A PROJECT REPORT

ON

“CALCULATED PREDICTIVE ANALYSIS MODEL USING XGBoost AND DECISION TREE”

SUBMITTED BY:
Shivanand Thakur
Master of Computer Application
Vellore Institute of Technology

Table of Contents

1. Introduction 1

1.1 Calculated Predictive Analysis Model an Overview 1

1.2 Objective and Scope of the Project 2

1.3 Literature Survey 3

2. SYSTEM ANALYSIS 7

2.1 Proposed system 7

2.1.1 Defining the Problem 7

Definition of Problem 7

2.1.2 Developing Solution Strategies 8

2.1.4 Data Flow Diagram 10

2.2 Dataset Descriptions 12

2.3 Coding Module (Modular Descriptions) 14

1. Importing Libraries 14
2. Importing Dataset 14
3. Counting the total number of Rows and Columns 15
4. Describing the Dataset 15
5. Information of the Data type in the Dataset 15
6. Exploratory Data Analysis 16
7. Head Description for similar data 16
8. Checking for the availability of null values 17
9. Creating SubPlots 17
10. Creating BoxPlot 18
11. Feature Engineering 18
12. Determining whether it is a Quarter end or not using Boolean values 19
13. Creating Bar graph 19
14. Calculating Quarter End 20
15. Creating Pie-Chart 20
16. Creating Heatmap 21
17. Data Splitting and Normalization 22
18. Model Development and Evaluation 22
19. Confusion Matrix 23
20. Calculated Confusion Matrix 24
21. Calculating the Root Mean Square and Accuracy of the Model 25

Conclusion 25
22. Comparison With Existing Models 26
23. RMS, MAE and ME Values of ARIMA & ETS Model 27
24. RMS, MAE & ME Values of Decision Tree Model 28
References 29

1. Introduction

1.1 Calculated Predictive Analysis Model an Overview

Stock Market prediction and analysis is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. Stock market is the important part of economy of the country and plays a vital role in the growth of the industry and commerce of the country that eventually affects the economy of the country. Both investors and industry are involved in stock market and wants to know whether some stock will rise or fall over certain period of time. The stock market is the primary source for any company to raise funds for business expansions. It is based on the concept of demand and supply. If the demand for a company's stock is higher, then the company share price increases and if the demand for company's stock is low then the company share price decrease.

The **Bombay Stock Exchange** (BSE) is the leading stock exchange of India, located in Mumbai. The BSE was established in 1986 as the first demutualized electronic exchange in the country. BSE became the first company to be popular among the people after NSE as it has lot more companies listed as compared to NSE. BSE has over 5,500 listed companies while NSE had 1,600 listed companies which make BSE more diversified in terms of the companies that can be

The **BSE** index is National Stock Exchange of India's benchmark broad based stock market index for the Indian equity market. It represents the weighted average of 500 Indian company stocks in 12 sectors and is one of the two main stock indices used in India, the other being the NSE, Sensex.

Due to involvement of many numbers of industries and companies, it contains very large sets of data from which it is difficult to extract information and analyse their trend of work manually. The application developed in this project, not only helps in prediction the future movement if the stock in the market, but also automate the data retrieval, trend analysis, predictive analysis and insights generation of a stock, just at the click of a button. Stock market analysis and prediction will reveal the market patterns and predict the time to purchase stock. The successful prediction of a stock's future price could yield significant profit. This is done using large historic market data of 12 months in this project, to represent varying conditions and confirming that the time series patterns have statistically significant predictive power[1].

1.2 Objective and Scope of the Project

Objective:

- **To add to the academic understanding of stock market prediction:**
  - With a greater understanding of how the market moves, investors will be better equipped to prevent another financial crisis.
  - Evaluate some existing strategies from a rigorous scientific perspective and provide a quantitative evaluation of new strategies.

- **Provide an automated Stock Prediction Tool to Traders to:**
  - Make Buy/Sell Decisions
  - Distinguish between conservative and risky stocks

Scope:

- No attempt is made in this project at portfolio management. Portfolio management is largely an extra step done after an investor has made a prediction on which direction any particular stock will move. The investor may choose to allocate funds across a range of stocks in such a way to minimize his or her risk. For instance, the investor may choose not to invest all of their funds into a single company lest that company takes unexpected turn. A more common approach would be for an investor to invest across a broad range of stocks based on some criteria he has decided on before. **This project will focus exclusively on predicting the daily trend (price movement) of individual stocks. The project will make no attempt to deciding how much money to allocate to each prediction. More so, the project will analyse the accuracies of these predictions**[2].
- A distinction must be made between the trading algorithms studied in this project and high frequency trading (HFT) algorithms. HFT algorithms make little use of intelligent prediction and instead rely on being the fastest algorithm in the market. These algorithms operate in fractions of a second. **The algorithms presented in this report will operate on the order of days and will attempt to be truly predictive of the market.**

1.3 Literature Survey

In the last few decades forecasting of stock returns has become an important field of research. In most of the cases the researchers had attempted to establish a linear relationship between the input macroeconomic variables and the stock returns. After the discovery of nonlinearity in the stock market index returns, many literatures have come up in nonlinear statistical modelling of the stock returns, most of them required that the nonlinear model be specified before the estimation is done. But since stock market return is noisy, uncertain, chaotic and nonlinear in nature, Predictive Modelling & Machine Learning has evolved in capturing the structural relationship between a stock's performance and its determinant factors more accurately than many other statistical techniques.

In literature, different sets of input variables are used to predict stock returns. In fact, different input variables are used to predict the same set of stock return data. Some researchers used input data from a single time series where others considered the inclusion of heterogeneous market information and macroeconomic variables. Some researchers even pre-processed these input data sets before feeding it to the Predictive Model for forecasting.

