces, and collaborations and their relationships. These diagrams are the most common diagram found in modeling object-oriented systems.

Class diagrams address the static design view of a system. Class diagrams that include active classes address the static process view of a system.

- An object diagram shows a set of objects and their relationships. Object diagrams represent static snapshots of instances of the things found in class diagrams. These diagrams addressthe static design view or static process view of a system as do class diagrams, but from the perspective of real or prototypical case

- A use case diagram shows a set of use cases and actors (a special kind of class) and their relationships. Use case diagrams address the static use case view of a system. These diagrams are especially important in organizing and modeling the behaviour of a system.

- Both sequence diagrams and collaboration diagrams are kinds of interaction diagrams. An interaction diagram shows an interaction, consisting of a set of objects and their relationships, including the messages that may be dispatched among them. Interaction diagrams address the dynamic view of a system. A sequence diagram is an interaction diagram that emphasizes the time-ordering of messages; a collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. Sequence diagrams and collaboration diagrams are isomorphic, meaning that we can take one and transform it into the other.
- A state chart diagram shows a state machine, consisting of states, transitions, events, and activities. State chart diagrams address the dynamic view of a system.

6.3 UML Diagrams

6.3.1 Use Case Diagram:

A use case diagram is a type of behavioral diagram defined in UML and creates from use case analysis. The main purpose of a use case diagram is to show what the system functions are performed for which actors. Roles of actors in the system can be depicted. Here user can play any media using media player when web camera detects user is looking at the system it plays media.

6.3.2 Class Diagram:

A class diagram shows a set of classes, interfaces and collaborations and their relationships. These diagrams are the most common diagrams found in modelling object-oriented systems. Class diagrams address the static design view of a system. Class diagrams that include active classes address the static process view of a system.

