

# **TESLA STOCK PRICE PREDICTION USING REGRESSION MODEL**

**A PROJECT REPORT**

*Submitted by*

**ARUNA.K** [REGISTER NO:211419104020]

**ASTALAKSHMI.G** [REGISTER NO: 211419104028]

**PREETHI.B** [REGISTER NO: 211419104200]

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**APRIL 2023**

**PANIMALAR ENGINEERING COLLEGE**  
(An Autonomous Institution, Affiliated to Anna University, Chennai)

**BONAFIDE CERTIFICATE**

Certified that this project report **“TESLA STOCK PRICE PREDICTION USING REGRESSION MODEL”** is the bonafide work of **“ARUNA.K(211419104020), ASTALAKSHMI.G(211419104028) , PREETHI.B(211419104200)”** who carried out the project work under my supervision.

**SIGNATURE**

**Dr.L.JABASHEELA,M.E.,Ph.D.,  
HEAD OF THE DEPARTMENT**

DEPARTMENT OF CSE,  
PANIMALAR ENGINEERING COLLEGE,  
NASARATHPETTAI,  
POONAMALLEE,  
CHENNAI-600 123.

**SIGNATURE**

**Dr.K.SANGEETHA M.E, Ph.D  
SUPERVISOR  
ASSOCIATE PROFESSOR**

DEPARTMENT OF CSE,  
PANIMALAR ENGINEERING COLLEGE,  
NASARATHPETTAI,  
POONAMALLEE,  
CHENNAI-600 123.

Certified that the above candidate(s) was examined in the End semester  
Project Viva-Voce Examination held on.....

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **DECLARATION**

We..... **ARUNA.K[211419104020]**, **ASTALAKSHMI.G[211419104028]**, **PREETHI.B [211419104200]** ..... hereby declare that this project report titled **“TESLA STOCK PRICE PREDICTION USING REGRESSION MODEL”**, under the guidance of **Dr.K.SANGEETHA M.E ,PhD .**,is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

**ARUNA.K**

**ASTALAKSHMI.G**

**PREETHI.B**

## **ACKNOWLEDGEMENT**

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHI KUMAR,M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L.JABASHEELA , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank our **Dr.K.SANGEETHA M.E ,PhD.,**and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

**ARUNA.K**  
**ASTALAKSHMI.G**  
**PREETHI.B**

## **ABSTRACT**

Stock prices are initially set by an initial public offering (IPO), which occurs when a company first issues its shares on the market. Investment companies use a variety of metrics, including the total number of shares being offered, to calculate the price of a stock. The share price will then fluctuate as a result of the factors mentioned, with potential business earnings having a substantial impact. Using financial metrics including a company's earnings history, market movements, and the profit it is predicted to make, traders regularly evaluate a company's value. As a result, stock price forecasting has grown in significance as a field of study. Forecasting machine learning-based stock price prediction techniques is the aim. Univariate, bivariate, and multivariate analysis are used to analyse the dataset using SMLT's supervised machine learning technique (SMLT). to offer an approach based on machine learning for accurately forecasting stock price. The proposed machine learning algorithm technique can be compared to the best accuracy in terms of precision, recall, and F1 Score. The best model, or the one with the highest accuracy, is then chosen after a comparison of the four, and it is implemented into a webpage. The algorithms used are Adaboost algorithm, Decision tree algorithm, Lasso regression and Ridge regression. The four models are then compared and the best one ie. the one with highest accuracy is deployed into a webpage. The highest accuracy obtained from the above algorithms is 99.83%. This work provides a webpage for Tesla stock price prediction with accurate results.

## **LIST OF FIGURES**

<b>FIGURE NO</b>	<b>NAME OF THE FIGURE</b>	<b>PAGE NO</b>
<b>3.3</b>	CPM Diagram	17
<b>4.1</b>	ER Diagram	20
<b>4.2</b>	Data flow diagram	21
<b>4.3.1</b>	Use case Diagram	22
<b>4.3.2</b>	Class Diagram	23
<b>4.3.3</b>	Activity Diagram	24
<b>4.3.4</b>	Sequence Diagram	25
<b>5.1</b>	System Architecture	27
<b>5.2</b>	Workflow diagram	28
<b>5.3.1</b>	Data Preprocessing	31
<b>5.3.2</b>	Data Visualization	32
<b>5.3.3.1</b>	Ada Boost Algorithm	34
<b>5.3.3.2</b>	Decision Tree Algorithm	36
<b>5.3.3.3</b>	Ridge Algorithm	37
<b>5.3.3.4</b>	Lasso Algorithm	38
<b>5.3.4</b>	Deployment	39
<b>7.1</b>	Accuracy of algorithm	54
<b>8.1</b>	Results and discussion	57

## LIST OF SYMBOLS, ABBREVIATIONS

<b>PCC</b>	Pearson correlation coefficient
<b>BLS</b>	Board learning system
<b>SVR</b>	Support Vector Regression
<b>GBDT</b>	Gradient Boosting Decision Tree
<b>CNN</b>	Convolutional Neural Network
<b>LSTM</b>	Long Short -Term Memory
<b>SVR</b>	Support Vector Regression
<b>RF</b>	Random Forest
<b>MLP</b>	Multi Layer Perceptron
<b>GRU</b>	Gated Recurrent unit
<b>LOB</b>	Limit Order Book
<b>HFS</b>	Hybrid Feature Selection
<b>RSI</b>	Relative Strength Index
<b>DNN</b>	Deep Neural Network
<b>MSE</b>	Mean squared Error
<b>MAE</b>	Mean Absolute Error
<b>RMSE</b>	Root Mean Square Error
<b>SVC</b>	Support Vector Classifier
<b>ANN</b>	Artificial Neural Network
<b>KNN</b>	K-Nearest Neighbors
<b>RNN</b>	Recurrent Neural Network

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	v
	<b>LIST OF FIGURES</b>	vi
	<b>LIST OF SYMBOLS, ABBREVIATIONS</b>	vii
<b>1.</b>	<b>INTRODUCTION</b>	3
	1.1 Problem Definition	4
<b>2.</b>	<b>LITERATURE SURVEY</b>	7
<b>3.</b>	<b>SYSTEM ANALYSIS</b>	16
	3.1 Existing System	17
	3.2 Proposed system	17
	3.3 Feasibility Study	18
	3.4 Project Requirements	19
<b>4.</b>	<b>SYSTEM DESIGN</b>	22
	4.1 UML Diagrams	23
	4.1.1 Use case Diagram	23
	4.1.2 Class Diagram	24
	4.1.3 Activity Diagram	25
	4.1.4 Sequence diagram	26
	4.2 Dataflow Diagram	27
<b>5.</b>	<b>SYSTEM ARCHITECTURE</b>	28
	5.1 System Architecture	29



5.2	Workflow Diagram	31
<b>6.</b>	<b>SYSTEM IMPLEMENTATION</b>	32
6.1	Module Description	33
6.1.1	Data preprocessing	33
6.1.2	Data visualization	34
6.1.3	ML model development	35
6.1.3.1	Adaboost Algorithm	36
6.1.3.2	Decision tree Algorithm	37
6.1.3.3	Ridge Algorithm	38
6.1.3.4	Lasso Algorithm	39
6.1.4	Deployment	39
<b>7.</b>	<b>PERFORMANCE EVALUATION</b>	40
7.1	Results & Discussion	41
7.2	Comparative analysis	42
<b>8.</b>	<b>CONCLUSION</b>	45
8.1	Conclusion and Future Enhancements	46
	<b>APPENDIX</b>	47
	APPENDIX 1-Sample Dataset	47
	APPENDIX 2-Sample Coding	49
	APPENDIX 3-Sample Screens	61
	<b>REFERENCES</b>	69

# **CHAPTER 1**

## **INTRODUCTION**

# **CHAPTER-1**

## **INTRODUCTION**

### **1.1. PROBLEM DEFINITION**

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labelling to learn data has to be labelled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where we have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is  $y = f(X)$ . The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for regression and categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as "red" or "blue".

Regression is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables. It is mainly used for prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables.

In Regression, we plot a graph between the variables which best fits the given datapoints, using this plot, the machine learning model can make predictions about the data. In simple words, "Regression shows a line or curve that passes through all the datapoints on target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum." The distance between datapoints and line tells whether a model has captured a strong relationship or not.

Regression analysis helps in the prediction of a continuous variable. There are various scenarios in the real world where we need some future predictions such as weather condition, sales prediction, marketing trends, etc., for such case we need some technology which can make predictions more accurately. So for such case we need Regression analysis which is a statistical method and used in machine learning and data science.

One of the more difficult responsibilities is to foresee and scrutinise the stock market. Numerous factors, including market volatility and a variety of autonomous and affect an individual, alter the market value of a stock. Due of these variables, no stock market expert can anticipate the market's rise and collapse with certainty. However, breakthroughs in Stock Market Prediction have begun to utilise Machine Learning and its strong algorithms to analyse stock market data in recent years. Several organisations, it is fair to assume, use algorithms that employ machine learning to make stock market predictions. This article will guide you through some kind of simple Python application of assessing and anticipating a popular worldwide company's stock prices utilizing various algorithms that employ machine learning. The proposed machine learning algorithm technique can be compared to the best accuracy in terms of precision, recall, and F1 Score. The best model, or the one with the highest accuracy, is then chosen after a comparison of the four, and it is implemented into a webpage. The algorithms used are Adaboost algorithm, Decision tree algorithm, Lasso regression and Ridge regression. The four models are then compared and the best one ie. the one with highest accuracy is deployed into a webpage. The highest accuracy obtained from the above algorithms is 99.83%. This work provides a webpage for Tesla stock price prediction with accurate results.

# **CHAPTER 2**

# **LITERATURE**

# **SURVEY**

## **CHAPTER-2**

### **LITERATURE SURVEY**

#### **[1] Pearson Correlation Coefficient-Based Performance Enhancement of Broad Learning System for Stock Price Prediction**

**Author:** Guanzhi Li, Aining Zhang, Qizhi Zhang, Di Wu , and Choujun Zhan

**Year** : 2022

Accurate prediction of a stock price is a challenging task due to the complexity, chaos, and non-linearity nature of financial systems. In this brief, we proposed a multi-indicator feature selection method for stock price prediction based on Pearson correlation coefficient (PCC) and Broad learning system (BLS), named the PCC-BLS framework. Firstly, PCC was used to select the input features from 35 features, including original stock price, technical indicators, and financial indicators. Secondly, these screened input features were used for rapid information feature extraction and training a BLS. Four stocks recorded on the Shanghai Stock Exchange or Shenzhen Stock Exchange were adopted to evaluate the performance of the proposed method.

**Methodologies:** Broad Learning System, Pearson Correlation Coefficient

**Merit:** Purpose of this brief is to help investors make reasonable trading decisions by accurately predicting stock price movements using machine learning methods.

**Demerit :** Deployment is not implemented. It is very complex system as Shanghai Stock Exchange price is predicted (Chinese Yuan (CNY)).

#### **[2] Predicting Stock Price Changes Based on the Limit Order Book: A Survey**

**Author :** Ilia Zaznov      Julian Kunkel

**Year** : 2022

This survey starts with a general overview of the strategies for stock price change predictions based on market data and in particular Limit Order Book (LOB) data. The main discussion is devoted to the systematic analysis, comparison, and critical evaluation of the state-of-the-art studies in the research area of stock price movement predictions based on LOB data. LOB and Order Flow data are two of the most valuable information sources available to traders on the stock markets. Academic researchers are actively exploring the application of different quantitative methods and algorithms for this type of data to predict stock price movements.

With the advancements in machine learning and subsequently in deep learning, the complexity and computational intensity of these models was growing, as well as the claimed predictive power. Some researchers claim accuracy of stock price movement prediction well in excess of 80%. These models are now commonly employed by automated market-making programs to set bids and ask quotes. If these results were also applicable to arbitrage trading strategies, then those algorithms could make a fortune for their developers. Thus, the open question is whether these results could be used to generate buy and sell signals that could be exploited with active trading. Therefore, this survey paper is intended to answer this question by reviewing these results and scrutinising their reliability. The ultimate conclusion from this analysis is that although considerable progress was achieved in this direction, even the state-of-art models can not guarantee a consistent profit in active trading. Taking this into account several suggestions for future research in this area were formulated along the three dimensions: input data, model's architecture, and experimental setup. In particular, from the input data perspective, it is critical that the dataset is properly processed, up-to-date, and its size is sufficient for the particular model training. From the model architecture perspective, even though deep learning models are demonstrating a stronger performance than classical models, they are also more prone to over-fitting. To avoid over-fitting it is suggested to optimize the feature space, as well as a number of layers and neurons, and apply dropout functionality. The over-fitting problem can be also addressed by optimising the experimental setup in several ways: Introducing the early stopping mechanism; Saving the best weights of the model achieved during the training; Testing the model on the out-of-sample data, which should be separated from the validation and training samples. Finally, it is suggested to always conduct the trading simulation under realistic market conditions considering transactions costs, bid–ask spreads, and market impact

**Methodologies** : Long short-term memory(LSTM)

**Merit** : It is focused on the critical evaluation of the current studies in the subject area of stock price movement predictions based on LOB data.

**Demerit** : It is focused on the critical evaluation of the current studies in the subject area of stock price movement predictions based on LOB data.

### **[3] Stock Movement Prediction using Technical and Data**

**Author** : Dylan M. Crain

**Year** : 2021



Stock price prediction is a notoriously challenging problem. Typically, when trying to solve it, researchers and individuals use either technical (prices and volumes) or fundamental (text-based) data. However, it is exceedingly rare for both forms to be used. This work utilizes Tesla data with sentiment analysis performed on news titles pertaining to the company as well as technical data on its stock over a three year period in order to predict closing price movement. For a next day prediction, the final model architecture (a blended model) results in an accuracy of 65%, which is just under the highest accuracy observed in the literature review. Also, as time to predict goes from 1 day to 3, the GRU model does not have a large drop in accuracy, which insinuates it could be used for later predictions.