Min and Lee were doing prediction of bankruptcy using machine learning. They evaluated methods based on Support Vector Machine, multiple discriminant analysis, logistic regression analysis, and three-layer fully connected back-propagation neural networks. Their results indicated that support vector machines outperformed other approaches. A Decision Tree is a useful and popular classification technique that inductively learns a model from a given set of data. One reason for its popularity stems from the availability of existing algorithms that can be used to build decision trees, such as CART [3], ID3 [4], and C4.5 [5]. These algorithms all learn decision trees from a supplied set of training data, but do so in slightly different ways. As discussed in the introduction, a classifier is built by analysing training data. That is to say, a classifier is built by analysing a collection of instances where each instance is composed of a set of attribute values and a known class value. These decision tree algorithms build top-down structures that partition instances into separate classes, and it is hoped that these structures generalize well to instances with unknown class values. This would mean that the decision trees have fulfilled their objectives and have indeed discovered some underlying property of the data [4].

Tsai and Wang did research where they tried to predict stock prices by using ensemble learning, composed of decision trees and artificial neural networks. They created dataset from Taiwanese stock market data, taking into account fundamental indexes, technical indexes, and macroeconomic indexes. The performance of Decision Tree + Artificial Neural Network trained on Taiwan stock exchange data showed Score performance of 77%. Single algorithms showed F-score performance up to 67%.

Josip Arneric, Elza Jurun, Snjezana pivac describes that technical analysis is done to find out the price movements whereas fundamental analysis is done to predict values by looking at the fundamentals of a particular company. They focused on technical analysis and define that trend can be of two types on the basis of either time structure or general direction.

Professor Veroljub says that the way of investing is to sell when prices are at top and to buy when prices are at lower whatever the patterns are. In his articles he has discussed the market efficiency theory, Classical theory, confidence theory and Dow Theory. He also differentiates between the Classical and Confidence theory

Wing-Keung Wong, Meher Manzur and Boon-Kiat Chew (2002) article discuss that the helpful principle of technical analysis is to identify trends and then go with the trend whether it is occurring randomly or due to fundamental factors. He also discussed the techniques of moving averages and relative strength index (RSI) by applying it on Singapore stock exchanges.

There are many tools and software available out there that provide forecasting of stock market entities, share quantity and share value for a given financial organization. Most of them claim to predict the stock market with near to 100% accuracy but the opinions from the users vary. Some of the popular tools and software with their methodologies are mentioned as follows.

- **inteliCharts Predictive Stock Market Analytics:**
  It is a quantitative modelling tool used for financial time series forecasting. The system is adaptive in its core as it learns the patterns and geometrical relationships defined by historical time series data points, which are unique for each individual stock, index, or another financial instrument.
- **Markettrak**
  Its stock market forecast system consists of two major parts: an extensive database and a forecast model. The forecast model reads the database and then makes a prediction of where the market is headed. From this prediction, it determines a trading position for the Dow Diamonds or the SP500 Spiders. The database and forecast are updated daily at the close of trading.

- **Stock-Forecasting.com**

  www.stock- forecasting.com (Centre of Mathematics & Science, Inc., Chicago, United States of America) provides innovative price-prediction technology for active Day Traders, Short- and Long-term Investors. They develop web-based software for stock market forecasting and analysis [2].

The Stock-Forecasting software predicts stock prices, generates trading "Buy-Hold-Sell" signals, computes the most profitable company to invest in and analyses the accuracy of predictions.

The researcher conducted a study of intemal representation and a purchasing and selling time prediction system for Tokyo Stock Exchange stocks are covered. Modular neural networks are the foundation of it. For the TOPIX (Tokyo Stock Exchange Prices Indexes) prediction system, we created a number of learning algorithms and prediction techniques. The simulation on stock trading produced outstanding profits, and the prediction system produced precise predictions. Fujitsu and Nikko Securities created the forecasting system[6].

The author of the paper says that intelligent systems may be used to anticipate the stock market. The use of hybridised soft computing approaches for automated trend analysis and stock market forecasting is the topic of this article. We employ a neural network for stock forecasting one day in advance and a neuro-fuzzy system for trend analysis of the anticipated stock values[7]. We took into consideration the well-known Nasdaq-100 index of Nasdaq Stock MarketSM to illustrate the suggested methodology. We looked at 24 months' worth of stock data for the Nasdaq-100 main index as well as six of the Nasdaq100 index businesses. Principal component analysis was used to preprocess the input data before feeding it to an artificial neural network for stock forecasting. A neuro-fuzzy system is further given the expected stock values.

The researcher says that The ways low- and medium-technology (LMT) companies use to create technical innovation are discussed in this research, which also looks at how these strategies affect the innovation performance of the companies. Based on a sample from the Taiwanese Technological Innovation Survey, these analyses a total of 753 LMT companies. According to the descriptive data, 95% of businesses got their technology through technology licencing, whereas 32% of businesses outsourced their R&D. About 20%, 18%, 8%, and 23% of the companies in the sample collaborate with suppliers, clients, rivals, and research organisations to acquire external technological expertise. The results of this study, which use a moderated hierarchical regression analysis, are intriguing. First, the performance of innovation is not considerably impacted by internal technology licencing. Second, internal R&D spending has a detrimental moderating impact [8].

The Researcher says The genetic quantum algorithm (GQA) is a brand-new evolutionary computing technique that is proposed in this research. The foundation of GQA is the idea and fundamentals of quantum computing, including qubits and superposition of states. Due to the qubit chromosome's probabilistic representation, GQA may express a linear superposition of solutions in place of binary, numeric, or symbolic representation. Quantum gates are used to seek for the optimal answer as genetic operators. The performance of GQA is characterised by quick convergence and strong global search capabilities. Experimental findings on the knapsack problem, a well-known combinatorial optimisation issue, show the efficacy and applicability of GQA. The findings demonstrate that GQA outperforms alternative genetic algorithms that make use of penalty functions, repair techniques and decoder [9].

Given how complicated and volatile the stock market is, experts have long been interested in predicting price movements. stock market intrinsic turbulence throughout the globe makes making predictions difficult. Even if they are useful, forecasting and diffusion modelling cannot be the answer to all of the issues associated with prediction, whether it be long-term or short-term. To ensure low investment risk, market risk, which is closely connected with forecasting mistakes, needs to be reduced. By framing the forecasting problem as a classification problem, which is addressed by a common set of machine learning methods, the authors suggest reducing forecasting error. In this study, we provide a unique approach for reducing the risk of stock market investment through stock return forecasting [10].