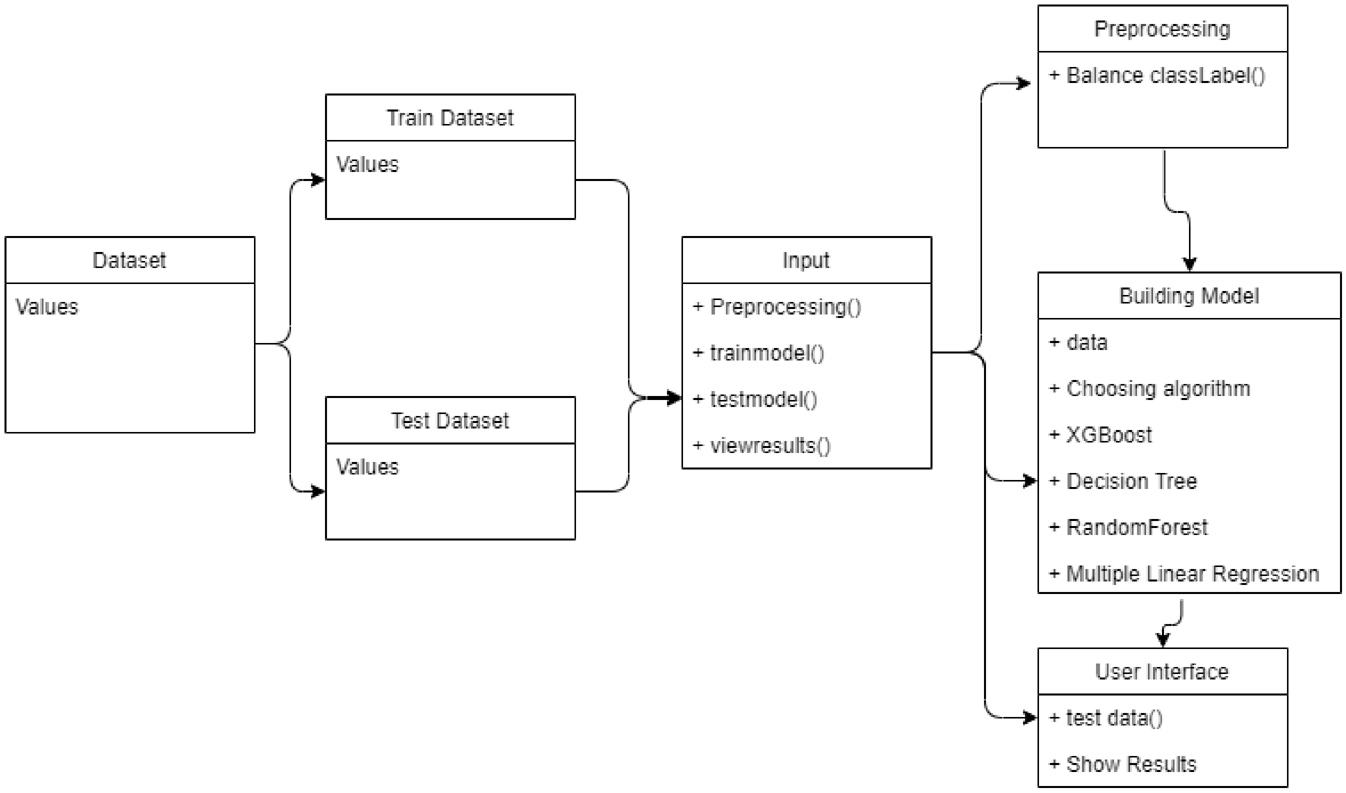
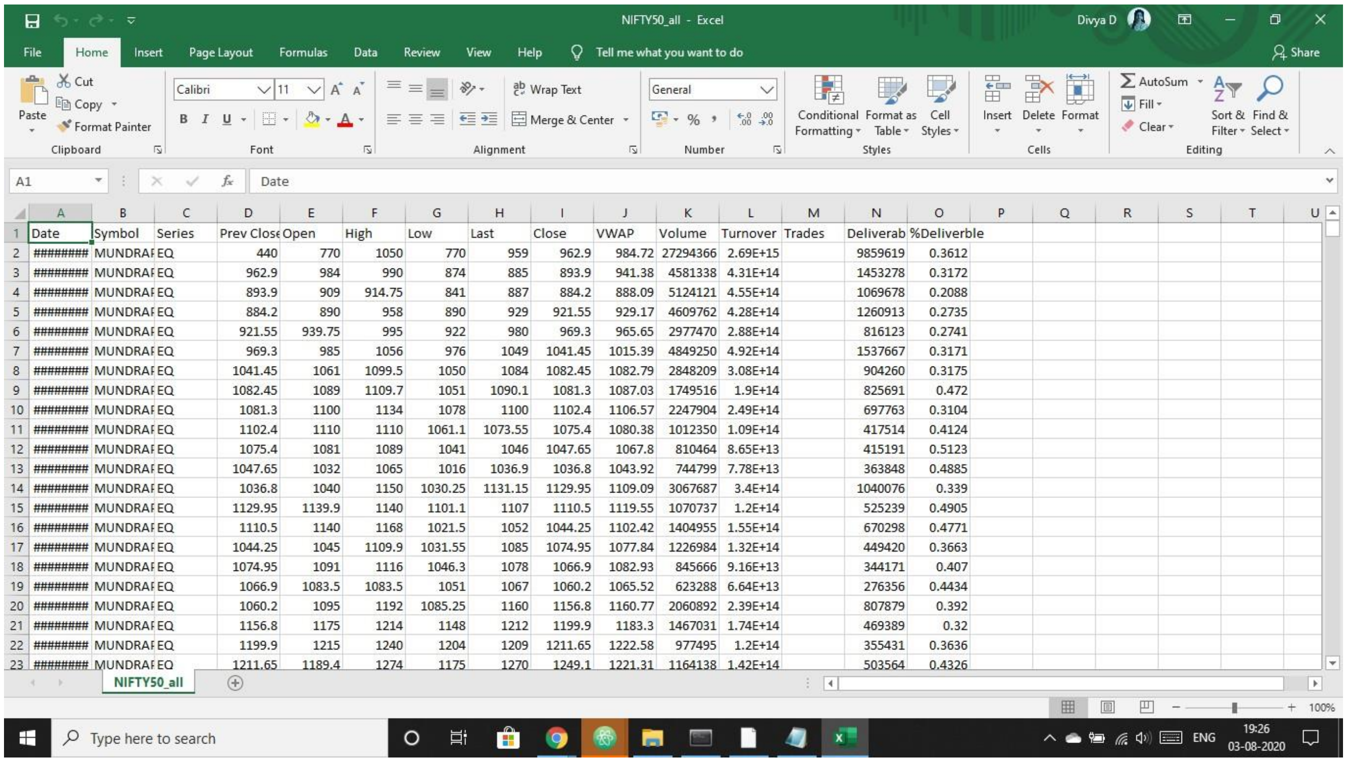


Fig:Class Diagram

7.DEVELOPMENT

7.1 Dataset Used:

The dataset we used contains 223422 Rows and 15 Columns which is available in Kaggle Datasets.The data is the price history and trading volumes of the fifty stocks in the index NIFTY 50 from NSE (National Stock Exchange) India. All datasets are at a day-level with pricing and trading values split across .cvs files for each stock along with a metadata file with some macro-information about the stocks itself. The data spans from 1st January, 2000 to 31st July, 2020.



Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverab	%Deliverble
########	MUNDRAF	EQ	440	770	1050	770	959	962.9	984.72	27294366	2.69E+15		9859619	0.3612
########	MUNDRAF	EQ	962.9	984	990	874	885	893.9	941.38	4581338	4.31E+14		1453278	0.3172
########	MUNDRAF	EQ	893.9	909	914.75	841	887	884.2	888.09	5124121	4.55E+14		1069678	0.2088
########	MUNDRAF	EQ	884.2	890	958	890	929	921.55	929.17	4609762	4.28E+14		1260913	0.2735
########	MUNDRAF	EQ	921.55	939.75	995	922	980	969.3	965.65	2977470	2.88E+14		816123	0.2741
########	MUNDRAF	EQ	969.3	985	1056	976	1049	1041.45	1015.39	4849250	4.92E+14		1537667	0.3171
########	MUNDRAF	EQ	1041.45	1061	1099.5	1050	1084	1082.45	1082.79	2848209	3.08E+14		904260	0.3175
########	MUNDRAF	EQ	1082.45	1089	1109.7	1051	1090.1	1081.3	1087.03	1749516	1.9E+14		825691	0.472
########	MUNDRAF	EQ	1081.3	1100	1134	1078	1100	1102.4	1106.57	2247904	2.49E+14		697763	0.3104
########	MUNDRAF	EQ	1102.4	1110	1110	1061.1	1073.55	1075.4	1080.38	1012350	1.09E+14		417514	0.4124
########	MUNDRAF	EQ	1075.4	1081	1089	1041	1046	1047.65	1067.8	810464	8.65E+13		415191	0.5123
########	MUNDRAF	EQ	1047.65	1032	1065	1016	1036.9	1036.8	1043.92	744799	7.78E+13		363848	0.4885
########	MUNDRAF	EQ	1036.8	1040	1150	1030.25	1131.15	1129.95	1109.09	3067687	3.4E+14		1040076	0.339
########	MUNDRAF	EQ	1129.95	1139.9	1140	1101.1	1107	1110.5	1119.55	1070737	1.2E+14		525239	0.4905
########	MUNDRAF	EQ	1110.5	1140	1168	1021.5	1052	1044.25	1102.42	1404955	1.55E+14		670298	0.4771
########	MUNDRAF	EQ	1044.25	1045	1109.9	1031.55	1085	1074.95	1077.84	1226984	1.32E+14		449420	0.3663
########	MUNDRAF	EQ	1074.95	1091	1116	1046.3	1078	1066.9	1082.93	845666	9.16E+13		344171	0.407
########	MUNDRAF	EQ	1066.9	1083.5	1083.5	1051	1067	1060.2	1065.52	623288	6.64E+13		276356	0.4434
########	MUNDRAF	EQ	1060.2	1095	1192	1085.25	1160	1156.8	1160.77	2060892	2.39E+14		807879	0.392
########	MUNDRAF	EQ	1156.8	1175	1214	1148	1212	1199.9	1183.3	1467031	1.74E+14		469389	0.32
########	MUNDRAF	EQ	1199.9	1215	1240	1204	1209	1211.65	1222.58	977495	1.2E+14		355431	0.3636
########	MUNDRAF	EQ	1211.65	1189.4	1274	1175	1270	1249.1	1221.31	1164138	1.42E+14		503564	0.4326

Fig:Nifty Dataset

7.2 Source Code

```
# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd


 # Importing the dataset

dataset = pd.read_csv(r'F:/Projects/stock market prediction/NIFTY50_all.csv')

X = dataset.iloc[:, [4,5,6]].values

y = dataset.iloc[:, 8].values

print(dataset.shape)

print(X)

print(X.shape)

print(y)

print(y.shape)

# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)


y_train.shape

X_test.shape

y_test.shape

X_train.shape
```

1. Multiple Linear Regression

```
from sklearn.linear_model import LinearRegression
regressor1=LinearRegression()
regressor1.fit(X_train,y_train)
```

```
y_pred1=regressor1.predict(X_test)
print(y_pred1)
print(y_test)
```

Calculating Errors

```
#Mean Squared Error
from sklearn.metrics import mean_squared_error
MSE1=mean_squared_error(y_test,y_pred1)
print(MSE1)

#Root Mean Squared Error
from math import sqrt
RMSE1 = sqrt(MSE1)
print(RMSE1)

#Mean Absolute Error
from sklearn.metrics import mean_absolute_error
MAE1=mean_absolute_error(y_test, y_pred1)
print(MAE1)
```

2. Decision Tree Regression

```
#fitting decision tree into dataset
from sklearn.tree import DecisionTreeRegressor
regressor2=DecisionTreeRegressor(random_state=0)
regressor2.fit(X_train,y_train)

y_pred2=regressor2.predict(X_test)
print(y_pred2)
print(y_test)
```

Calculating Errors

```
#Mean Squared Error
from sklearn.metrics import mean_squared_error
MSE2=mean_squared_error(y_test,y_pred2)
print(MSE2)

#Root Mean Squared Error
from math import sqrt
RMSE2 = sqrt(MSE2)
print(RMSE2)

#Mean Absolute Error
from sklearn.metrics import mean_absolute_error
MAE2=mean_absolute_error(y_test, y_pred2)
print(MAE2)
```

3. Random Forest Regression

```
# Fitting Random Forest Regression to the dataset
from sklearn.ensemble import RandomForestRegressor
regressor3 = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor3.fit(X_train, y_train)

y_pred3=regressor3.predict(X_test)
print(y_pred3)
print(y_test)
```

Calculating Errors

```
#Mean Squared Error
from sklearn.metrics import mean_squared_error
MSE3=mean_squared_error(y_test,y_pred3)
print(MSE3)

#Root Mean Squared Error
from math import sqrt
RMSE3 = sqrt(MSE3)
print(RMSE3)
```

```
#Mean Absolute Error
from sklearn.metrics import mean_absolute_error
MAE3=mean_absolute_error(y_test, y_pred3)
print(MAE3)
```

4. XGBoost Regression

```
import xgboost
from xgboost import XGBRegressor
regressor4=XGBRegressor()
regressor4.fit(X_train,y_train)

y_pred4=regressor4.predict(X_test)
print(y_pred4)
print(y_test)
```