**Methodologies :** Gated recurrent unit (GRU) and Long short-term memory (LSTM)

**Merit :** Both technical and fundamental data is used.

**Demerit :** It would be a useful alteration to include more than just one stock in the analysis.

#### **[4] Novel Stock Crisis Prediction Technique—A Study on Indian Stock Market**

**Author :** Nagaraj Naik, Biju R. Mohan

**Year :** 2021

A stock market crash is a drop in stock prices more than 10% across the major indices. Stock crisis prediction is a difficult task due to more volatility in the stock market. Stock price sell-offs are due to various reasons such as company earnings, geopolitical tension, financial crisis, and pandemic situations. Crisis prediction is a challenging task for researchers and investors. We proposed a stock crisis prediction model based on the Hybrid Feature Selection (HFS) technique. First, we proposed the HFS algorithm to remove the irrelevant financial parameters/features of stock. The second is the Naive Bayes method is considered to classify the strong fundamental stock. The third is we have used the Relative Strength Index (RSI) method to find a bubble in stock price. The fourth is we have used moving average statistics to identify the crisis point in stock prices. The fifth is stock crisis prediction based on Extreme Gradient Boosting (XGBoost) and Deep Neural Network (DNN) regression method. The performance of the model is evaluated based on Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE). HFS based XGBoost method was performed better than HFS based DNN method for predicting the stock crisis. The experiments considered the Indian datasets to carry out the task. In the future, the researchers can

explore other technical indicators to predict the crisis point. There is more scope to improve and fine-tune the XGBoost method with a different optimizer.

**Methodologies :** Log-Periodic Power Law(LPPL)

**Merit :** Proposed the Hybrid Feature Selection (HFS) algorithm to remove irrelevant stock financial parameters features.

**Demerit :** Explored only a limited number of technical parameters of stock prices

## **[5] Stock Market Prediction Using RNN LSTM**

**Author :** Priyanka Srivastava; P K Mishra

**Year : 2021**

Bond price prediction is a trendy, demanding, hard and complicated problem in the realm of computation that usually includes considerable interaction between people and computers. The trends for predicting stock market physical aspects versus physiological, rational, and illogical conduct, investor emotion, market whispers are engaged in various factors. All those facets mix to make stock values extremely sophisticated and exceedingly difficult to accurately anticipate. Because of the linked nature of stock prices, sequential prediction algorithms may be used for stock market prediction efficiently. ML methods can identify patterns and determine the logic and forecasts and can be utilized to produce unerringly correct predictions. We have explored several different algorithms to forecast the stock market from simple algorithms, such as Simple Average, Linear regression, to advance algorithms like ARIMA, LSTM, and compare what gives us a more accurate result and works more efficiently. We offer a research technology that employs the improved Long Short Term Memory (LSTM) version of RNN, with stochastic gradient descent maintaining the weights for each data variable. To help us deliver more efficient and accurate outcomes than existing stock price prediction systems. We have utilized the TSLA dataset to create the stock prediction model: this is TESLA Inc. from Yahoo Finance. We have analyzed future stock prices using data-frame closing prices, built up and trained the LSTM model, and have taken a data set sample to generate stock forecasts and computed additional RMSE for correctness and effectiveness. We have also displayed several algorithms for comparative predictions, Based on these outcomes, LSTM is recommended for stock market forecasts.

**Methodologies :** Long Short Term Memory (LSTM), Recurrent neural network (RNN)

**Merit :** Due to its capability of storing past information, LSTM is very useful in predicting stock prices. This is because the prediction of a future stock price is dependent on

the previous prices.

**Demerit** : Statistical relevance of the result is needed. 5. More careful data design is needed and systematically analyzed.

## **[6] Predicting Stock Market Trends Using Machine Learning and DeepLearning Algorithms Via Continuous and Binary Data; a Comparative Analysis**

**Author** :Mojtaba Nabipour ,Pooyan Nayyeri, Hamed Jabani ,Shahab S, AmirMosavi

**Year** : 2019

The nature of stock market movement has always been ambiguous for investors because of various influential factors. This study aims to significantly reduce the risk of trend prediction with machine learning and deep learning algorithms. Four stock market groups, namely diversified financials, petroleum, non- metallic minerals and basic metals from Tehran stock exchange, are chosen for experimental evaluations. This study compares nine machine learning models (Decision Tree, Random Forest, Adaptive Boosting (Adaboost), eXtreme Gradient Boosting (XGBoost), Support Vector Classifier (SVC), Naïve Bayes, K-Nearest Neighbors (KNN), Logistic Regression and Artificial Neural Network (ANN)) and two powerful deep learning methods (Recurrent Neural Network (RNN) and Long short-term memory (LSTM)). Ten technical indicators from ten years of historical data are our input values, and two ways are supposed for employing them. Firstly, calculating the indicators by stock trading values as continuous data, and secondly converting indicators to binary data before using. Each prediction model is evaluated by three metrics based on the input ways. The evaluation results indicate that for the continuous data, RNN and LSTM outperform other prediction models with a considerable difference. Also, results show that in the binary data evaluation, those deep learning methods are the best; however, the difference becomes less because of the noticeable improvement of models' performance in the second way.

**Methodologies** : Nine machine learning methods (Decision Tree, Random Forest, Ada boost, XG Boost, SVC, Naïve Bayes, KNN, Logistic Regression and ANN) and two deep learning algorithms (RNN and LSTM) are used.

**Merit** : The evaluation results indicate that for the continuous data, RNN and LSTM outperform other prediction models with a considerable difference

**Demerit** : Complexity and nonlinearity are two main challenges faced.

## **[7] Stock Price Prediction: A Comparative Study between Traditional Statistical Approach and Machine Learning Approach**

**Author :** Indronil Bhattacharjee, Pryonti Bhattacharja

**Year :** 2019

Stock market is one of the most important sectors of a country's economy. Prediction of stock prices is not easy since it is not stationary in nature. The objective of this paper is to find the best possible method to predict the closing prices of stocks through a comparative study between different traditional statistical approaches and machine learning techniques. Predictions using statistical methods like Simple Moving Average, Weighted Moving Average,

Exponential Smoothing, Naive approach, and machine learning methods like Linear Regression, Lasso, Ridge, K-Nearest Neighbors, Support Vector Machine, Random Forest, Single Layer Perceptron, Multi-layer Perceptron, Long Short Term Memory are performed. Moreover, a comparative study between statistical approaches and machine learning approaches has been done in terms of prediction performances and accuracy. After studying all the methods individually, the machine learning approach, especially the neural network models are found to be the most accurate for stock price prediction.

**Methodologies :** Simple Moving Average, Weighted Moving Average, Exponential Smoothing, Naive approach, and machine learning methods

**Merit :** To predict the closing prices of stocks through comparative study between different traditional statistical approaches and machine learning Techniques.

**Demerit :** Only the the neural network models are found to be the most accurate for stock price prediction.

## **[8] Indian stock market prediction using artificial neural networks on tickdata**

**Author :** Dharmaraja Selvamuthu ,Vineet Kumar & Abhishek Mishra

**Year :** 2019

Nowadays, the most significant challenges in the stock market is to predict the stock prices. The stock price data represents a financial time series data which becomes more difficult to predict due to its characteristics and dynamic nature.

Support Vector Machines (SVM) and Artificial Neural Networks (ANN) are widely used for prediction of stock prices and its movements. Every algorithm has its way of learning patterns and then predicting. Artificial Neural Network (ANN) is a popular method which also incorporate technical analysis for making predictions in financial markets.

Most common techniques used in the forecasting of financial time series are Support Vector Machine (SVM), Support Vector Regression (SVR) and Back Propagation Neural Network (BPNN). In this article, we use neural networks based on three different learning algorithms, i.e., Levenberg-Marquardt, Scaled Conjugate Gradient and Bayesian Regularization for stock market prediction based on tick data as well as 15-min data of an Indian company and their results compared.

**Methodologies** : Support Vector Machines (SVM) and Artificial Neural Networks(ANN)

**Merit** : Easy and simple algorithms were used for prediction

**Demerit** : Recurrent Neural Networks may provide better predictions than the neural networks used in this study.

## **[9] Stock Market Prediction Using Machine Learning**

**Author** :Yogita Deshmukh, Deepmala Saratkar, Harshal Hiratkar, Sudhanshu Dhopte, Swapnil Patankar, Triveni Jambhulkar, Yash Tiwari

**Year:** 2019

The stock market has always been a promising avenue for lucrative investing, but most of the profit making depends on the analysis of the current and past market scenario followed by subsequent predictive actions. The currently overblown market economy has given rise to numerous variables which need to be considered before making a beneficial transaction in the stock market. Manually analysing all these variables and affecting factors is too cumbersome and error prone. Therefore, a Machine Learning approach is best suited for analysis of such a seemingly chaotic system. In this project we are using Machine learning, which give a prediction of various aspects of a particular stock or an index, such as future values of the opening price, closing price, index value etc. This will help investors and traders make better and faster decisions.

**Methodologies** : Multi Layer Perception (MLP) model

**Merit** : A neural network is a model characterized by an activation function, which

is used by interconnected information processing units to transform input into output. The first layer of the neural network of the neural network receives the raw input, process it and passes the processed information to the hidden layer

**Demerit** : NN require very large number of previous data. The best NN architecture topology is still unknown. For complex networks the result and accuracy may decrease

## **[10] Stock Price Prediction Using Machine Learning Techniques**

**Author:** Sumeet Sarode, Harsha G. Tolani, Prateek Kak, C S Lifna

**Year:** 2019

In today's economy, there is a profound impact of the stock market or equity market. Prediction of stock prices is extremely complex, chaotic, and the presence of a dynamic environment makes it a great challenge. Behavioural finance suggests that decision-making process of investors is to a very great extent influenced by the emotions and sentiments in response to a particular news. Thus, to support the decisions of the investors, we have presented an approach combining two distinct fields for analysis of stock exchange. The system combines price prediction based on historical and real-time data along with news analysis. LSTM (Long Short-Term Memory) is used for predicting. It takes the latest trading information and analysis indicators as its input. For news analysis, only the relevant and live news is collected from a large set of business new. The filtered news is analyzed to predict sentiment around companies. The results of both analyses are integrated together to get a response which gives a recommendation for future increases.

**Methodologies** : LSTM (Long Short-Term Memory), Forecast of Stock Prices, Support Vector Machine (SVM),Efficient Market Hypothesis (EMH)

**Merit** : The system combines price prediction based on historical and real-time data along with new analysis

**Demerit** : less pliable and study model

# **CHAPTER 3**

## **SYSTEM ANALYSIS**

## **CHAPTER - 3**

### **SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEM**

Accurately estimating stock values is a tricky issue due to the complexity of balance sheets. Based on the Pearson coefficient of correlation (PCC) and the Wide Learning System, a multi-indicator features extraction strategy for stock price projection was developed (BLS). The ultimate goal of all this report is to aid traders in arriving at sound investment choices by accurately order to forecast price fluctuations using machine learning algorithms. In this report, we created a new framework out of PCC and BLS and utilised it to forecast stock prices mostly on Stock Exchange of Shenzhen or the Shanghai Exchange for Stocks in the near term.

##### **Disadvantages:**

- Shanghai Stock Exchange price is predicted.
- Deployment is not implemented.
- It is very complex system.

#### **3.2 PROPOSED SYSTEM**

In proposed system, the Datasets from different sources would be combined to form a generalized dataset. After the generalized dataset is prepared it is checked for cleanliness, and then trimmed dataset is analyzed. The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The DataModel which was created using machine learning algorithms are applied on the Training set and based on the test result accuracy, test set prediction is done. In addition, we are going to compare the forecasting results with four machine learning methods, like Adaptive Boosting (Adaboost), Decision Tree, Ridge regression and Lasso regression . Among all algorithms used for testing, the algorithm that provides the best performance is finally used. Then the system is deployed using flask. For predicting the tesla stock problem, ML prediction model is effective because it is strong in preprocessing of data, irrelevant variables, and a mix of continuous, categorical and discrete variables.



### **Advantages:**

- Tesla stock price is predicted.
- Performance metrics are compared.
- Project will be deployed.

## **3.3 FEASIBILITY STUDY**

### **DATA WRANGLING**

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

### **DATA COLLECTION**

This dataset contains 950 records of features extracted from Vehicles, which were then used to find the smog rating from the vehicles.

### **PREPROCESSING**

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

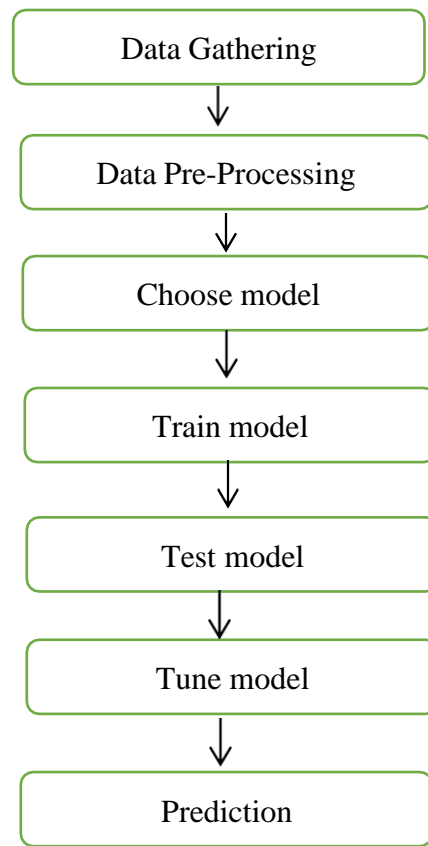
### **BUILDING THE REGRESSION MODEL**

The prediction of tesla stock price, a high accuracy prediction model is effective because of the following reasons: It provides better results in regressions problem.

- It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

### **CONSTRUCTION OF A PREDICTIVE MODEL**

Machine learning needs data gathering have lot of past data's. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can't be used directly. It's used to pre-process then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.