The field of software engineering was created in response to the perceived software crisis in the industry. A well, that is. It is a well-known truth that the software business needs to know how much a project will cost to produce and how much time it will take. Compared to resource estimate in other industries, software engineering presents the greatest challenge. The neural network model is one of the resource estimating techniques that are currently accessible. This study suggests comparing several neural network model training techniques and demonstrates which is most appropriate for software engineering applications [11].

2. SYSTEM ANALYSIS

2.1 Proposed system

2.1.1 Defining the Problem

The Stock Market is a complex and dynamical system, & is influenced by many factors that are subject to uncertainty. So, it is a difficult task to forecast stock price movements. Due to technology and globalization of business & financial markets it is important to predict the stock prices more quickly & accurately. Automated User-friendly Trading application can be developed based on financial predictive indicator algorithms & machine learning techniques to predict the performance of stocks in NSE's BSE Index.

Definition of Problem

The Stock Market is a complex and dynamical system, & is influenced by many factors that are subject to uncertainty. So, it is a difficult task to forecast stock price movements. Due to technology and globalization of business & financial markets it is important to predict the stock prices more quickly & accurately. Automated User-friendly Trading application can be developed based on financial predictive indicator algorithms & machine learning techniques to predict the performance of stocks in BSE Index.

Investors prefer stock market investments as they have the opportunity of highest return over other schemes. Nifty (benchmark of NSE India) is a well-diversified index consisting of 50 major stocks from 21 sectors of the Indian economy. However, trading through Stock buy/sell prediction computer algorithms is still in its nascent stage in the Indian stock market. The need of an automated user-friendly trading predictor system, which predicts stock price upward/downward movements, is necessary in the Indian stock market, given the explosion of algorithmic trading, being one of the most prominent trends in the global financial industry over recent decade. A Stock Prediction Application will be developed in this project using Nifty data, keeping in mind the following three steps:

1. The system has to have some models generating Stock Market predictions using **Financial Stock Predictor** Functions **(E.g.: Williams %R)** and **Machine Learning** Techniques **(E.g.: Decision Trees)**[12].
2. **Back testing** of the **Prediction Models** is essential to evaluate the trading system's performance on historical market data and thus determine the viability of the system
3. An analytical insight has to be provided, of whether the stock is: **"Bullish"** or **"Bearish".**

2.1.2 Developing Solution Strategies

Developing a stock market prediction system using machine learning involves several steps and strategies. Here's a general outline of the process:

Data Collection: Gather historical stock market data, including price, volume, and other relevant indicators. You can obtain this data from various financial data providers or APIs.

Data Preprocessing: Clean and preprocess the collected data to handle missing values, outliers, and inconsistencies. Perform tasks like data normalization, feature scaling, and handling categorical variables.

Feature Selection: Identify the most relevant features that have a significant impact on stock price movements. This can be done through techniques like correlation analysis, feature importance, or domain expertise.

Feature Engineering: Create additional features that can capture meaningful patterns or trends in the data. For example, you can calculate moving averages, technical indicators (e.g., RSI, MACD), or derive features from news sentiment analysis.

Model Selection: Choose an appropriate machine learning algorithm for your prediction task. Commonly used algorithms for stock market prediction include linear regression, decision trees, random forests, support vector machines (SVM), or more advanced techniques like deep learning models (e.g., recurrent neural networks or long short-term memory networks).

Model Training: Split the pre-processed data into training and testing sets. Train the selected model using the training data while adjusting the model's hyperparameters. Cross-validation techniques like k-fold cross-validation can be used to optimize the model's performance and prevent overfitting.

Model Evaluation: Assess the performance of your trained model using appropriate evaluation metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or accuracy metrics depending on the nature of the problem (regression or classification).

Model Refinement: Fine-tune your model by iterating on the previous steps. Experiment with different algorithms, hyperparameters, or feature combinations to improve the predictive performance. Consider techniques like ensemble learning (combining multiple models) to enhance accuracy.

Back testing and Validation: Apply your trained model to a separate validation dataset or historical data to evaluate its performance in real-world scenarios. Back testing can help simulate how the model would have performed in the past.

Deployment and Monitoring: Once you have a well-performing model, deploy it to a production environment. Continuously monitor its performance and update the model periodically to adapt to changing market conditions and incorporate new data.

It's important to note that stock market prediction is a highly complex and volatile domain, and accurate predictions are challenging to achieve consistently. Financial markets are influenced by various factors, including economic indicators, political events, investor sentiment, and more. Therefore, it's crucial to set realistic expectations and consider the limitations of any prediction system.

2.1.4 Data Flow Diagram

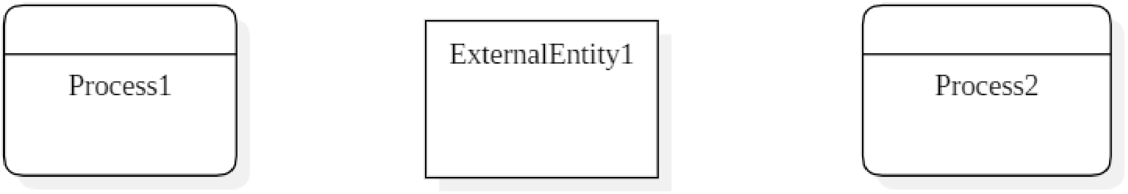
A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various sub processes the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships.

Data flow diagram levels:

Data flow diagrams are also categorized by level. Starting with the most basic, level 0, DFDs get increasingly complex as the level increases.

Context Diagrams/Level 0 DFDs are the most basic data flow diagrams. They provide a broad view that is easily digestible but offers little detail.