Calculating Errors

```
#Mean Squared Error
from sklearn.metrics import mean_squared_error
MSE4=mean_squared_error(y_test,y_pred4)
print(MSE4)

#Root Mean Squared Error
from math import sqrt
RMSE4 = sqrt(MSE4)
print(RMSE4)

#Mean Absolute Error
from sklearn.metrics import mean_absolute_error
MAE4=mean_absolute_error(y_test, y_pred4)
print(MAE4)



import joblib
joblib.dump(regressor1,open('ccc.pkl','wb'))
```

```
model=joblib.load(open('ccc.pkl','rb'))
```

Flask App:

```
import numpy as np
from flask import Flask, request, jsonify, render_template,redirect,url_for
import pandas as pd
from flask_material import Material
import joblib
app = Flask(_name_)
Material(app)
model = joblib.load(open('ccc.pkl', 'rb'))
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/preview')
def preview():
    df = pd.read_csv("F:/Projects/stock market prediction/sm.csv")
    return render_template("preview.html", df_view=df)
@app.route('/Submit',methods=['POST'])
def Submit():
    open1=request.form['open1']
    high=request.form['high']
    low=request.form['low']
    close=request.form['close']
    res_val=model.predict([[np.float64(open1),np.float64(high),np.float64(low)]])[0]
    return render_template('index1.html',prediction_text=' {}'.format(res_val), x=' {}'.format(close))
@app.route('/index.html')
def Goback():
    return render_template('index.html')
if__name__== "_main_":
            app.run(debug=True)
```

Index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Stock Market</title>
</head>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">
<link
href="https://fonts.googleapis.com/css2?family=Merienda+One&family=Sacramento&display=swap"
rel="stylesheet">
<link
href="https://fonts.googleapis.com/css2?family=Fondamento:ital@1&family=Merienda+One&family
=Sacramento&display=swap" rel="stylesheet">
<body>
  <h1style="padding-top:70px;padding-bottom:70px;text-align:center;font-
size:85px;color:#FEFEFA;font-family: 'Merienda One', cursive;">Stock Market Price Prediction</h1>
  <table border='0' width='480px' cellpadding='0' cellspacing='0' align='center'>
    <form action="{{ url_for('Submit')}}" method="POST">
      <tr>
        <td class="d" align='center'>Open : </td>
        <td><input type='text' name='open1' required autofocus></td>
      </tr>
      <tr>
        <td class="d" align='center'>High : </td>
        <td><input type='text' name='high' required></td>
      </tr>
      <tr>
        <td class="d" align='center'>Low : </td>
        <td><input type='text' name='low' required></td>
      </tr>
      <tr>
        <td class="d" align='center'>Close : </td>
        <td><input type='text' name='close' required></td>
      </tr>
      <tr>
        <td>
          <div class="vertical-center"><a style="text-decoration:none" href="/index1.html"><input type='submit' class="block" id='Submit' value="Submit"></a></div>
        </td>
      </tr>


    </form>
    <tr>
      <div class="vertical-center1"><a style="text-decoration:none" href="{{url_for('preview')}}"><button class="block1">View Dataset</button></a></div>
    </tr>
  </table>
  <style>
    body {
      background-image: url('static/images/image1.jpg');
      background-position: center;
      background-repeat: no-repeat;
      background-size: 100% 140vh;
      background-attachement: fixed;
```

```
}
.d {
  color: #FEFEFA;
  font-weight: bold;
  font-family: 'Fondamento', cursive;
  font-size: 25px;
}
.divya {
  align: center;
  width: 300px;
  border: 5px solid blue;
  padding: 50px;
  margin: 100px;
  margin-left: 450px;
}
.vertical-center {
  margin-top: 60px;
  position: absolute;
  left: 50%;
  -ms-transform: translateX(-50%);
  transform: translateX(-50%);
}
.vertical-center1 {
  margin-top: 140px;
  position: absolute;
  left: 50%;
  -ms-transform: translateX(-50%);
  transform: translateX(-50%);
}
```

```
.block {
  display: block;
  width: 100%;
  border: none;
  background-color: purple;
  color: white;
  font-size: 18px;
  padding: 10px 18px;
  cursor: pointer;
  text-align: center;
  border-radius: 6px;
  margin-top: 30px;
  text-decoration: none;
}

.block1 {
  display: block;
  width: 100%;
  border: none;
  background-color: green;
  color: white;
  font-size: 15px;
  padding: 7px 16px;
  cursor: pointer;
  text-align: center;
  border-radius: 6px;
  margin-top: 30px;
  text-decoration: none;
}
```

```
    .block:hover {
      background-color: #fff;
      color: purple;
      border: 1px solid grey;

    }

    .block1:hover {
      background-color: #fff;
      color: green;
      border: 1px solid grey;