Process of dataflow diagram

### 3.4 PROJECT REQUIREMENTS

#### FUNCTIONAL REQUIREMENTS

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

#### NON-FUNCTIONAL REQUIREMENTS

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

## ENVIRONMENTAL REQUIREMENTS

### Hardware requirements:

Processor	: Intel i3 or later
Hard disk	: minimum 10 GB
RAM	: minimum 4 GB

### Software Requirements:

Operating System	: Windows 10 or later
Tool	: Anaconda with Jupyter Notebook
Software	: Python language
Tool	: Flask

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository. Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly. Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser. The *Jupyter Notebook App* can be executed

on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet. In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their Kernels. The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms including structured, object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. The flask run command is capable of doing more than simply starting the development server. When we enable debug mode, the server will immediately reload if the code changes, and an interactive debugger will appear in the browser if an error occurs during a request.

# **CHAPTER 4**

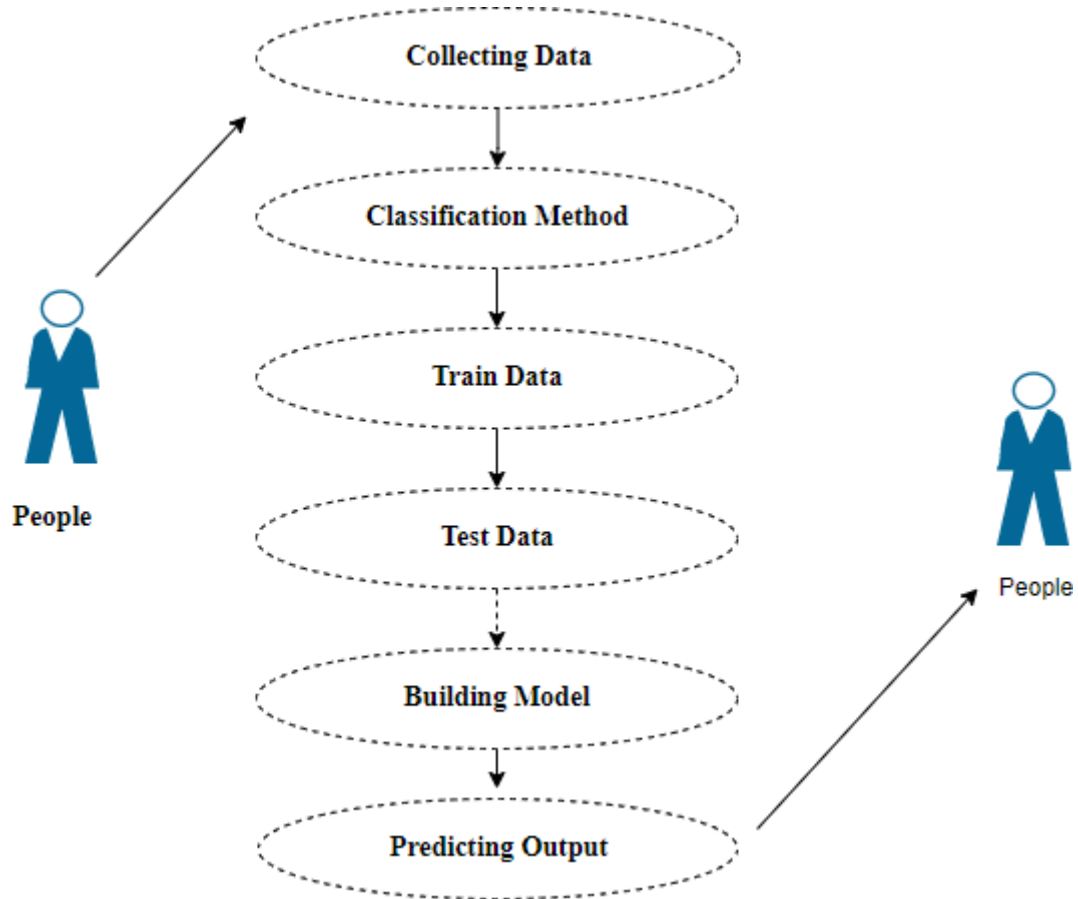
## **SYSTEM DESIGN**

## CHAPTER-4

### SYSTEM DESIGN

#### 4.1. UML DIAGRAMS

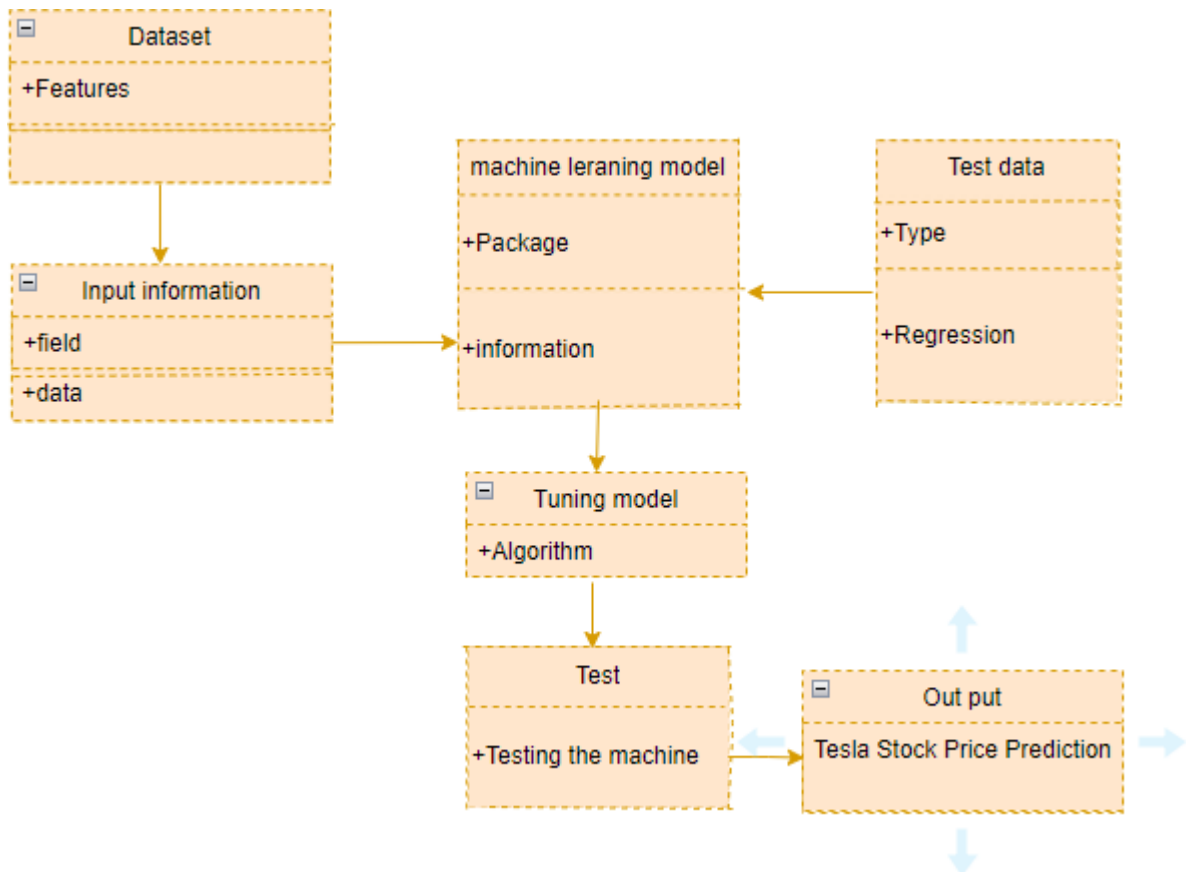
##### 4.1.1. USE CASE DIAGRAM



*Figure 4.1.1 Use case diagram*

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that use cases are nothing but the system functionalities written in an organized manner.

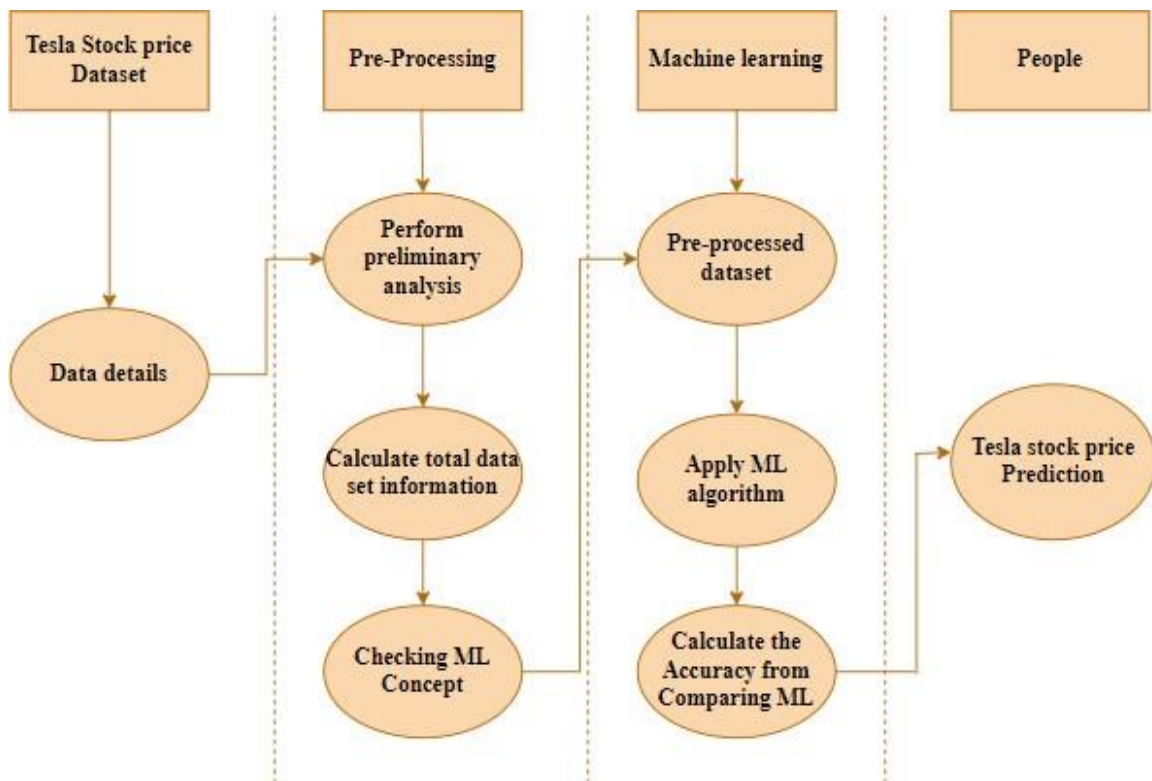
#### 4.1.2 Class Diagram:



**Figure 4.1.2** Class diagram

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance. Responsibility (attributes and methods) of each class should be clearly identified for each class. Minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

#### 4.1.3.Activity Diagram:

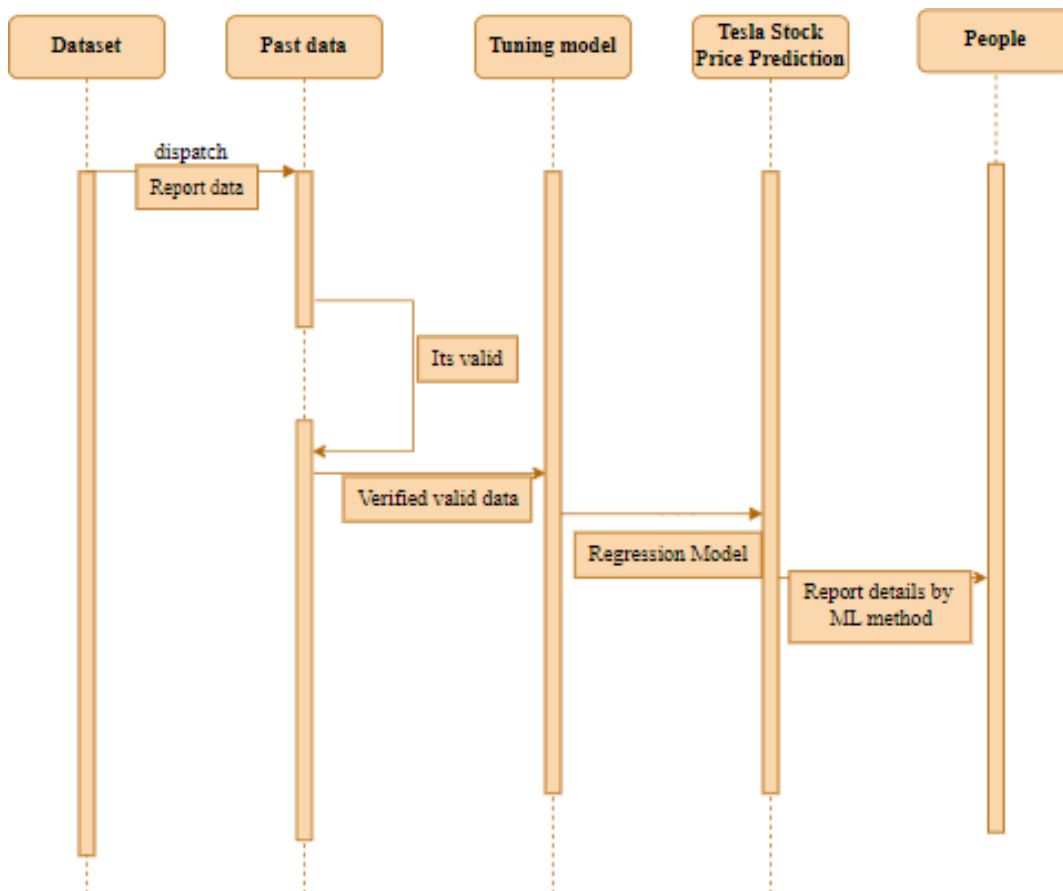


**Figure 4.1.3** Activity diagram

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.