Context Diagram:



Level -0 DFD:

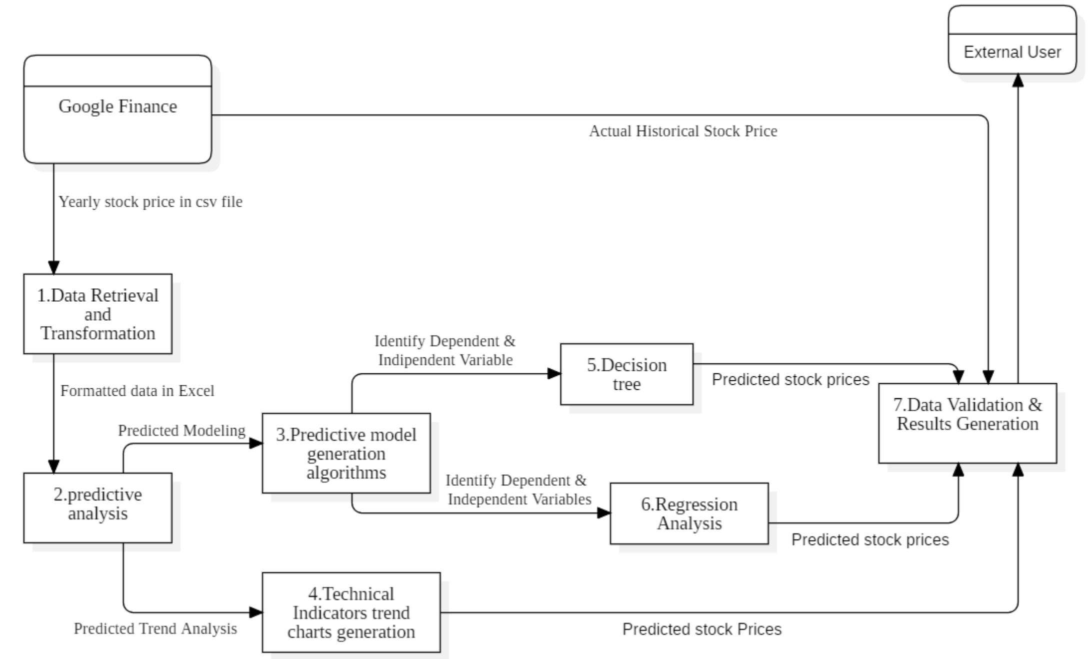

External User
Google Finance
Actual Historical Stock Price
Yearly stock price in csv file
1.Data Retrieval and Transformation
Identify Dependent & Indipendent Variable
5.Decision tree
Predicted stock prices
7.Data Validation & Results Generation
Formatted data in Excel
Predicted Modeling
3.Predictive model generation algorithms
Identify Dependent & Independent Variables
6.Regression Analysis
2.predictive analysis
Predicted stock prices
4.Technical Indicators trend charts generation
Predicted Trend Analysis
Predicted stock Prices

Fig 2: Level 0- Basic Data Flow of the System.

Level 1 DFDs go into more detail than a Level 0 DFD. In a level 1 data flow diagram, the single process node from the context diagram is broken down into sub processes. As these processes are added, the diagram will need additional data flows and data stores to link them together.

Level – 1 DFD:

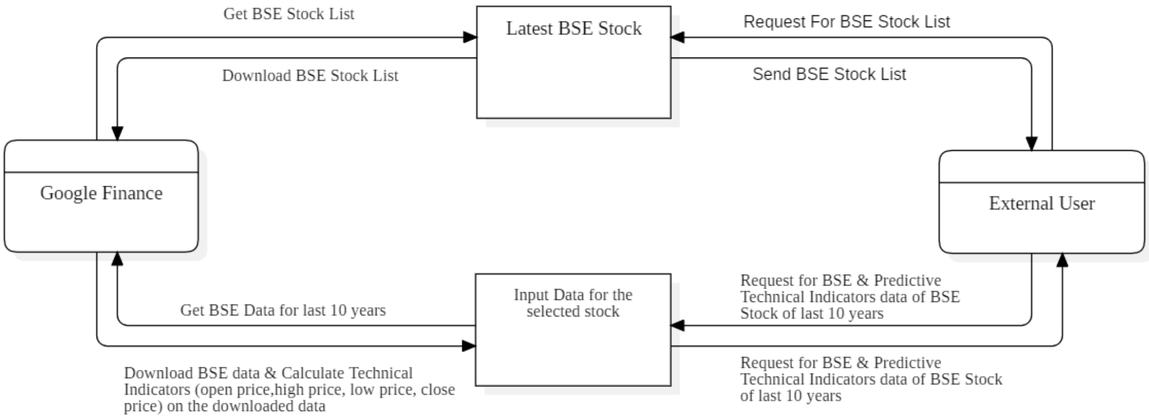

Get BSE Stock List
Latest BSE Stock
Request For BSE Stock List
Download BSE Stock List
Send BSE Stock List
Google Finance
External User
Input Data for the selected stock
Request for BSE & Predictive Technical Indicators data of BSE Stock of last 10 years
Get BSE Data for last 10 years
Download BSE data & Calculate Technical Indicators (open price,high price, low price, close price) on the downloaded data
Request for BSE & Predictive Technical Indicators data of BSE Stock of last 10 years

Fig 3: Level 1- Data Retrieval & Transformation

2.2 Dataset Descriptions

Dataset Description: Calculated Predictive Analysis Model

Dataset Name: BSE Dataset

Dataset Overview: Calculated Predictive Analysis Model Dataset is a collection of data gathered for the purpose of analysing and understanding various aspects related to fine the Stock price of the Companies listed in the BSE Index. It contains a diverse range of data points, including the Date on which the Stock Price movement has taken place, High_price, Low_price, opening_price, Closing_price, No. of share, No. of trades, Total Turn Over, etc,

Data Sources: The dataset is sourced from BSE India (Bombay Stock Exchange) which is the oldest and one of the major stock exchanges in India. It is located in Mumbai and serves as a platform for trading equities, derivatives, debt instruments, mutual funds, and other financial products. The sources include the data of the listed companies on the basis of their daily trades, Monthly trades and yearly trades. It also has the information about the company's Equity, Mutual Funds, Debt and others.

Data Format and Structure: The dataset is provided in CSV file format. It is organized into Rows and Columns. Each record within the dataset represents a distinct information about the date on which the trade has taken place, the opening and the closing price of the trade. It also provides the information about the High_price and Low_price that the stock showed on the specific date. The respective data set of the company has the information about the total number of shares used in the trade that day and even the total number of shares involved in the process.