    }
  </style>
</body>
</html>
```

Index1.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Prediction Analysis</title>
</head>
<body>
  <h1style="padding-top:20px;padding-bottom:25px;text-align:center;font-
size:70px;color:white;">Stock Market Price Prediction</h1>
  <style>
```

```
.divya {
  align: center;
  width: 300px;
  border: 3px solid white;
  padding: 50px;
  margin: 40px;
  margin-left: 450px;
}

body {
  background-image: url('static/images/image4.jpg');
  background-position: center;
  background-repeat: no-repeat;
  background-size: 100% 200vh;
  background-attachement: fixed;
}
h3 {
  color: white;
}
.vertical-center {
  margin: 6px;
  position: absolute;
  left: 48%;
  -ms-transform: translateX(-50%);
  transform: translateX(-50%);
}
.block {
  display: block;
  width: 100%;
```

```
      border: none;
      background-color: purple;
      color: white;
      font-size: 18px;
      padding: 10px 18px;
      cursor: pointer;
      text-align: center;
      border-radius: 6px;
      margin-top: 8px;
      text-decoration: none;
    }
    .block:hover {
      background-color: #fff;
      color: purple;
      border: 1px solid grey;
    }
  </style>
  <div class="divya">
    <h3 align='center'>Predicted Value</h3>
    <h3 align='center'>{{prediction_text}}</h3>
    <h3 align='center'>Actual Value</h3>
    <h3 align='center'>{{x}}</h3>
  </div>
  <table border='0'>
    <tr>
      <td>
        <div class="vertical-center"><a style="text-decoration:none" href="/index.html"><input
type='submit' class="block" id='Goback' value="Goback"></a></div>
      </td>
    </tr>
  </table>
</body>
</html>
```

Preview.html

```
{% extends "material/base.html" %}
{% block content %}
<html>
<head>
  <title>Dataset</title>
</head>
<body>
  <div class="showcase container black">
    <div class="row">
      <div class="col 12 m10 offset-ml center white-text">
        <h3>Stock Market Price Prediction</h3>
        <a href="{{url_for('home')}}" class="btn btn-medium primary">Back</a>
      </div>
    </div>
  </div>
  <div class="con">
    {{ df_view.to_html(classes="table striped" ,na_rep="-") | safe}}
  </div>
  <style>
    .con {
      margin-left: 70px;
    }
  </style>
</body>
```

```
</html>
{% endblock %}
{% block scripts %}
{{ super() }}
{% endblock %}
```