#### 4.1.4. Sequence Diagram:

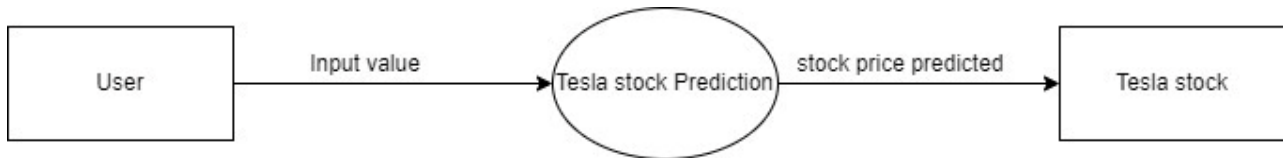


*Figure 4.1.4 Sequence diagram*

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

## 4.2.DATA FLOW DIAGRAM

### 4.2.1 LEVEL 0

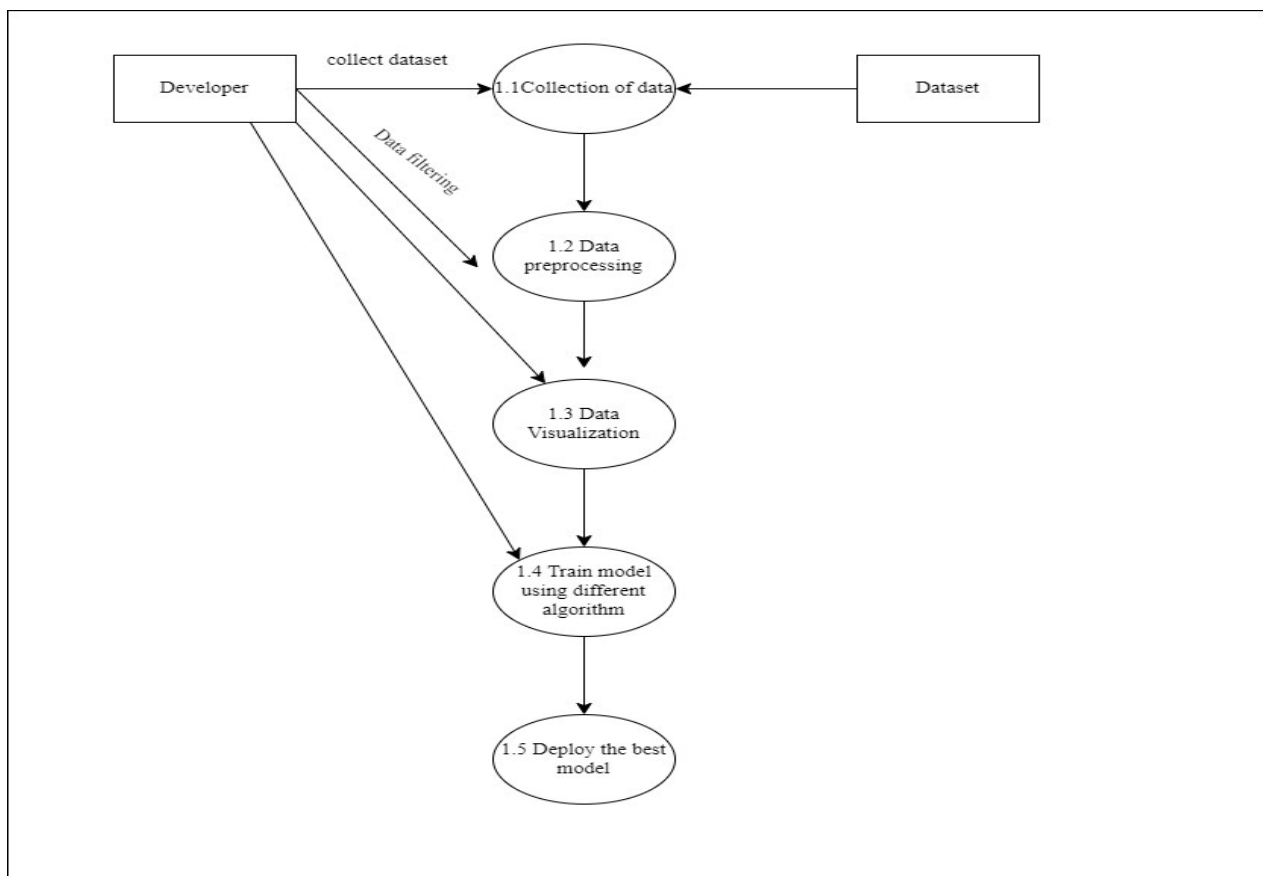


*Figure 4.2.1 level 0*

The Level 0 DFD diagram shows that the input will be given as text/audio by the end user and the emotions are classified

### 4.2.2 LEVEL 2

The below diagram shows the flow of data in this project starting from collection of data to deployment containing data preprocessing, cleaning and the building model for the project.



*Figure 4.2.2 level 1*

# **CHAPTER 5**

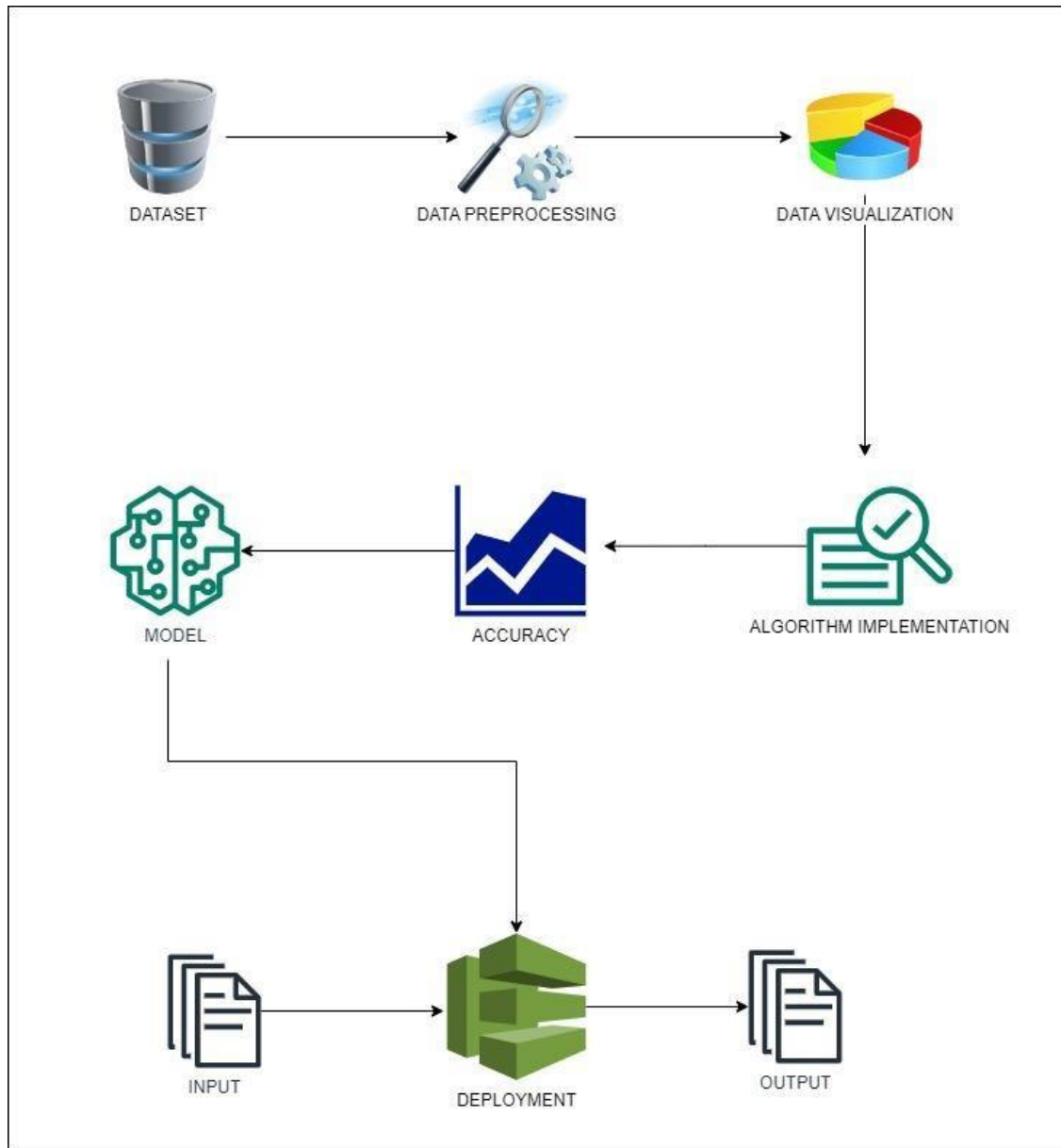
## **SYSTEM**

### **ARCHITECTURE**

## CHAPTER -5

### SYSTEM ARCHITECTURE

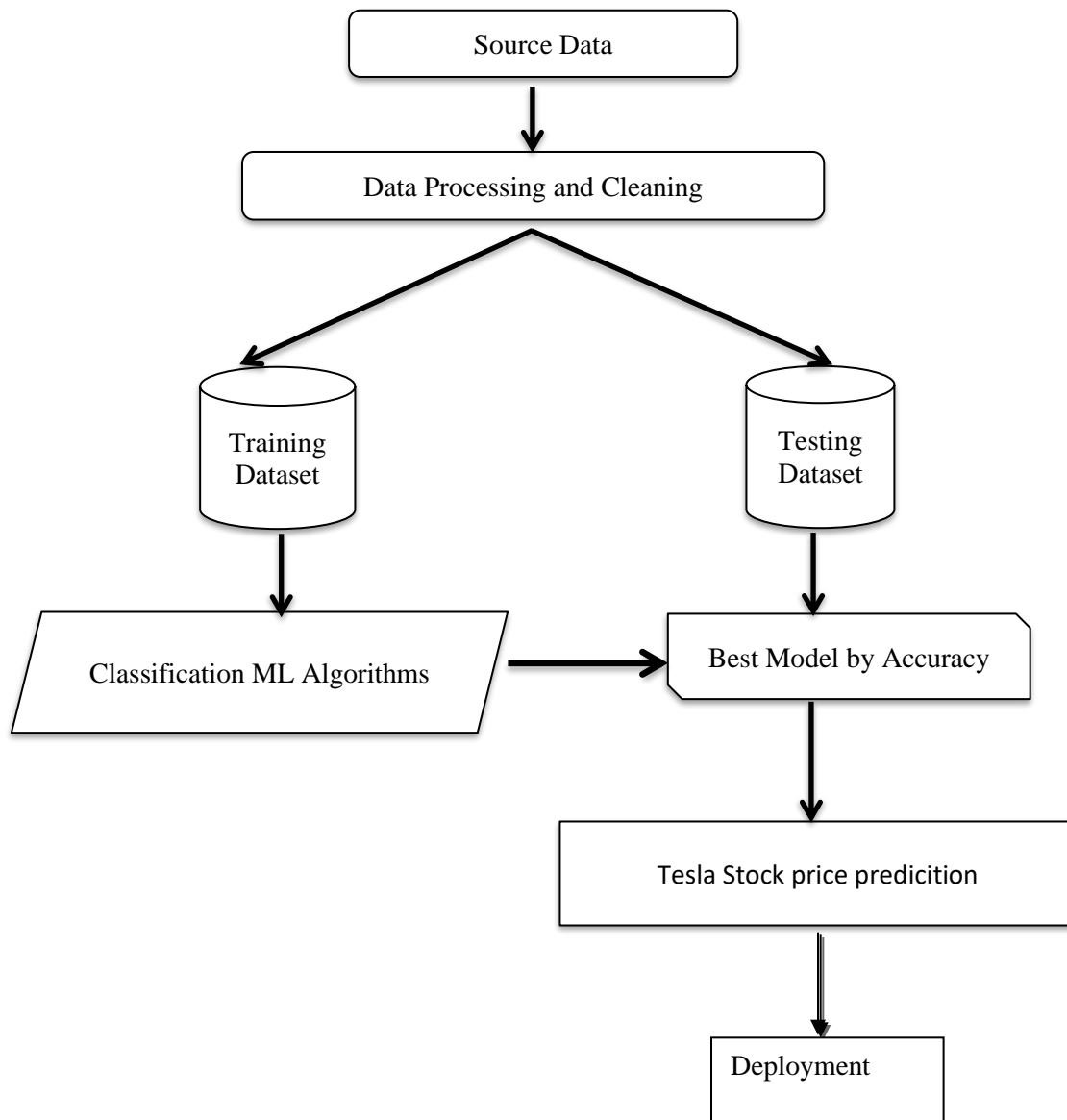
#### 5.1. SYSTEM ARCHITECTURE:



*Figure 5.1 System Architecture*

The architecture diagram shows the processes involved for building the project. The Datasets from different sources would be combined to form a generalized dataset. After the generalized dataset is prepared it is checked for cleanliness, and then the trimmed dataset is analyzed. The preprocessed dataset is represented in a graph or chart. The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using machine learning algorithms are applied on the Training set and based on the test result accuracy, test set prediction is done. In addition, we are going to compare the forecasting results with four machine learning methods, like Adaptive Boosting (Adaboost), Decision Tree, Ridge regression and Lasso regression. Among all algorithms used for testing, the algorithm that provides the best performance is finally used. Then the system is deployed using flask. For predicting the tesla stock problem, ML prediction model is effective because it is strong in preprocessing of data, irrelevant variables, and a mix of continuous, categorical and discrete variables.

## 5.2 WORK FLOW DIAGRAM



**Figure 5.2** Workflow diagram

The workflow diagram shows the flow of proposed work. That is it starts from dataset collection, then processing the data and cleaning. After splitting it into test and train data, the algorithm is implemented to train the model on emotion recognition and then deploying it as webpage by finding the best model.

# **CHAPTER 6**

## **SYSTEM**

### **IMPLEMENTATION**

## **CHAPTER-6**

### **SYSTEM IMPLEMENTATION**

#### **6.1 MODULE DESCRIPTION**

##### **LIST OF MODULES:**


- Data Pre-processing
- Data Visualization
- Implementing AdaBoost Algorithm
- Implementing Decision Tree Algorithm
- Implementing Ridge regression Algorithm
- Implementing Lasso regression Algorithm
- Deployment Using Flask

##### **6.1.1. DATA PRE-PROCESSING:**

Machine learning validation approaches are used to obtain the margin of error of such Machine Learning (ML) system, which would be close to the genuine error rate of something like the dataset. If the data is substantial enough to be considered accurately representing the population, validation approaches may not be required. Yet, in real-world circumstances, it is necessary to work with data samples that are not always accurately reflecting the population of the given dataset. To identify the standard error, duplicate value, and data type description, whether it is a float variable or an integer variable. For tuning model hyper parameters, a sample of data is employed to offer an empirical assessment of an observed performance on the training dataset.