Variables and Features: The dataset comprises several variables or features that capture relevant information for the project. Some of the key variables includes total turnover of the company, Deliverable Quantity, % Deli. Qty to Traded Qty, Spread High-Low, Spread Close-Open

Data Preprocessing: The dataset has undergone certain preprocessing steps to ensure its quality and usability. This includes removal of all the null and the black data. The dataset is processed in such a way that on the days when the market was closed the data is not left blank at those days (data has be removed of those dates), the dates has been mentioned in a way that the software can read it easily to find the quarterly data of the companies. The data is present in the form of rows and columns therefore it is easy to find the total number of data available in our dataset.

Data Size: The dataset consists of 100gb of records.

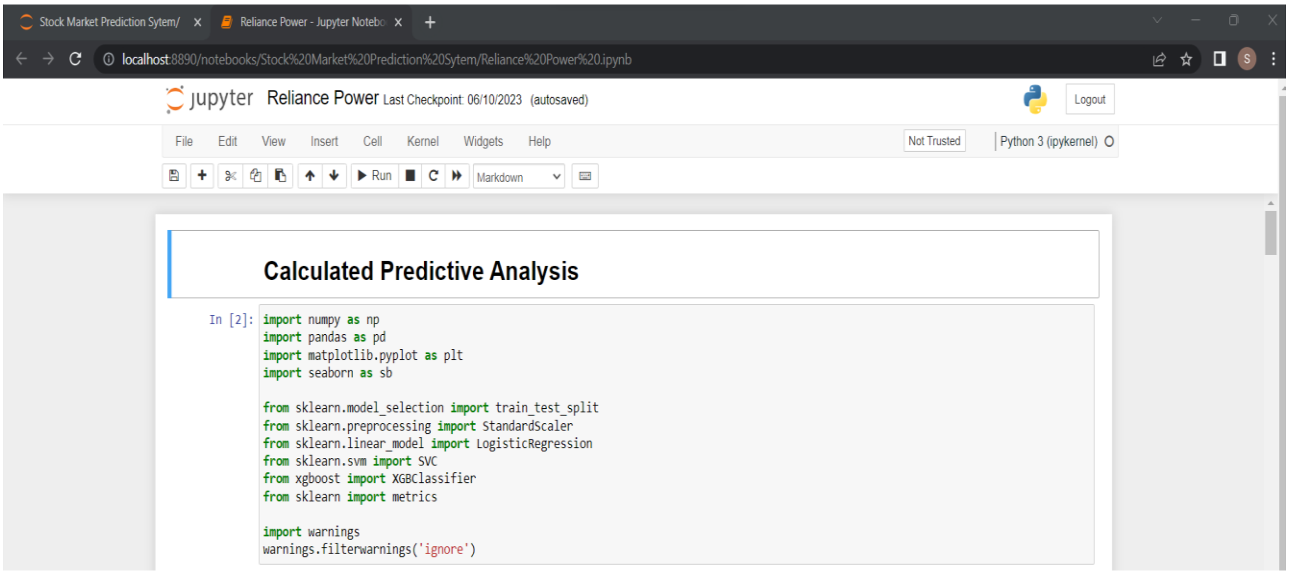
Data Usage and License: The Project Calculated Predictive Analysis Model Dataset is intended for research and analysis purposes related to the specific project. It is important to note that the dataset is subject to BSE stock price.

Related Metadata: Alongside the dataset, supplementary metadata is provided to aid in understanding the data. This may include data dictionaries, codebooks, or additional documentation that explains the variables, data collection process, or any transformations applied.

Data Availability: The dataset is available BSE Stock Price. Instructions for accessing or obtaining the dataset can be found at:[21]

2.3 Coding Module (Modular Descriptions)

1. Importing Libraries


Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter
Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

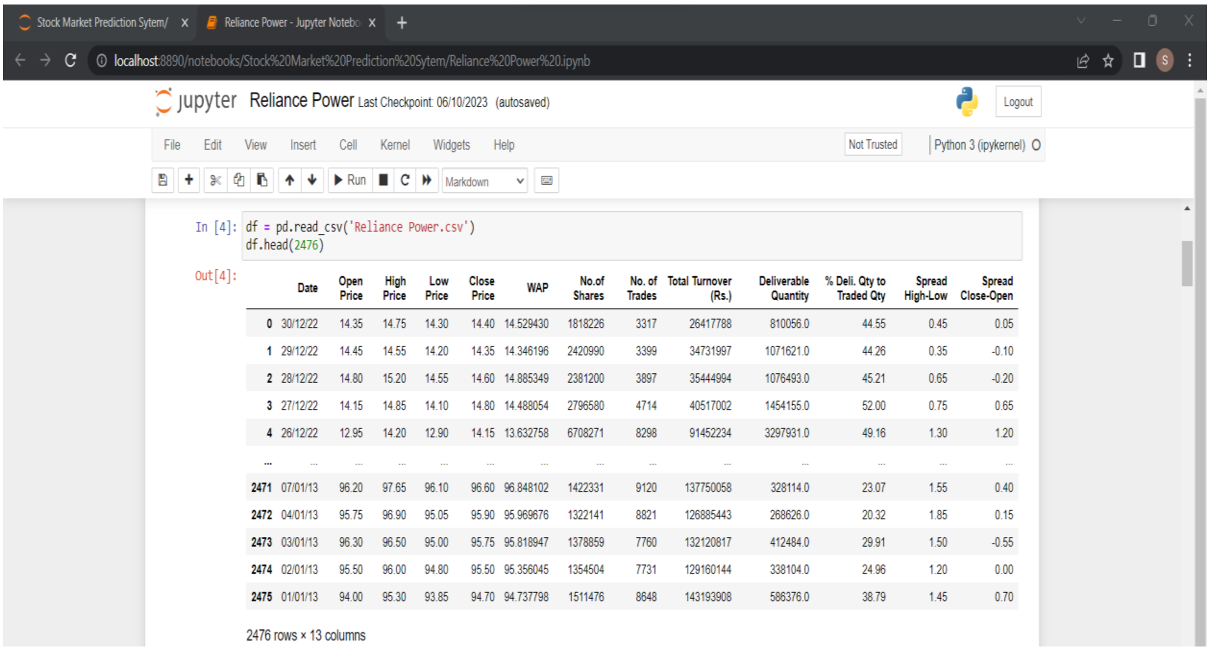
Calculated Predictive Analysis

```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sb

        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
        from sklearn.linear_model import LogisticRegression
        from sklearn.svm import SVC
        from xgboost import XGBClassifier
        from sklearn import metrics

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

Fig 8: Importing Libraries

2. Importing Dataset

The dataset we will use here to perform the analysis and build a predictive model is Reliance Stock Price data. We will use OHLC (‘Open’, ‘High’, ‘Low’, ‘Close’) data from 1st January 2013 to 31st December 2022 which is for 9 years for the Reliance Power.


Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter
Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [4]: df = pd.read_csv('Reliance Power.csv')
        df.head(2476)
```