7.3 Input/Output Screens

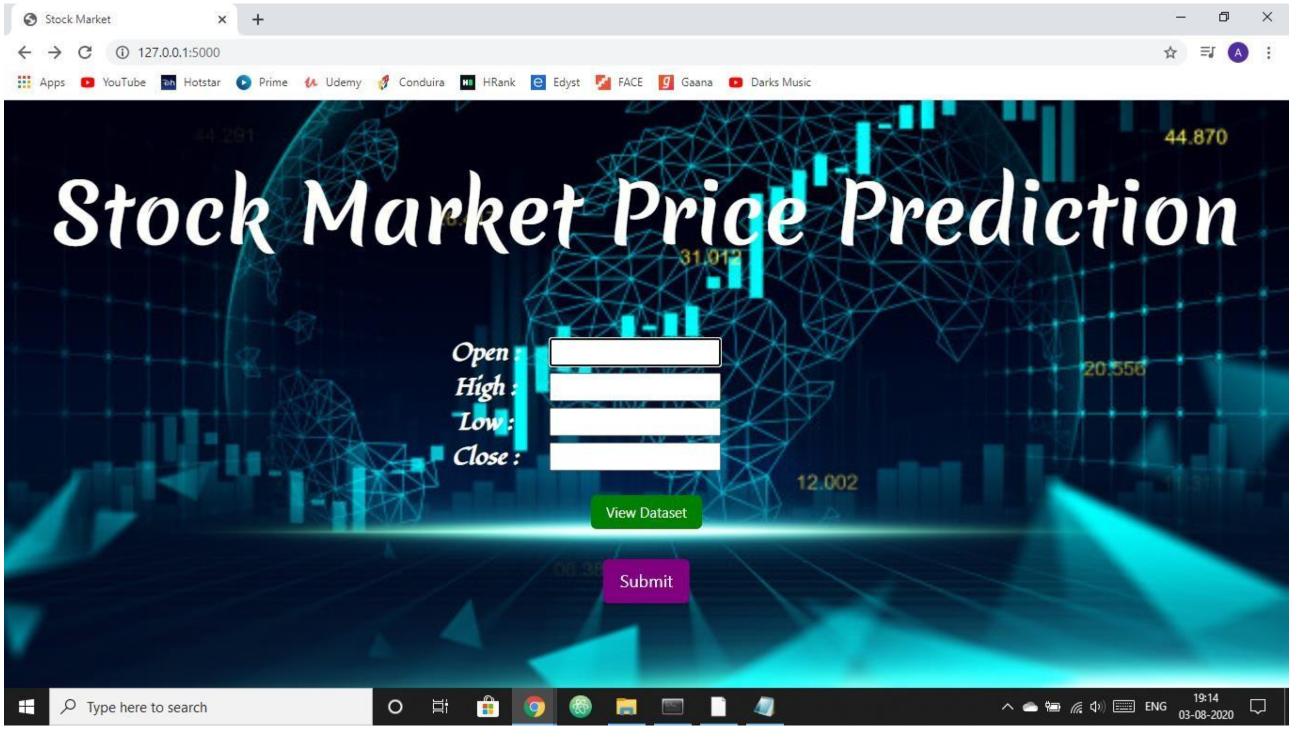
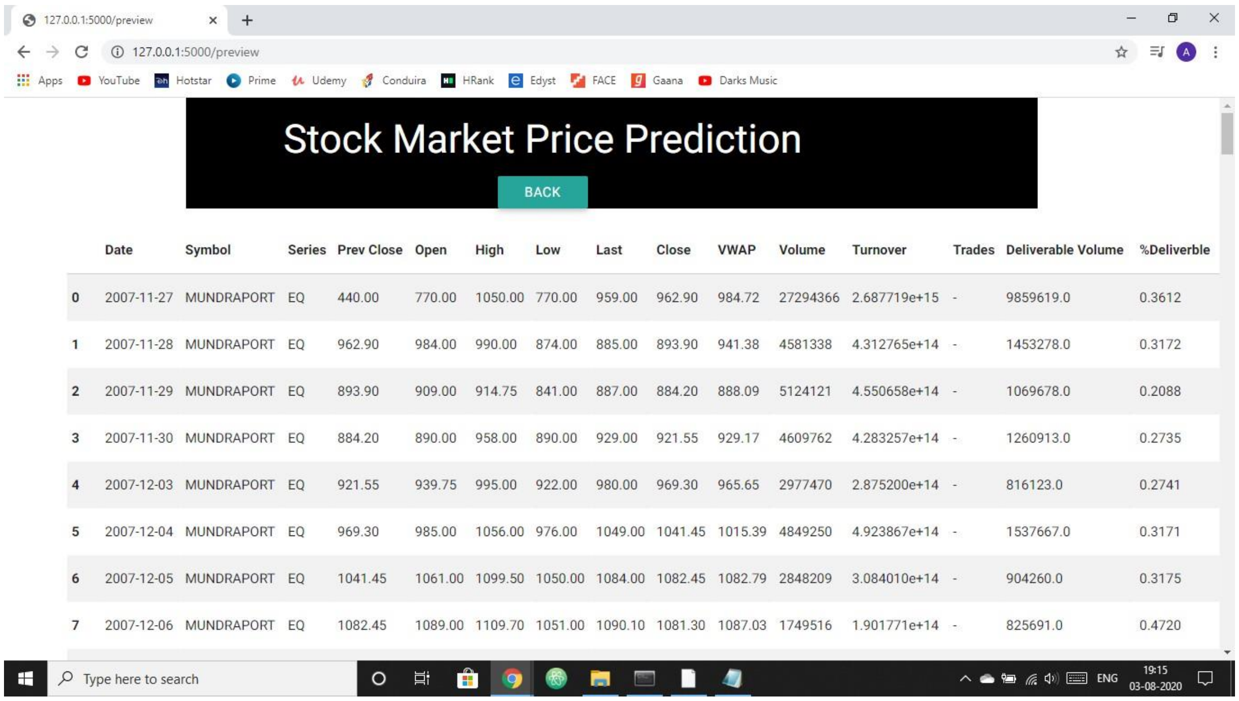
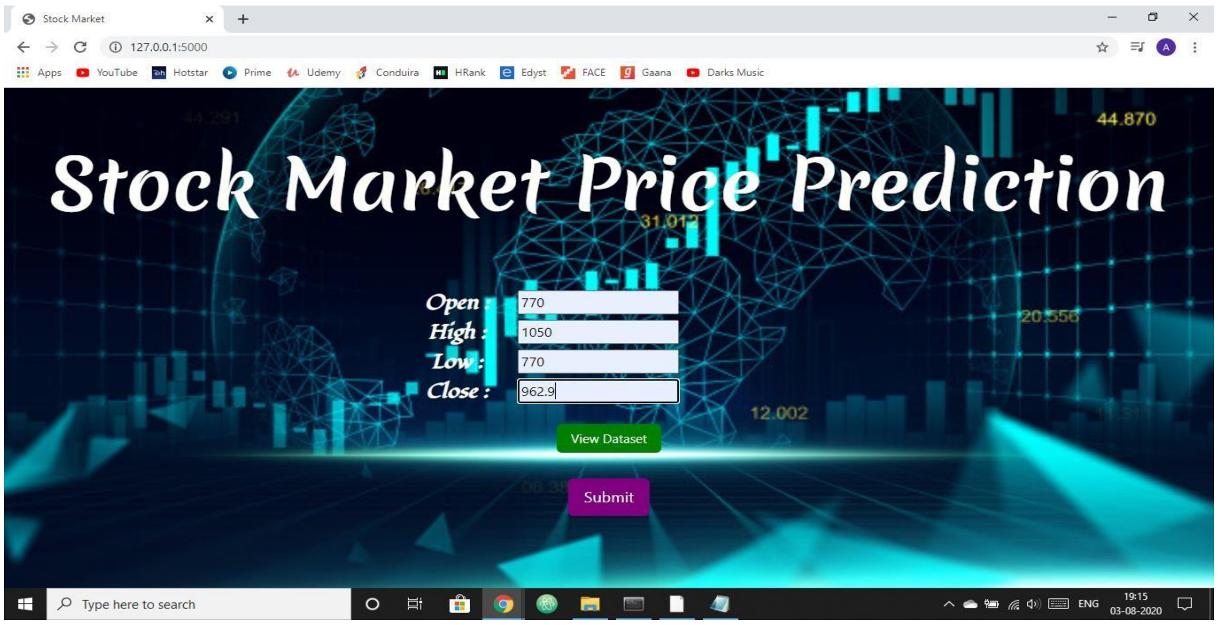