When skill from the training data is used to configure the model, overall evaluation becomes more unbalanced. The training set can be utilized to test a particular model, however this is done frequently. The modelling hyper parameters are adjusted while using data by machine learning specialists. The process of gathering data, analysing it, and dealing with its structure, quality, and substance can take a lot of time. Understanding your data and its characteristics is helpful during the data evaluation stage since it will assist you decide which algorithm to employ to create your model. A variety of data cleaning jobs are underwent using Python's Pandas library, with a particular emphasis on the most important data cleaning work, missing values, and the ability to clean information more rapidly. It would like to devote almost no time cleaning data and far more length exploring and modelling.





	Date	Open	High	Low	Close	Adj Close	Volume
0	28-06-2010	3.800	6.084	3.508	3.840	3.840	246560000
1	05-07-2010	4.000	4.000	2.996	3.480	3.480	127753000
2	12-07-2010	3.590	4.300	3.380	4.128	4.128	77194500
3	19-07-2010	4.274	4.450	3.900	4.258	4.258	35878500
4	26-07-2010	4.300	4.300	3.910	3.988	3.988	15260000
5	02-08-2010	4.100	4.436	3.904	3.918	3.918	21998500
6	09-08-2010	3.980	3.996	3.478	3.664	3.664	21083000
7	16-08-2010	3.690	3.918	3.652	3.820	3.820	12050500
8	23-08-2010	3.818	4.078	3.712	3.940	3.940	15389500
9	30-08-2010	3.940	4.260	3.866	4.210	4.210	11752500
10	06-09-2010	4.122	4.210	3.952	4.034	4.034	6473000

	Date	Open	High	Low	Close	Adj Close	Volume
0	28-06-2010	3.800000	6.084000	3.508000	3.840000	3.840000	246560000
1	05-07-2010	4.000000	4.000000	2.996000	3.480000	3.480000	127753000
2	12-07-2010	3.590000	4.300000	3.380000	4.128000	4.128000	77194500
3	19-07-2010	4.274000	4.450000	3.900000	4.258000	4.258000	35878500
4	26-07-2010	4.300000	4.300000	3.910000	3.988000	3.988000	15260000
...	...	...	...	...	...	...	...
603	17-01-2022	1026.609985	1070.790039	940.500000	943.900024	943.900024	105363500
604	24-01-2022	904.760010	987.690002	792.010010	846.349976	846.349976	208309200
605	31-01-2022	872.710022	943.700012	862.049988	923.320007	923.320007	132213500
606	07-02-2022	923.789978	947.770020	850.700012	860.000000	860.000000	103196000
607	14-02-2022	861.570007	898.880005	853.150024	875.760010	875.760010	22585500

608 rows x 7 columns

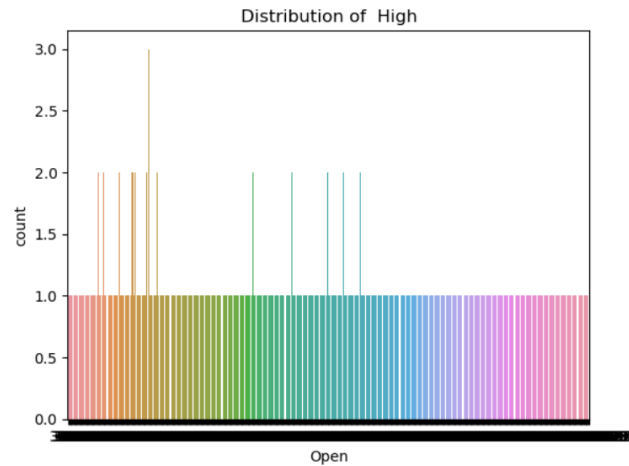
**Figure 6.1.1** Screenshot of Data Pre-processing

## 6.1.2. DATA VISUALIZATION

Data visualization is a crucial ability in machine learning and applied statistics. In fact, the main focus of statistics is on numerical estimates and descriptions of data. An essential set of tools for obtaining a qualitative understanding is provided by data visualization. This can be useful for discovering trends, corrupt data, outliers, and much more while exploring and getting to know a dataset. Data visualizations can be utilised to convey and illustrate critical relationships in plots and charts that are more visceral and engaging to stakeholders than measurements of association or importance with a little subject knowledge. It will suggest a deeper look into some of the books suggested at the conclusion.

Data visualization and exploratory data analysis are entire fields in themselves. Data may not always make sense unless it is presented visually, such as through charts and graphs. Both applied statistics and applied machine learning value fast visualization of data samples and other objects. It will identify the many plot types you must be aware of while Python data visualization techniques and how to apply them to your own data.

1. Learn to visualize categorical data using bar charts and time series data using line graphs.
2. Using histograms and box graphs to condense data distributions.



*Figure 6.1.2 Screenshot of Data Visualization*

### 6.1.3. ML MODEL DEVELOPMENT

It is crucial to regularly compare the results of various machine learning algorithms, and it will be found that using scikit-learn and Python, it is possible to develop a test harness for this purpose. You can apply this test harness as a model for your own machine learning issues and include additional and various algorithms to contrast. There will be variations in the performance attributes of each model. You may estimate each model's potential accuracy on unobserved data by using resampling techniques like cross validation. It must be able to select one or two of the best models from the group of models you have developed using these estimates. In order to view a fresh dataset from various angles, it is a good idea to visualize the data using a variety of ways. The choice of models follows the same logic. In order to select the one or two that will be used for finalization, you need consider a lot of various angles when evaluating the machine learning algorithms' predicted accuracy. One way to achieve this is to demonstrate the average accuracy, variance, and other characteristics of the distribution of model accuracies using various visualization techniques.

You will learn precisely how to achieve that in Python using scikit-learn in the following section. Making sure that each algorithm is evaluated uniformly on the same data is essential for conducting a fair comparison of machine learning algorithms, and this may be done by requiring that each algorithm be tested using a uniform test harness.

$$\text{Train set} + \text{Test set} = \text{Total Dataset}$$

The below 4 different algorithms are compared:

- Ada Boost
- Decision Tree
- Ridge
- Lasso

### 6.1.3.1 ADABOOST ALGORITHM:

AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous regressors. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.

Although AdaBoost is typically used to combine weak base learners (such as decision stumps), it has been shown that it can also effectively combine strong base learners (such as deep decision trees), producing an even more accurate model.

Every learning algorithm tends to suit some problem types better than others, and typically has many different parameters and configurations to adjust before it achieves optimal performance on a dataset. AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box regressor. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-regressor examples.

---

**Input:** training sample  $S$ , Classifier  $L$ , iterations  $I$   
**Output:** result  $L_E$   
**Training:**  
*normalize the weights and make the total weight is  $n$*   
 *$S_i$  = sample from  $S$  according to the distribution*  
 *$L_i$  = train a classifier on  $S_i$  via  $L$*   

$$e_i = \frac{1}{m} \sum_{x_i \in S_i: L_i(x_i) \neq y_i} \text{weight}(x_i)$$

$$\beta_i = \frac{e_i}{1 - e_i}$$
*weight( $x_i$ ) = weight( $x_i$ ) $\beta_i$ , for all  $x_i$  where  $L_i(x_i) = y_i$*   
*end for*  

$$L_E = \arg \max_{y \in Y} \sum_{i: L_i(x) = y} \log(1 / \beta_i)$$

---

*Pseudocode of AdaBoost*

### Finding the R2 score

```
In [22]: R2 = (r2_score(y_test, predictD)*100)
print("The Accuracy of adaboost regressor:", R2)
print("")
The Accuracy of adaboost regressor: 99.06166312809103
```

### 6.1.3.2. DECISION TREE ALGORITHM:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems. It is a tree-structured regressor, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further splits the tree into subtrees.

```
GenDecTree(Sample S, Features F)
Steps:
1. If stopping_condition(S, F) = true then
    a. Leaf = createNode()
    b. leafLabel = classify(s)
    c. return leaf
2. root = createNode()
3. root.test_condition = findBestSplit(S, F)
4. V = {v | v a possible outcome of root.test_condition}
5. For each value v in V:
    a. S_v = {s | root.test_condition(s) = v and s in S};
    b. Child = TreeGrowth(S_v, F);
    c. Add child as descent of root and label the edge {root -> child} as v
6. return root
```

*Pseudocode of decision tree*

Find Accuracy Score

```
In [20]: R2=(r2_score(y_test,predictD)*100)
print('Accuracy result of Decision Tree regressor is :',R2)
print("")
Accuracy result of Decision Tree regressor is : 99.83258573196214
```

### 6.1.3.3. RIDGE ALGORITHM:

Ridge regression is a regularization technique, which is used to reduce the complexity of the model. It is also called as L2 regularization. In this technique, the cost function is altered by adding the penalty term to it. The amount of bias added to the model is called Ridge Regression penalty.

Lasso tends to do well if there are a small number of significant parameters and the others are close to zero (ergo: when only a few predictors actually influence the response). Ridge works well if there are many large parameters of about the same value (ergo: when most predictors impact the response).

Ridge regression aims at reducing the standard error by adding some bias in the estimates of the regression. The reduction of the standard error in regression estimates significantly increases the reliability of the estimates.

The equation for ridge regression will be:

$$L(x, y) = \text{Min}(\sum_{i=1}^n (y_i - w_i x_i)^2 + \lambda \sum_{i=1}^n (w_i)^2)$$

A general linear or polynomial regression will fail if there is high collinearity between the independent variables, so to solve such problems, Ridge regression can be used.

Find the R2\_score

```
In [23]: R2_SCORE = (r2_score(y_test, predictR)*100)
print("The Accuracy of Ridge regressor :", R2_SCORE)
print("")
The Accuracy of Ridge regressor : 99.82611262743984
```

#### 6.1.3.4. LASSO ALGORITHM:

Lasso regression is a regularization technique. It is used over regression methods for a more accurate prediction. This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination. Lasso Regression uses L1 regularization technique. It is used when we have more features because it automatically performs feature selection.

The equation for Lasso regression will be:

$$L(x, y) = \text{Min} \left( \sum_{i=1}^n (y_i - w_i x_i)^2 + \lambda \sum_{i=1}^n |w_i| \right)$$

Find the r2score

```
In [21]: R2_SCORE = (r2_score(y_test, predictLR)*100)
print("The Accuracy of Lasso:", R2_SCORE)
print('')
```

The Accuracy of Lasso: 99.74023051964012

#### 6.1.4. DEPLOYMENT

The model with high accuracy is converted to .pkl file in order to use it for webpage. Thus the best model obtained from the four algorithm is Decision Tree . This model is deployed as a webpage using Flask framework. Using this framework, we will connect the saved model of decision tree model and HTML, CSS code to create a webpage.

# **CHAPTER 7**

# **PERFORMANCE**

# **EVALUATION**

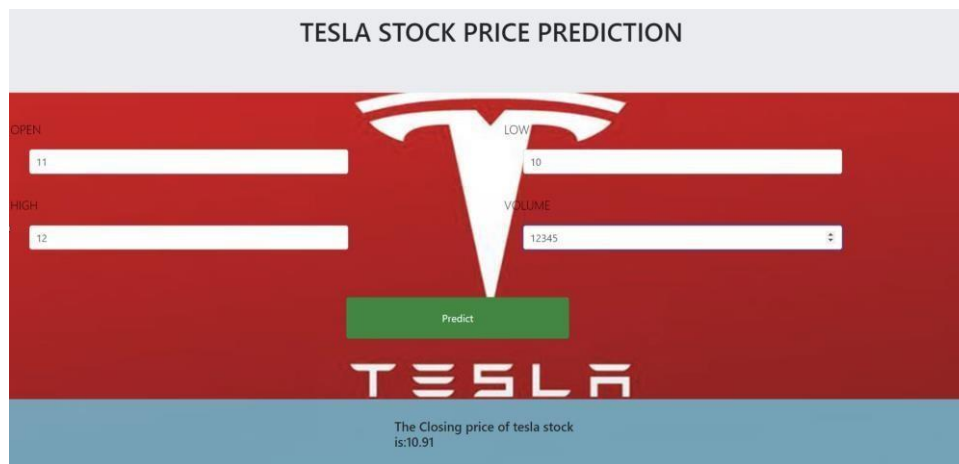
## CHAPTER-7 PERFORMANCE EVALUATION

### 7.1. RESULTS AND DISCUSSIONS:



The screenshot shows a web page titled "TESLA STOCK PRICE PREDICTION". The page has a red background with a white Tesla logo in the center. There are four input fields: "OPEN" with the value "open", "LOW" with the value "Low", "HIGH" with the value "High", and "VOLUME" with the value "Volume". A green "Predict" button is located below the input fields. The Tesla logo is also visible at the bottom of the page.

**Figure 7.1** Screenshot of web page for Tesla stock price prediction



The screenshot shows the same web page as Figure 7.1, but with the predicted closing price displayed. The input fields now contain the values: "OPEN" is "11", "LOW" is "10", "HIGH" is "12", and "VOLUME" is "12345". The "Predict" button is still present. At the bottom of the page, below the Tesla logo, the text "The Closing price of tesla stock is:10.91" is displayed.

**Figure 7.2** Screenshot of the predicted stock price is displayed

The output screen includes a heading that describes the purpose of the application, which is to predict the tesla stock price. There are four input boxes where users can enter the values of Open, Low, High, Volume to predict the stock price. A "Predict" button is provided to initiate the prediction process. After the analysis is completed, the output displays the closing price of tesla stock. The example shows how the output screen will work. Depending on the specific implementation of the application and the machine learning model behind it, the output could predict closing price for different set of inputs.