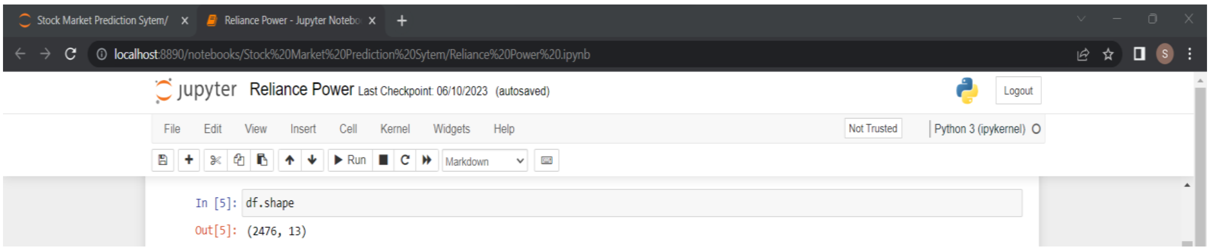
Out[4]:

	Date	Open Price	High Price	Low Price	Close Price	WAP	No.of Shares	No. of Trades	Total Turnover (Rs.)	Deliverable Quantity	% Deli. Qty to Traded Qty	Spread High-Low	Spread Close-Open
0	30/12/22	14.35	14.75	14.30	14.40	14.529430	1818226	3317	26417788	810056.0	44.55	0.45	0.05
1	29/12/22	14.45	14.55	14.20	14.35	14.346196	2420990	3399	34731997	1071621.0	44.26	0.35	-0.10
2	28/12/22	14.80	15.20	14.55	14.60	14.885349	2381200	3897	35444994	1076493.0	45.21	0.65	-0.20
3	27/12/22	14.15	14.85	14.10	14.80	14.488054	2796580	4714	40517002	1454155.0	52.00	0.75	0.65
4	26/12/22	12.95	14.20	12.90	14.15	13.632758	6708271	8298	91452234	3297931.0	49.16	1.30	1.20
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2471	07/01/13	96.20	97.65	96.10	96.60	96.848102	1422331	9120	137750058	328114.0	23.07	1.55	0.40
2472	04/01/13	95.75	96.90	95.05	95.90	95.969676	1322141	8821	126885443	268626.0	20.32	1.85	0.15
2473	03/01/13	96.30	96.50	95.00	95.75	95.818947	1378859	7760	132120817	412484.0	29.91	1.50	-0.55
2474	02/01/13	95.50	96.00	94.80	95.50	95.356045	1354504	7731	129160144	338104.0	24.96	1.20	0.00
2475	01/01/13	94.00	95.30	93.85	94.70	94.737798	1511476	8648	143193908	586376.0	38.79	1.45	0.70

2476 rows × 13 columns

Fig 9: Reliance Stock Data

3. Counting the total number of Rows and Columns

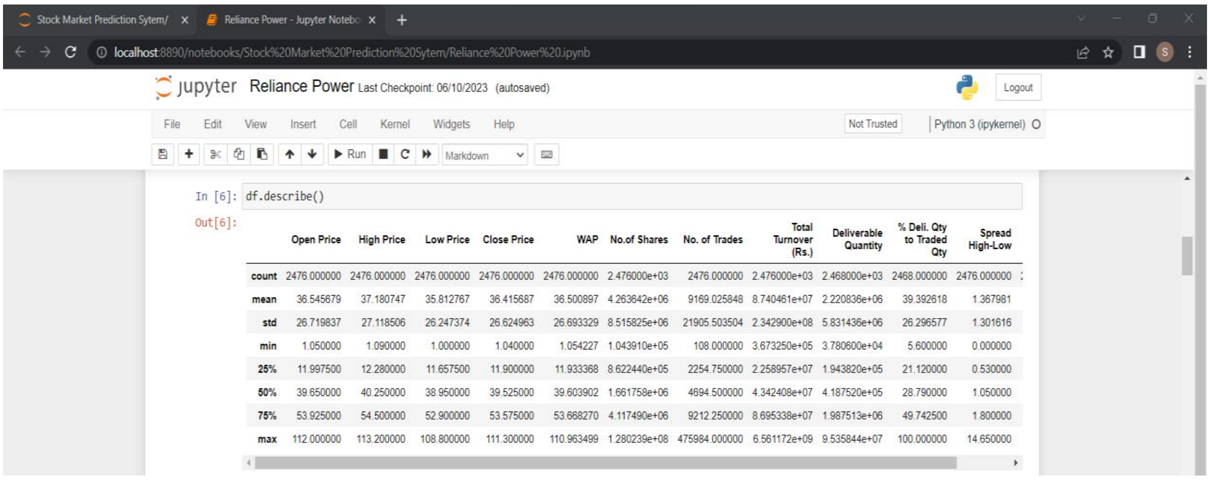

Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [5]: df.shape
Out[5]: (2476, 13)
```

Fig 10: Total Numbers of Rows and columns

4. Describing the Dataset

Describing the data on the basis of count value, mean value, std value, min value, 25% of the data, 50% of the data, 75% of the data and max value.


Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [6]: df.describe()
```