Fig:Input Screen

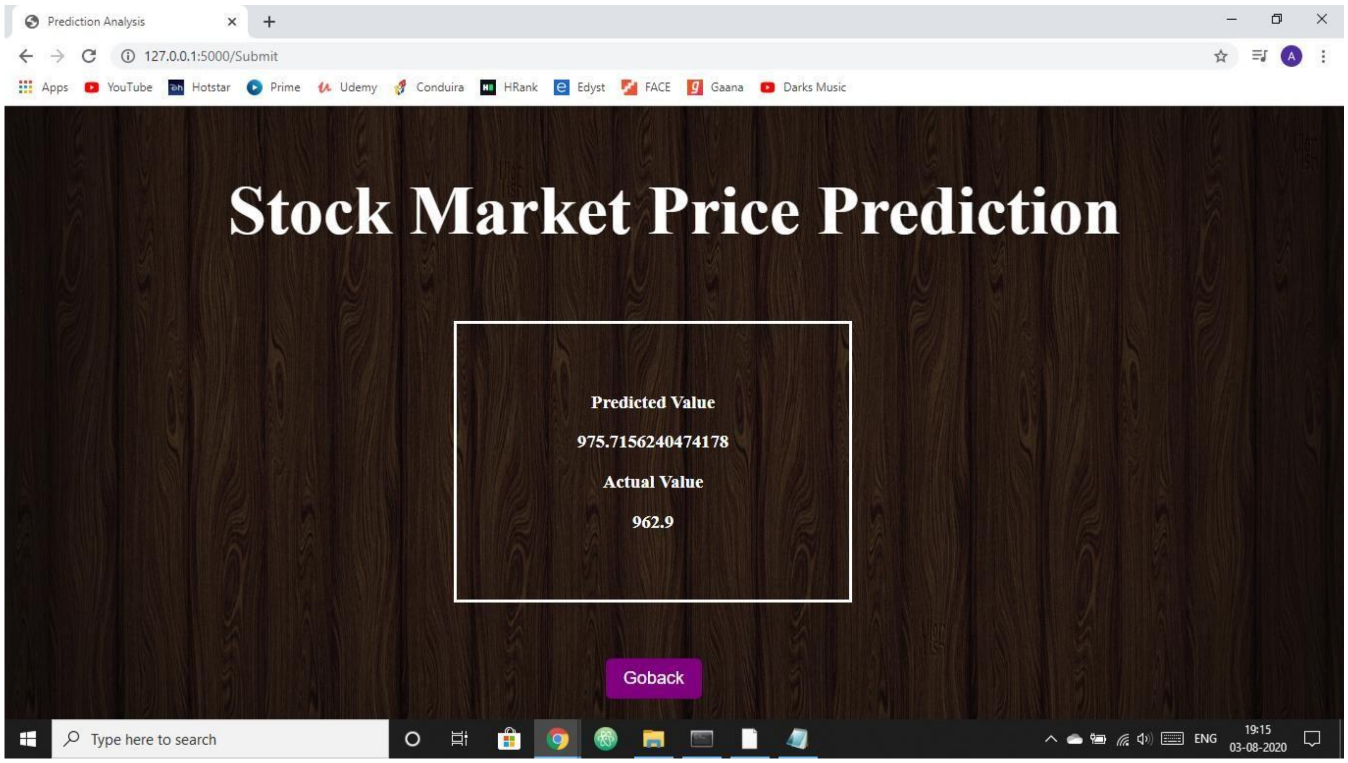


	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverble
0	2007-11-27	MUNDRAPORT	EQ	440.00	770.00	1050.00	770.00	959.00	962.90	984.72	27294366	2.687719e+15	-	9859619.0	0.3612
1	2007-11-28	MUNDRAPORT	EQ	962.90	984.00	990.00	874.00	885.00	893.90	941.38	4581338	4.312765e+14	-	1453278.0	0.3172
2	2007-11-29	MUNDRAPORT	EQ	893.90	909.00	914.75	841.00	887.00	884.20	888.09	5124121	4.550658e+14	-	1069678.0	0.2088
3	2007-11-30	MUNDRAPORT	EQ	884.20	890.00	958.00	890.00	929.00	921.55	929.17	4609762	4.283257e+14	-	1260913.0	0.2735
4	2007-12-03	MUNDRAPORT	EQ	921.55	939.75	995.00	922.00	980.00	969.30	965.65	2977470	2.875200e+14	-	816123.0	0.2741
5	2007-12-04	MUNDRAPORT	EQ	969.30	985.00	1056.00	976.00	1049.00	1041.45	1015.39	4849250	4.923867e+14	-	1537667.0	0.3171
6	2007-12-05	MUNDRAPORT	EQ	1041.45	1061.00	1099.50	1050.00	1084.00	1082.45	1082.79	2848209	3.084010e+14	-	904260.0	0.3175
7	2007-12-06	MUNDRAPORT	EQ	1082.45	1089.00	1109.70	1051.00	1090.10	1081.30	1087.03	1749516	1.901771e+14	-	825691.0	0.4720

Fig:Dataset



Open : 770
High : 1050
Low : 770
Close : 962.9



Stock Market Price Prediction

Predicted Value

975.7156240474178

Actual Value

962.9

Goback

Fig:Output Screen

8.Testing

8.1 Introduction

The development of software includes series of productive activities and testing is an important activity of them. This phase is a critical element of software quality assurance and represents the ultimate review of specification, coding and testing.