## 7.2. COMPARATIVE ANALYSIS

Finding the R2 score

```
In [22]: R2 = (r2_score(y_test, predictD)*100)
print("The Accuracy of adaboost regressor:", R2)
print("")
```

The Accuracy of adaboost regressor: 99.06166312809103

**Figure 7.2.1** Screenshot of Accuracy using Adaboost Regression

Find the R2\_score

```
In [23]: R2_SCORE = (r2_score(y_test, predictR)*100)
print("The Accuracy of Ridge regressor :", R2_SCORE)
print("")
```

The Accuracy of Ridge regressor : 99.82611262743984

**Figure 7.2.2** Screenshot of Accuracy using Ridge Regression

Find the r2score

```
In [21]: R2_SCORE = (r2_score(y_test, predictLR)*100)
print("The Accuracy of Lasso:", R2_SCORE)
print('')
```

The Accuracy of Lasso: 99.74023051964012

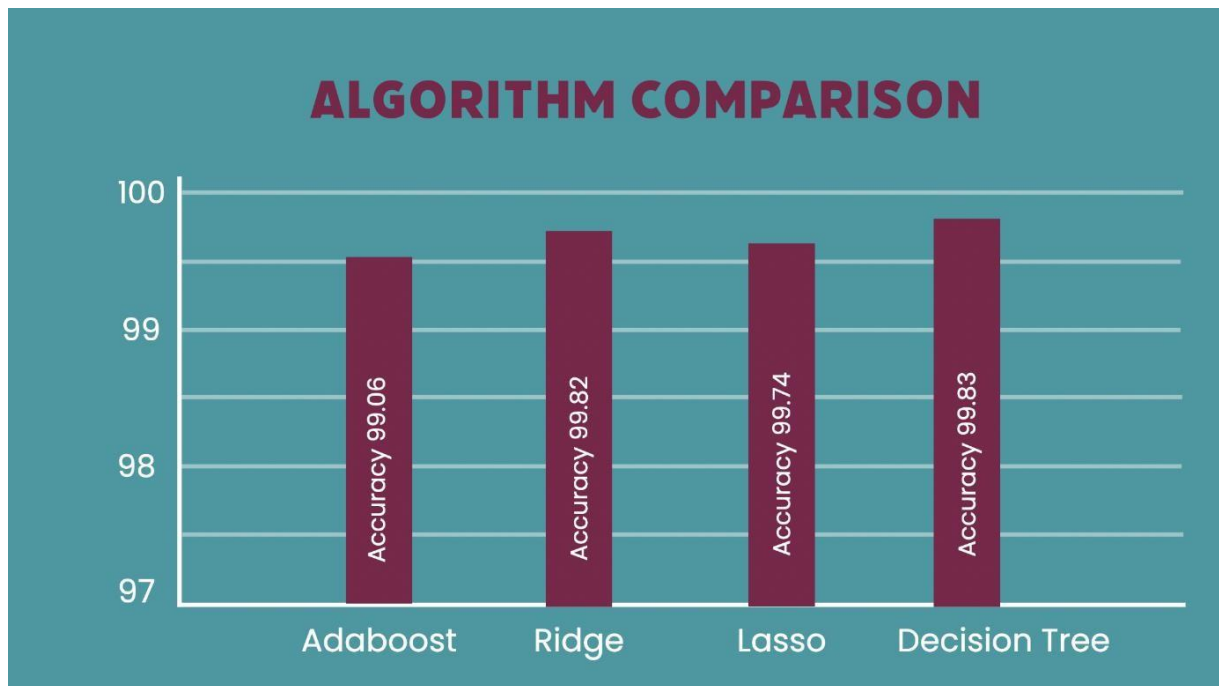
**Figure 7.2.3** Screenshot of Accuracy using Lasso Regression

Find Accuracy Score

```
In [20]: R2=(r2_score(y_test,predictD)*100)
print('Accuracy result of Decision Tree regressor is :',R2)
print("")
```

Accuracy result of Decision Tree regressor is : 99.83258573196214

*Figure 7.2.4 Screenshot of Accuracy using Decision Tree Regression*



*Figure 7.2.5 Comparison of four algorithms*

	Mean Absolute Error	Mean Absolute Percentage Error	Explained Variance Score	Mean Squared Error	Median Absolute Error	R2 score
Adaptive boosting	19.57	1.66	99.07	611.39	19.17	99.06
Ridge Regression	4.00	0.04	99.82	87.95	0.95	99.82
Lasso Regression	5.15	0.07	99.75	199.66	1.19	99.74
Decision Tree Regression	4.06	0.04	99.83	92.38	1.03	99.83

*Table 7.2.1 Performance Metric*

Thus, we have built four models based on four algorithms and each has yielded different accuracies. The first model which was built using Adaboost regression yielded an accuracy of 99.06%., the Ridge regression yielded an accuracy of 99.82%, the Lasso regression yielded an accuracy of 99.74% and Decision Tree regression model yielded an accuracy of 99.83%. From this we can say that Decision tree algorithm yields the highest accuracy. It is then followed by Ridge regression. The least accuracy was yielded by Adaboost regression algorithm. These models were built using Machine Learning algorithms.

# **CHAPTER 8**

# **CONCLUSION**

## **CHAPTER-8**

### **CONCLUSION**

#### **8.1. CONCLUSION AND FUTURE ENHANCEMENTS**

##### **CONCLUSION:**

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The performance of these models largely depends on the quality and size of the training data, feature engineering, and the choice of the regression algorithm. The best accuracy on public test set of higher accuracy score algorithm will be found out. The founded one is used in the application which can help to find the tesla stock price.

##### **FUTURE WORK:**

Deploying the project in the cloud. To optimize the work to implement in the IOT system. In future ,we can extend this application for predicting crypto currency trading and also we can add sentiment analysis for Better prediction. It would be a useful alteration to include more than just one stock in the analysis. This would make prediction more challenging, but, at the same time, there are likely to be correlations between different companies that may be learned by the model architecture

## APPENDIX- SAMPLE DATASET

### SAMPLE DATASET

Open-High-Low-Close Charts show the high and low price a stock attained for a particular period of time as well as the opening and closing prices of the stock for the same period. The chart shows the same type of vertical lines displayed on a high-low chart.

Open represents the start value of the stock for the particular day, High represents the highest value of the stock on that particular date, Low represents the lowest value of the stock on the particular date, Close represents the close value of the stock, Adjust close represents the nearby close value of the stock and Volume represents the overall shares that has been bought and sold.

Date	Open	High	Low	Close	Adj Close	Volume
29-06-2010	19	25	17.54	23.89	23.89	18766300
30-06-2010	25.79	30.42	23.3	23.83	23.83	17187100
01-07-2010	25	25.92	20.27	21.96	21.96	8218800
02-07-2010	23	23.1	18.71	19.2	19.2	5139800
06-07-2010	20	20	15.83	16.11	16.11	6866900
07-07-2010	16.4	16.63	14.98	15.8	15.8	6921700
08-07-2010	16.14	17.52	15.57	17.46	17.46	7711400
09-07-2010	17.58	17.9	16.55	17.4	17.4	4050600
12-07-2010	17.95	18.07	17	17.05	17.05	2202500
13-07-2010	17.39	18.64	16.9	18.14	18.14	2680100
14-07-2010	17.94	20.15	17.76	19.84	19.84	4195200
15-07-2010	19.94	21.5	19	19.89	19.89	3739800
16-07-2010	20.7	21.3	20.05	20.64	20.64	2621300
19-07-2010	21.37	22.25	20.92	21.91	21.91	2486500
20-07-2010	21.85	21.85	20.05	20.3	20.3	1825300
21-07-2010	20.66	20.9	19.5	20.22	20.22	1252500
22-07-2010	20.5	21.25	20.37	21	21	957800
23-07-2010	21.19	21.56	21.06	21.29	21.29	653600
26-07-2010	21.5	21.5	20.3	20.95	20.95	922200
27-07-2010	20.91	21.18	20.26	20.55	20.55	619700

28-07-2010	20.55	20.9	20.51	20.72	20.72	467200
29-07-2010	20.77	20.88	20	20.35	20.35	616000
30-07-2010	20.2	20.44	19.55	19.94	19.94	426900
02-08-2010	20.5	20.97	20.33	20.92	20.92	718100
03-08-2010	21	21.95	20.82	21.95	21.95	1230500
04-08-2010	21.95	22.18	20.85	21.26	21.26	913000
05-08-2010	21.54	21.55	20.05	20.45	20.45	796200
06-08-2010	20.1	20.16	19.52	19.59	19.59	741900
09-08-2010	19.9	19.98	19.45	19.6	19.6	812700
10-08-2010	19.65	19.65	18.82	19.03	19.03	1281300
11-08-2010	18.69	18.88	17.85	17.9	17.9	797600
12-08-2010	17.8	17.9	17.39	17.6	17.6	691000
13-08-2010	18.18	18.45	17.66	18.32	18.32	634000
16-08-2010	18.45	18.8	18.26	18.78	18.78	485800
17-08-2010	18.96	19.4	18.78	19.15	19.15	447900
18-08-2010	19.59	19.59	18.6	18.77	18.77	601300
19-08-2010	18.54	19.25	18.33	18.79	18.79	579100
20-08-2010	18.65	19.11	18.51	19.1	19.1	296000
23-08-2010	19.09	20.39	19	20.13	20.13	1088100
24-08-2010	19.25	19.71	18.95	19.2	19.2	673100
25-08-2010	19.16	19.98	18.56	19.9	19.9	503300
26-08-2010	19.89	20.27	19.6	19.75	19.75	433800
27-08-2010	19.75	19.87	19.5	19.7	19.7	379600
30-08-2010	19.7	20.19	19.61	19.87	19.87	732800
31-08-2010	19.66	19.79	19.33	19.48	19.48	201100
01-09-2010	19.62	20.69	19.6	20.45	20.45	494900
02-09-2010	20.37	21.24	20.31	21.06	21.06	487100
03-09-2010	20.87	21.3	20.66	21.05	21.05	434600
07-09-2010	20.61	21	20.5	20.54	20.54	243400

## APPENDIX 2 - SAMPLE CODING

### SAMPLE CODING

#### MODULE 1

#### DATA PRE-PROCESSING:

```
# Import the libraries.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Read the Dataset.
data = pd.read_csv('Dataset.csv')
import warnings
warnings.filterwarnings('ignore')
data.head()
df = data.dropna()
df.columns

# Describe the data in statistical view.
df.describe()
df.info()
df.rename({'Adjust_close':'Adjust close'})
df.duplicated()
df.isnull().sum()
df['Open'].unique()
df['Close'].unique()
df.corr()
df.groupby(['Open','High']).groups
df.nunique()
df.size
```



```
df.shape
df.tail()
```

## MODULE 2

### DATA VISUALIZATION:

```
# Import the Libraries.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
data = pd.read_csv('Dataset.csv')
data.head()
data.size
data.columns
data.isnull()
df = data.dropna()
df.head()
df.size

# Display the Heatmap
fig, ax = plt.subplots(figsize=(22,10))
sns.heatmap(df.corr(),annot = True, fmt='0.2%',cmap = 'vlag',ax=ax)

#YlGnBu, Reds, tab10, rocket, mako, flare, crest, magma, viridis, cubehelix, YlOrBr, vlag,
icefire, Spectral, coolwarm, Blues

#plotting graph for distribution
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x = "Open", data = df)
df.loc[:, 'High'].value_counts()
plt.title('Distribution of High')

# Display the distplot.
sns.distplot(x = df['High'], bins = 10)

# Display the Jointplot.
sns.jointplot(x = df['High'], y = df['Low'], kind = 'scatter')

# Display the Pairplot
sns.pairplot(df)

# Display the barplot.
sns.barplot(x = df['Open'], y = df['Close'])

# Display the stripplot.
```

```
sns.stripplot(y = df['Low'], x = df['High'])
# Display the subplots of clustermap
#fig, ax = plt.subplots(figsize=(22,10))
sns.clustermap(df.corr(),annot = True, fmt='0.2%',cmap ='coolwarm')
```

## MODULE 3

### IMPLEMENTING ADABOOST ALGORITHM

```
import pandas as pd
import matplotlib.pyplot as pd
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
data = pd.read_csv('Dataset.csv')
data.head()
df = data.dropna()
df.shape
df['Open'].unique()
df.columns
df.duplicated()
del df['Date']
df.columns
df.describe()
X = df.drop(labels='Close', axis=1)
#Response variable
y = df.loc[:, 'Close']
```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
Implementing Adaboost regressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error,
explained_variance_score, r2_score, mean_squared_error
from sklearn.metrics import median_absolute_error
```

#### Training

```
svr= AdaBoostRegressor()

svr.fit(X_train,y_train)

predictD = svr.predict(X_test)
```

#### Finding the mean absolute error

```
MAE=(mean_absolute_error(y_test,predictD))
print("MEAN_ABSOLUTE_ERROR:", MAE)
print("")
```

### **Finding the mean absolute error**

```
MAPE =(mean_absolute_percentage_error(y_test, predictD))
print("MEAN_ABSOLUTE_PERCENTAGE_ERROR:", MAPE)
print("")
```

### **Finding the Accuracy**

```
EVS = (explained_variance_score(y_test, predictD)*100)
print("ACCURACY RESULT OF ADABOOST REGRESSOR IS :", EVS)
print("")
```

### **Finding the Mean squared error**

```
MSE = (mean_squared_error(y_test, predictD))
print("MEAN_SQUARED_ERROR:", MSE)
print("")
```

### **Finding the Median absolute error**

```
MEDIAN_S_E = (median_absolute_error(y_test, predictD))
print("MEDIAN_ABSOLUTE_ERROR:", MEDIAN_S_E)
print("")
```

### **Finding the R2 score**

```
R2 = (r2_score(y_test, predictD))
print("R2 score:", R2)
print("")
```

## **MODULE 4**

### **IMPLEMENTING DECISION TREE ALGORITHM**

```
# Importing the libraries.
import pandas as pd
import numpy as np
# Load the Dataset.
data = pd.read_csv('Dataset.csv')
data.head()
df = data.dropna()
# Delete the Columns.
del df['Date']
del df['Adj Close']
df.head()
df.shape
df.info
# Display the Dataset in stastical perspective.
df.describe()
```

```
X = df.drop(labels='Close', axis=1)
#Response variable
y = df.loc[:, 'Close']
X
```