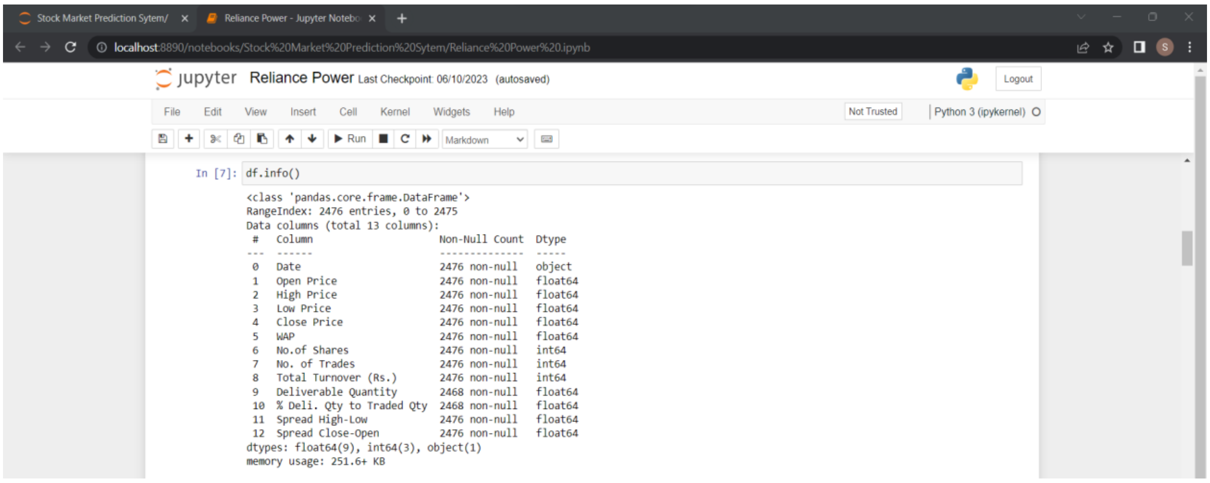
Out[6]:

	Open Price	High Price	Low Price	Close Price	WAP	No.of Shares	No. of Trades	Total Turnover (Rs.)	Deliverable Quantity	% Deli. Qty to Traded Qty	Spread High-Low
count	2476.000000	2476.000000	2476.000000	2476.000000	2476.000000	2.476000e+03	2476.000000	2.476000e+03	2.468000e+03	2468.000000	2476.000000
mean	36.545679	37.180747	35.812767	36.415687	36.500897	4.263642e+06	9169.025848	8.740461e+07	2.220836e+06	39.392618	1.367981
std	26.719837	27.118506	26.247374	26.624963	26.693329	8.515825e+06	21905.503504	2.342900e+08	5.831436e+06	26.296577	1.301616
min	1.050000	1.090000	1.000000	1.040000	1.054227	1.043910e+05	108.000000	3.673250e+05	3.780600e+04	5.600000	0.000000
25%	11.997500	12.280000	11.657500	11.900000	11.933368	8.622440e+05	2254.750000	2.258957e+07	1.943820e+05	21.120000	0.530000
50%	39.650000	40.250000	38.950000	39.525000	39.603902	1.661758e+06	4694.500000	4.342408e+07	4.187520e+05	28.790000	1.050000
75%	53.925000	54.500000	52.900000	53.575000	53.668270	4.117490e+06	9212.250000	8.695338e+07	1.987513e+06	49.742500	1.800000
max	112.000000	113.200000	108.800000	111.300000	110.963499	1.280239e+08	475984.000000	6.561172e+09	9.535844e+07	100.000000	14.650000

Fig 11: Describing the Data

5. Information of the Data type in the Dataset

Defining the data type of the available dataset.


Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

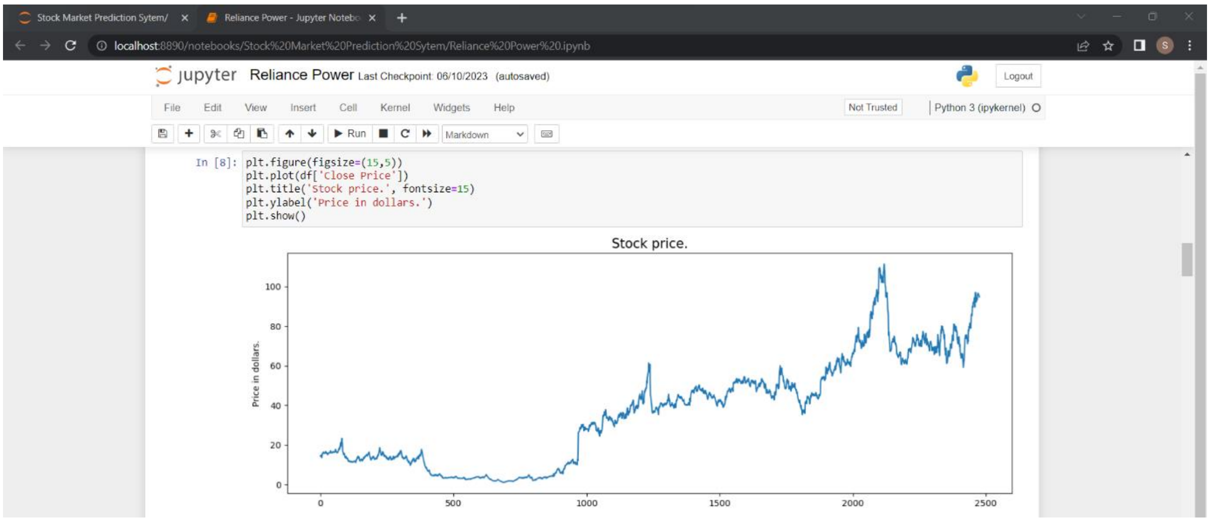
```
In [7]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2476 entries, 0 to 2475
Data columns (total 13 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Date                       2476 non-null   object 
 1   Open Price                 2476 non-null   float64
 2   High Price                 2476 non-null   float64
 3   Low Price                  2476 non-null   float64
 4   Close Price                2476 non-null   float64
 5   WAP                        2476 non-null   float64
 6   No.of Shares               2476 non-null   int64  
 7   No. of Trades              2476 non-null   int64  
 8   Total Turnover (Rs.)       2476 non-null   int64  
 9   Deliverable Quantity       2468 non-null   float64
 10  % Deli. Qty to Traded Qty  2468 non-null   float64
 11  Spread High-Low            2476 non-null   float64
 12  Spread Close-Open          2476 non-null   float64
dtypes: float64(9), int64(3), object(1)
memory usage: 251.6+ KB
```

Fig 12: Data Type Discription

6. Exploratory Data Analysis

EDA is an approach to analysing the data using visual techniques. It is used to discover trends, and patterns, or to check assumptions with the help of statistical summaries and graphical representations.