The main objectives of testing are as follows:

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an undiscovered error.
- A successful test is one uncovers an undiscovered error.

Testing can be done in different ways. Some of the types of testing are mentioned below. The main purpose of any type of test is to systematically uncover different classes of errors and do so with a minimum amount of time and effort.

Types of testing:

- Unit testing
- Integration testing
- Regression testing
- System testing
- Performance Testing
- Alpha testing
- Beta testing

Testing can be done manually or by testing tools. There are several testing tools fordifferent software.

Unit Testing: It is a method by which individual units of source code, sets of one or more computer modules together with associated control data, usage procedures and operating procedures, are tested to determine if they are fit for use. Unit testing focuses verification effort on the smallest unit of software design-the software component or module. The unit test is white-box oriented. The unit testing is implemented in every module of student attendance management System. By giving correct manual input to the system, the data's are stored in database and retrieved. If we want required module to access input or get the output, it should be checked for the accessibility and result is to be displayed else error message should be displayed.

Integration Testing: It is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Regression Testing: Regression testing is any type of software testing that seeks to uncover new software bugs, or regressions, in existing functional and non-functional areas of a system after changes such as enhancements, patches or configuration changes, have been made to them.

System Testing: System testing of software or hardware is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system.

Performance Testing: Performance testing is designed to test the run-time performance of software within the context of an integrated system. Performance testing occurs throughout all steps in the testing process. Even at the unit level, the performance of an individual module may be assessed as white-box tests are conducted. This project reduces manual work and helps in getting students attendance status in much faster. No extra time is required waiting for the results, as soon as the correct data is entered the resulted will be displayed in a few milliseconds.

Alpha Testing: Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developer’s site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

Beta testing: Beta testing comes after alpha testing and can be considered as a form of external user acceptance testing. Versions of the software, known as beta versions, are released to a limited outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

Each module can be tested using the following two strategies:

1. Internal program logic is exercised using ―White box ‖ test case design techniques.
2. Software requirements are exercised using ―Black box ‖ test case design techniques.

In both cases, the intent is to find the maximum number of errors with the Minimum amount of effort and time.

Black box Testing: In this strategy some test cases are generated as input conditions that execute all functional requirements for the program. This testing is used to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access
- Performance errors
- Initialization and termination errors

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

White box Testing: In this testing, test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It is a method of testing software tests internal structures or workings of an application as opposed to its functionality (i.e. Black box testing). In white box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths throw the code and determine appropriate outputs. This is analogous to testing nodes in a circuit. White box Testing can be applied at the unit, integration, system levels of the software testing process. Although traditional testers tended to think of White box testing as being one at the unit level.

8.2 Test code*:*

```
print(model.predict([[770,1050,770]]))
Predicted Value: [975.71562405]
Actual Value: [962.9]
```

8.3 Test Cases

Test case is an object for execution for other modules in the architecture does not represent any

interaction by itself. A test case is a set of sequential steps to execute a test operating on a set of predefined inputs to produce certain expected outputs.
There are two types of test cases: -manual and automated.

A manual test case is executed manually while an automated test case is executed using automation. In system testing, test data should cover the possible values of each parameter based on the requirements. Since testing every value is impractical, a few values should be chosen from each equivalence class. An equivalence class is a set of values that should all be treated the same. Ideally, test cases that check error conditions are written separately from the functional test cases and should have steps to verify the error messages and logs. Realistically, if functional test cases are not yet written, it is ok for testers to check for error conditions when performing normal functional test cases. It should be clear which test data, if any is expected to trigger errors.

9.Conclusion

We here by conclude that Stock Market Price Prediction is implemented in 3 modules, first module is inputting the dataset and splitting dataset and second one is choosing one of the regression algorithm and apply it to train, test and calculate Mean Square Error(MSE),Mean Absolute Error(MAE) and Root Mean Square Error(RMSE) of the model whereas third module is of deploying the model with the help of flask framework as a web application.To provide best model with less error we built our model using Multiple Linear Regression algorithm.

9.BIBLIOGRAPHY

References:

- “Machine Learning For Absolute Beginners: A Plain English Introduction”, 2nd edition, Scatterplot Press.
- Machine Learning A-Z™: Hands-On Python In Data Science on Udemy.s

Web References:

- Dataset: https://www.kaggle.com/rohanrao/nifty50-stock-market-data
- https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/
- https://en.wikipedia.org/wiki/Stock_market_prediction#Machine_learning