```
#We'll use a test size of 30%. We also stratify the split on the response variable, which is
very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
Implementing Decision tree classifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error,
median_absolute_error, explained_variance_score, r2_score
```

### **Training**

```
DT= DecisionTreeRegressor()
```

```
DT.fit(X_train,y_train)
```

```
predictD = DT.predict(X_test)
```

### **Finding Mean absolute error**

```
MAE= (mean_absolute_error(y_test,predictD))
print("MEAN_ABSOLUTE_ERROR:", MAE)
print("")
```

### **Find Mean squared error**

```
MSE=(mean_squared_error(y_test,predictD))
print('MEAN SQUARED ERROR VALUE IS :',MSE)
print("")
```

### **Find Median absolute error**

```
MedianAE=(median_absolute_error(y_test,predictD))
print('MEDIAN ABSOLUTE ERROR VALUE IS :',MedianAE)
print(" ")
```

### **Find explained variance score**

```
EVS=(explained_variance_score(y_test,predictD)*100)
print('EXPLAINED_VARIANCE_SCORE:',EVS)
print("")
```

### **Find Accuracy Score**

```
R2=(r2_score(y_test,predictD)*100)
print('Accuracy result of Decision Tree regressor is :',R2)
print("")
import joblib
```

```
joblib.dump(DT, 'Dt.pkl')
```

## MODULE 5

### IMPLEMENTING RIDGE REGRESSION ALGORITHM

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
data = pd.read_csv('Dataset.csv')
data.head()
data.drop_duplicates()
df = data.dropna()
df.head()
df.describe()
df.size
df.columns
del df['Date']
del df['Adj Close']
df.head()
X = df.drop(labels='Close', axis=1)
#Response variable
y = df.loc[:, 'Close']

#We'll use a test size of 30%. We also stratify the split on the response variable, which is
very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
Ridge Regressor.
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error, median_absolute_error,
mean_absolute_percentage_error, r2_score, explained_variance_score
```

#### Training

```
RFT = Ridge()
```

```
RFT.fit(X_train, y_train)
```

```
predictR = RFT.predict(X_test)
```

#### Find The mean absolute error.

```
MAE = (mean_absolute_error(y_test, predictR))
print("MEAN ABSOLUTE ERROR :", MAE)
print("")
```

#### Find the mean absolute percentage error.

```
MAPE = (mean_absolute_percentage_error(y_test, predictR))*100
```

```
print("MEAN ABSOLUTE PERCENTAGE ERROR:", MAPE)
print("")
```

#### **Find the Median absolute error.**

```
MEDIAN_AE = (median_absolute_error(y_test,predictR))
print("MEDIAN ABSOLUTE ERROR:", MEDIAN_AE)
print("")
```

#### **Find the Accuracy score.**

```
EVS = (explained_variance_score(y_test,predictR))
print("Explained variance score:", EVS)
print("")
```

#### **Find the R2\_score**

```
R2_SCORE = (r2_score(y_test, predictR)*100)
print("R2_SCORE :", R2_SCORE)
print("")
```

## **MODULE 6**

### **IMPLEMENTING LASSO REGRESSION ALGORITHM**

# Import the Libraries.

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

```
data = pd.read_csv('Dataset.csv')
data.head()
data.columns
data.info()
df = data.dropna()
df.columns
del df['Date']
del df['Adj Close']
df.head()
df.shape
```

# Split the test and training dataset.

```
X = df.drop(labels='Close', axis=1)
```

# Response variable.

```
y = df.loc[:, 'Close']
```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Implementing Lasso algorithms.

```
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error,
mean_squared_error, r2_score, explained_variance_score
from sklearn.metrics import median_absolute_error
```

### **Training**

```
LR = Lasso()
```

```
LR.fit(X_train,y_train)
```

```
predictLR = LR.predict(X_test)
```

### **Find the mean absolute error**

```
MAE = (mean_absolute_error(y_test,predictLR))
print("MEAN ABSOLUTE ERROR:", MAE)
print("")
```

### **Find the Mean squared error**

```
MSE = (mean_squared_error(y_test,predictLR))
print("MEAN SQUARED ERROR:", MSE)
print("")
```

### **Find the Mean absolute error**

```
MAPE = (mean_absolute_percentage_error(y_test,predictLR))
print("MEAN ABSOLUTE PERCENTAGE ERROR :", MAPE)
print("")
```

### **Find the Median absolute error**

```
MEDIAN_SE = (median_absolute_error(y_test, predictLR))
print("MEDIAN ABSOLUTE ERROR:", MEDIAN_SE)
print("")
```

### **Find the Accuracy of Lasso**

```
EVS = (explained_variance_score(y_test, predictLR))
print("Explained variance score :", EVS)
print("")
```

### **Find the r2score**

```
R2_SCORE = (r2_score(y_test, predictLR)*100)
print("The Accuracy of Lasso:", R2_SCORE)
print("")
import joblib
joblib.dump(LR,'Lr.pkl')
```

## MODULE 7

### DEPLOYMENT

#### HTML CODE

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>TESLA</title>
<link rel="stylesheet" href="{ { url_for('static', filename='css/bootstrap.min.css') } }">
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<style>
.back{
  background-image: url("{ { url_for('static', filename='image/tesla.jpg') } }");
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-size: 100% 100%;
}
.white{
color: black;
font-size: larger;
font-weight: 200;
}
.space{
margin:10px 30px;
padding:10px 10px;
background: rgb(255, 255, 255);
width:500px
}
.gap{
padding:10px 20px;
}
</style>

</head>

<body class="back">
  <div>
    <div class="jumbotron">
      <h1 style="text-align:center">TESLA STOCK PRICE PREDICTION</h1>
```



```

    </div>
    <!-- Main Input For Receiving Query to our ML -->
    <form class="form-group" action="{ { url_for('predict') } }" method="post">

        <div class="row">
            <div class="gap col-md-6 ">

                <label class="white" for="">OPEN</label>
                <input type="number" class="space form-control" step="0.01"
name="Open" placeholder="open" required="required" /><br>

                <label class="white" for="">HIGH</label>
                <input type="number" class="space form-control" step="0.01"
name="High" placeholder="High" required="required" /><br>

            </div>

        <div class="gap col-md-6">

            <label class="white" for="">LOW</label>
            <input type="number" class="space form-control" step="0.01"
name="Low" placeholder="Low" required="required" /><br>

            <label class="white" for="">VOLUME</label>
            <input type="number" class="space form-control" step="0.01"
name="Volume" placeholder="Volume" required="required" /><br>

        </div>
    </div>

    <div style="padding:2% 35%">
        <button type="submit" class="btn btn-success btn-block"
style="width:350px;padding:20px">Predict</button>
    </div>

    </form>

    <br>
    <br>
    <div style="background:rgba(80, 221, 246, 0.721);padding:2% 40%;">
        <h5> {{ prediction_text }} </h5>
    </div>
    </div>

</body>

```

</html>

## FLASK CODE

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
import joblib
```

```
app = Flask(__name__)
model = joblib.load('model.pkl')
```

```
@app.route('/')
def home():
    return render_template('index.html')
```

```
@app.route('/predict',methods=['POST'])
def predict():
    """
    For rendering results on HTML GUI
    """
    int_features = [(x) for x in request.form.values()]
    int_features = [int(i) for i in int_features]
    final_features = [np.array(int_features)]
    print(final_features)
    prediction = model.predict(final_features)

    output = prediction[0]
    print(output)
    if output == 'Moderate':
        output = 'Moderate'
    elif output == 'Poor':
        output = 'Poor'
    elif output == 'Very Poor':
```

```
        output = 'Very Poor'
    elif output == 'Satisfactory':
        output = 'Satisfactory'
    elif output == 'Good':
        output = 'Good'
    elif output == 'Severe':
        output = 'Severe'

    return render_template('index.html', prediction_text='Air Pollution is : {}'.format(output))

if __name__ == "__main__":
    app.run(host="localhost", port=5000)
```

## APPENDIX 3 - SAMPLE SCREENS

### SAMPLE SCREENS

```
In [1]: # Import the libraries.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

In [2]: # Read the Dataset.
data = pd.read_csv('Dataset.csv')

In [3]: import warnings
warnings.filterwarnings('ignore')

In [4]: data.head(11)

Out[4]:
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	28-06-2010	3.800	6.084	3.508	3.840	3.840	246560000
1	05-07-2010	4.000	4.000	2.996	3.480	3.480	127753000
2	12-07-2010	3.590	4.300	3.380	4.128	4.128	77194500
3	19-07-2010	4.274	4.450	3.900	4.258	4.258	35878500
4	26-07-2010	4.300	4.300	3.910	3.988	3.988	15260000
5	02-08-2010	4.100	4.436	3.904	3.918	3.918	21998500
6	09-08-2010	3.960	3.996	3.478	3.664	3.664	21063000
7	16-08-2010	3.690	3.918	3.652	3.820	3.820	12050500
8	23-08-2010	3.818	4.078	3.712	3.940	3.940	15389500
9	30-08-2010	3.940	4.260	3.866	4.210	4.210	11752500
10	06-09-2010	4.122	4.210	3.952	4.034	4.034	6473000

```
In [5]: df = data.dropna()
```

#### *Scr 1 Reading Dataset*

```
In [5]: df = data.dropna()

In [6]: df.columns

Out[6]: Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')

In [7]: # Describe the data in statistical view.
df.describe()

Out[7]:
```

	Open	High	Low	Close	Adj Close	Volume
count	608.000000	608.000000	608.000000	608.000000	608.000000	6.080000e+02
mean	131.854122	140.089835	124.765849	132.822142	132.822142	1.510910e+08
std	241.085132	256.121627	227.391014	242.137560	242.137560	1.210761e+08
min	3.590000	3.918000	2.996000	3.480000	3.480000	6.473000e+05
25%	19.031000	20.275500	17.390000	19.518000	19.518000	6.656125e+07
50%	46.310000	48.312000	44.451000	46.396999	46.396999	1.322868e+08
75%	67.497503	70.415001	64.464002	67.118000	67.118000	1.992209e+08
max	1162.329956	1243.489990	1118.660034	1222.089966	1222.089966	1.066860e+09

```
In [8]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 608 entries, 0 to 607
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Date        608 non-null    object
1   Open        608 non-null    float64
2   High        608 non-null    float64
3   Low         608 non-null    float64
4   Close       608 non-null    float64
5   Adj Close   608 non-null    float64
6   Volume      608 non-null    int64
dtypes: float64(5), int64(1), object(1)
memory usage: 33.4+ KB
```

#### *Scr 2 Dataset information*

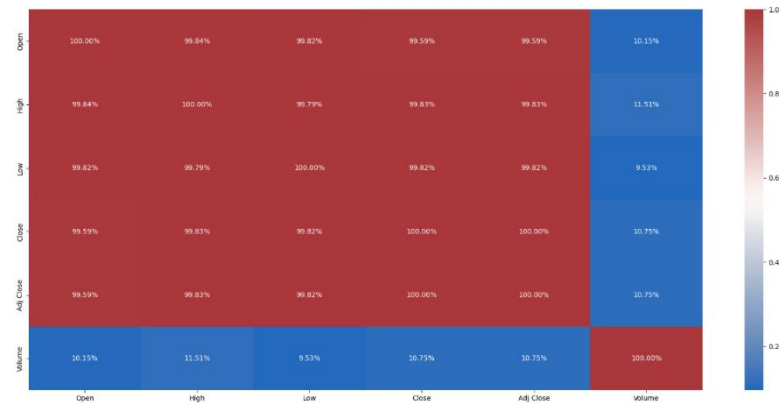
### Scr 3 Renaming column

#### Scr 4 Checking Null value in Dataset

### Scr 5 Shape and Size of the dataset

```
In [11]: # Display the Heatmap
fig, ax = plt.subplots(figsize=(22,10))
sns.heatmap(df.corr(),annot = True, fmt='0.2%', cmap = 'vlag',ax=ax)
#YlGnBu, Reds, tab10, rocket, mako, flare, crest, magma, viridis, cubehelix, YlOrBr, vlag, icefire, Spectral, coolwarm, Blues
```

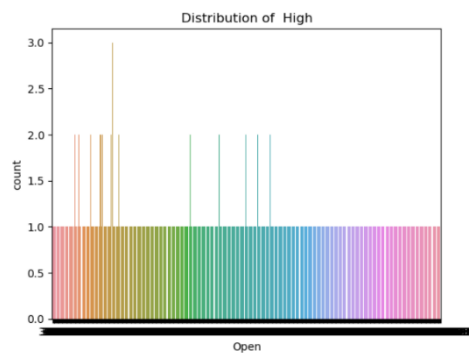
Out[11]: <AxesSubplot:>



*Scr 6 Heatmap*

```
In [12]: #plotting graph for distribution
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x = "Open", data = df)
df.loc[:, 'High'].value_counts()
plt.title('Distribution of High')
```

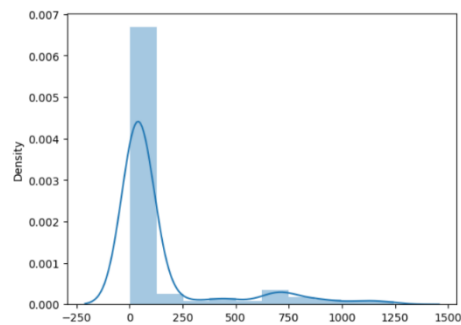
Out[12]: Text(0.5, 1.0, 'Distribution of High')



*Scr 7 Graph for distribution*

```
In [13]: # Display the distplot.
sns.distplot(x = df['High'], bins = 10)
```

Out[13]: <AxesSubplot:ylabel='Density'>

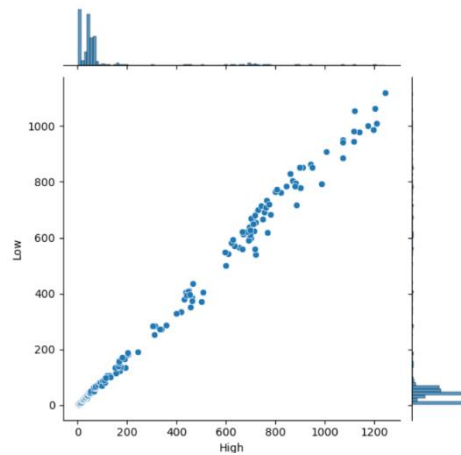


```
In [14]: # Display the Jointplot.
sns.jointplot(x = df['High'], y = df['Low'], kind = 'scatter')
```

Out[14]: <seaborn.axisgrid.JointGrid at 0x235929e8460>

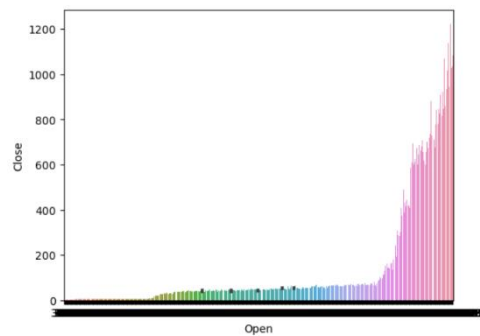
*Scr 8 Distplot*

```
In [14]: # Display the Jointplot.
sns.jointplot(x = df['High'], y = df['Low'], kind = 'scatter')
Out[14]: <seaborn.axisgrid.JointGrid at 0x235929e8460>
```

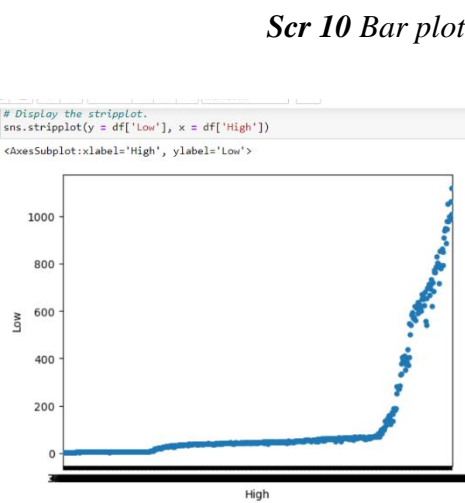


*Scr 9 Jointplot*

```
In [16]: # Display the barplot.
sns.barplot(x = df['Open'], y = df['Close'])
Out[16]: <AxesSubplot:xlabel='Open', ylabel='Close'>
```



```
In [17]: # Display the stripplot.
sns.stripplot(y = df['Low'], x = df['High'])
Out[17]: <AxesSubplot:xlabel='High', ylabel='Low'>
```



*Scr 11 Strip plot*

```
In [15]: from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, explained_variance_score, r2_score, mean_squared_error
from sklearn.metrics import median_absolute_error
```

Training

```
In [16]: SVR= AdaBoostRegressor()

SVR.fit(X_train,y_train)

predictD = SVR.predict(X_test)
```

Finding the mean absolute error

```
In [17]: MAE= (mean_absolute_error(y_test,predictD))
print("MEAN_ABSOLUTE_ERROR:", MAE)
print("")

MEAN_ABSOLUTE_ERROR: 19.571654473816686
```

Finding the mean absolute percentage error

```
In [18]: MAPE = (mean_absolute_percentage_error(y_test, predictD))
print("MEAN_ABSOLUTE_PERCENTAGE_ERROR:", MAPE)
print("")

MEAN_ABSOLUTE_PERCENTAGE_ERROR: 1.6679277734615692
```

Finding the Accuracy

```
In [19]: EVS = (explained_variance_score(y_test, predictD)*100)
print("ACCURACY RESULT OF ADABOOST REGRESSOR IS :", EVS)
```

## *Scr 12 Adaboost Regressor Training*

Finding the Accuracy

```
In [19]: EVS = (explained_variance_score(y_test, predictD)*100)
print("ACCURACY RESULT OF ADABOOST REGRESSOR IS :", EVS)
print("")

ACCURACY RESULT OF ADABOOST REGRESSOR IS : 99.074087755899
```

Finding the Mean squared error

```
In [20]: MSE = (mean_squared_error(y_test, predictD))
print("MEAN_SQUARED_ERROR:", MSE)
print("")

MEAN_SQUARED_ERROR: 611.394078453317
```

Finding the Median absolute error

```
In [21]: MEDIAN_S_E = (median_absolute_error(y_test, predictD))
print("MEDIAN_ABSOLUTE_ERROR:", MEDIAN_S_E)
print("")

MEDIAN_ABSOLUTE_ERROR: 19.171999375000006
```

Finding the R2 score

```
In [22]: R2 = (r2_score(y_test, predictD)*100)
print("The Accuracy of adaboost regressor:", R2)
print("")

The Accuracy of adaboost regressor: 99.06166312809103
```

## *Scr 13 Adaboost RegressorAccuracy*



```

In [15]: from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolute_error, mean_absolute_percentage_error, r2_score

Training

In [16]: RFT = Ridge()
RFT.fit(X_train, y_train)
predictR = RFT.predict(X_test)

Find The mean absolute error.

In [17]: MAE = (mean_absolute_error(y_test, predictR))
print("MEAN ABSOLUTE ERROR :", MAE)
print("")
MEAN ABSOLUTE ERROR : 4.000189629172735

Find the mean absolute percentage error.

In [18]: MAPE = (mean_absolute_percentage_error(y_test, predictR))
print("MEAN ABSOLUTE PERCENTAGE ERROR:", MAPE)
print("")
MEAN ABSOLUTE PERCENTAGE ERROR: 0.04862851623566244

In [19]: #Find the mean squared error.

In [20]: MSE = (mean_squared_error(y_test, predictR))
print("MEAN_SQUARED_ERROR:", MSE)

```

### *Scr 14 Ridge Regressor Training*

```

In [19]: #Find the mean squared error.

In [20]: MSE = (mean_squared_error(y_test, predictR))
print("MEAN_SQUARED_ERROR:", MSE)
print("")
MEAN_SQUARED_ERROR: 87.95006179505839

Find the Median absolute error.

In [21]: MEDIAN_AE = (median_absolute_error(y_test, predictR))
print("MEDIAN ABSOLUTE ERROR:", MEDIAN_AE)
print("")
MEDIAN ABSOLUTE ERROR: 0.951971487399959

Find the Accuracy of Ridge Regressor.

In [22]: EVS = (explained_variance_score(y_test, predictR)*100)
print("Explained variance score:", EVS)
print("")
Explained variance score: 99.82633338491216

Find the R2_score

In [23]: R2_SCORE = (r2_score(y_test, predictR)*100)
print("The Accuracy of Ridge regressor :", R2_SCORE)
print("")
The Accuracy of Ridge regressor : 99.82611262743984

```

### *Scr 15 Ridge Regressor Accuracy*

```

Find the Accuracy of Ridge Regressor

In [22]: EVS = (explained_variance_score(y_test, predictR)*100)
print("Explained variance score:", EVS)
print("")
Explained variance score: 99.82633338491216

Find the R2_score

In [23]: R2_SCORE = (r2_score(y_test, predictR)*100)
print("The Accuracy of Ridge regressor :", R2_SCORE)
print("")
The Accuracy of Ridge regressor : 99.82611262743984

In [24]: import joblib
joblib.dump(RFT, 'rft.pkl')

Out[24]: ['rft.pkl']

```

### *Scr 16 Ridge Regressor Accuracy*

```

In [14]: from sklearn.linear_model import Lasso
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_error, r2_score, explained_variance_score, median_absolute_error

Training

In [15]: LR = Lasso()
LR.fit(X_train,y_train)
predictLR = LR.predict(X_test)

Find the mean absolute error

In [16]: MAE = (mean_absolute_error(y_test,predictLR))
print("MEAN ABSOLUTE ERROR:", MAE)
print('')
MEAN ABSOLUTE ERROR: 5.156236207880183

Find the Mean squared error

In [17]: MSE = (mean_squared_error(y_test,predictLR))
print("MEAN SQUARED ERROR:", MSE)
print('')
MEAN SQUARED ERROR: 199.66178867088638

Find the Mean absolute error

In [18]: MAPE = (mean_absolute_percentage_error(y_test,predictLR))

```

## *Scr 17 LASSO Regressor Training*

```

Find the Mean absolute error

In [18]: MAPE = (mean_absolute_percentage_error(y_test,predictLR))
print("MEAN ABSOLUTE PERCENTAGE ERROR :", MAPE)
print('')
MEAN ABSOLUTE PERCENTAGE ERROR : 0.07621871222246727

Find the Median absolute error

In [19]: MEDIAN_SE = (median_absolute_error(y_test, predictLR))
print("MEDIAN ABSOLUTE ERROR:", MEDIAN_SE)
print('')
MEDIAN ABSOLUTE ERROR: 1.1917607384357067

Find the Accuracy of Lasso

In [20]: EVS = (explained_variance_score(y_test, predictLR)*100)
print("Explained variance score :", EVS)
print('')
Explained variance score : 99.75457930429607

Find the r2score

In [21]: R2_SCORE = (r2_score(y_test, predictLR)*100)
print("The Accuracy of Lasso:", R2_SCORE)
print('')
The Accuracy of Lasso: 99.74023051964012

```

## *Scr 18 LASSO Regressor Accuracy*

```
In [13]: from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_error, median_absolute_error, explained_variance_score

Training

In [14]: DT= DecisionTreeRegressor()
DT.fit(X_train,y_train)
predictD = DT.predict(X_test)

Finding Mean absolute error

In [15]: MAE= (mean_absolute_error(y_test,predictD))
print("MEAN_ABSOLUTE_ERROR:", MAE)
print("")
MEAN_ABSOLUTE_ERROR: 4.062514262295081

In [16]: MAPE = (mean_absolute_percentage_error(y_test,predictD))
print("MEAN_ABSOLUTE_PERCENTAGE_ERROR :", MAPE)
print("")
MEAN_ABSOLUTE_PERCENTAGE_ERROR : 0.04160659541589144

Find Mean squared error

In [17]: MSE=(mean_squared_error(y_test,predictD))
print("MEAN_SQUARED_ERROR_VALUE IS :",MSE)
print("")
MEAN_SQUARED_ERROR_VALUE IS : 92.38788935604
```

## *Scr 19 Decision Tree Regressor Training*

```
Find Median absolute error

In [18]: MedianAE=(median_absolute_error(y_test,predictD))
print("MEDIAN_ABSOLUTE_ERROR_VALUE IS :",MedianAE)
print("")
MEDIAN_ABSOLUTE_ERROR_VALUE IS : 1.0320019999999985

Find explained variance score

In [19]: EVS=(explained_variance_score(y_test,predictD)*100)
print("EXPLAINED_VARIANCE_SCORE:",EVS)
print("")
EXPLAINED_VARIANCE_SCORE: 99.83279017515059

Find Accuracy Score

In [20]: R2=(r2_score(y_test,predictD)*100)
print("Accuracy result of Decision Tree regressor is :",R2)
print("")
Accuracy result of Decision Tree regressor is : 99.83258573196214

In [21]: import joblib
joblib.dump(DT, 'Dt.pkl')
Out[21]: ['Dt.pkl']
```

## *Scr 20 Decision Tree Regressor Accuracy*

```
In [1]: import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
import joblib

app = Flask(__name__)
model = joblib.load('Dt.pkl')

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    """
    For rendering results on HTML GUI
    """
    int_features = [(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    print(final_features)
    prediction = model.predict(final_features)

    output = prediction[0]
    print(output)

    return render_template('index.html', prediction_text='The Closing price of tesla stock is:{}'.format(output))

if __name__ == "__main__":
    # app.run(host="localhost", port=5000)

if __name__ == '__main__':
    app.debug = True
    app.run(host="localhost", port=5000)
```

## *Scr 21 Flask code*

## REFERENCES

- [1] G. Li, A. Zhang, Q. Zhang, D. Wu and C. Zhan, "Pearson Correlation Coefficient-Based Performance Enhancement of Broad Learning System for Stock Price Prediction," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 5, pp. 2413-2417, May 2022, doi: 10.1109/TCSII.2022.3160266.
- [2] Zaznov, I.; Kunkel, J.; Dufour, A.; Badii, A. Predicting Stock Price Changes Based on the Limit Order Book: A Survey. *Mathematics* 2022, 10, 1234. <https://doi.org/10.3390/math10081234>
- [3] Dylan M. Crain, Stock Movement Prediction using Technical and Data, Stanford Department of Energy Resources Engineering 736 Serra St., Stanford CA, 94305
- [4] N. Naik and B. R. Mohan, "Novel Stock Crisis Prediction Technique—A Study on Indian Stock Market," in *IEEE Access*, vol. 9, pp. 86230-86242, 2021, doi: 10.1109/ACCESS.2021.3088999.
- [5] reference: P. Srivastava and P. K. Mishra, "Stock Market Prediction Using RNN LSTM," 2021 2nd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2021, pp. 1-5, doi: 10.1109/GCAT52182.2021.9587540.
- [6] M. Nabipour, P. Nayyeri, H. Jabani, S. S. and A. Mosavi, "Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data; a Comparative Analysis," in *IEEE Access*, vol. 8, pp. 150199-150212, 2020, doi: 10.1109/ACCESS.2020.3015966.
- [7] I. Bhattacharjee and P. Bhattacharja, "Stock Price Prediction: A Comparative Study between Traditional Statistical Approach and Machine Learning Approach," 2019 4th International Conference on Electrical Information and Communication Technology (EICT), Khulna, Bangladesh, 2019, pp. 1-6, doi: 10.1109/EICT48899.2019.9068850.
- [8] Selvamuthu, D., Kumar, V. & Mishra, A. Indian stock market prediction using artificial neural networks on tick data. *Financ Innov* 5, 16 (2019).doi: <https://doi.org/10.1186/s40854-019-0131-7>
- [9] A, Yashmita & D, Dr. (2023). Building a Stock Price Prediction Model using Random Forest Regression and Sentimental Analysis. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*. 07. 10.55041/IJSREM18258.
- [10] S. Sarode, H. G. Tolani, P. Kak and C. S. Lifna, "Stock Price Prediction Using Machine Learning Techniques," 2019 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 2019, pp. 177-181, doi: 10.1109/ISS1.2019.8907958.