While performing the EDA of the Reliance Stock Price data we will analyse how prices of the stock have moved over the period of time and how the end of the quarters affects the prices of the stock.

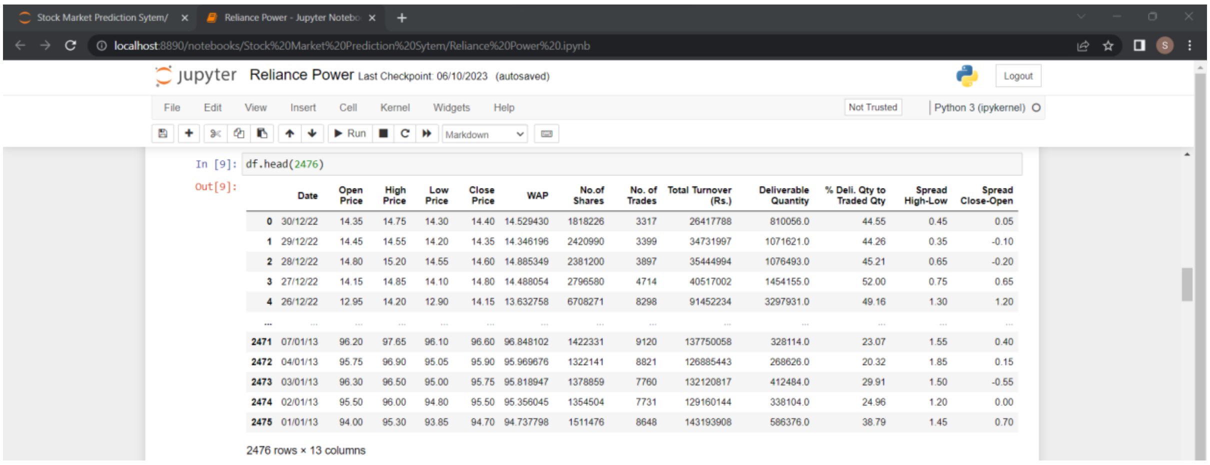

Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter
Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
Stock price.
Price in dollars.
0
20
40
60
80
100
0
500
1000
1500
2000
2500

```
In [8]: plt.figure(figsize=(15,5))
        plt.plot(df['Close Price'])
        plt.title('Stock price.', fontsize=15)
        plt.ylabel('Price in dollars.')
        plt.show()
```

Fig 13: Trend Analysis

The prices of Reliance Power stocks are showing a mix trend of upward trend and downward trend as depicted by the plot of the closing price of the stocks.

7. Head Description for similar data


Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter
Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [9]: df.head(2476)
Out[9]:
```

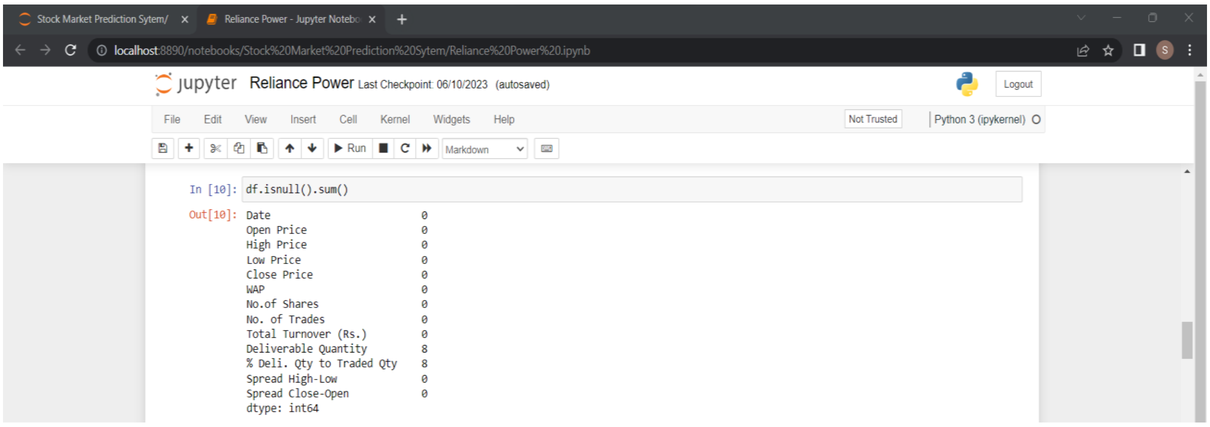
	Date	Open Price	High Price	Low Price	Close Price	WAP	No.of Shares	No. of Trades	Total Turnover (Rs.)	Deliverable Quantity	% Deli. Qty to Traded Qty	Spread High-Low	Spread Close-Open
0	30/12/22	14.35	14.75	14.30	14.40	14.529430	1818226	3317	26417788	810056.0	44.55	0.45	0.05
1	29/12/22	14.45	14.55	14.20	14.35	14.346196	2420990	3399	34731997	1071621.0	44.26	0.35	-0.10
2	28/12/22	14.80	15.20	14.55	14.60	14.885349	2381200	3897	35444994	1076493.0	45.21	0.65	-0.20
3	27/12/22	14.15	14.85	14.10	14.80	14.488054	2796580	4714	40517002	1454155.0	52.00	0.75	0.65
4	26/12/22	12.95	14.20	12.90	14.15	13.632758	6708271	8298	91452234	3297931.0	49.16	1.30	1.20
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2471	07/01/13	96.20	97.65	96.10	96.60	96.848102	1422331	9120	137750058	328114.0	23.07	1.55	0.40
2472	04/01/13	95.75	96.90	95.05	95.90	95.969676	1322141	8821	126885443	268626.0	20.32	1.85	0.15
2473	03/01/13	96.30	96.50	95.00	95.75	95.818947	1378859	7760	132120817	412484.0	29.91	1.50	-0.55
2474	02/01/13	95.50	96.00	94.80	95.50	95.356045	1354504	7731	129160144	338104.0	24.96	1.20	0.00
2475	01/01/13	94.00	95.30	93.85	94.70	94.737798	1511476	8648	143193908	586376.0	38.79	1.45	0.70

2476 rows × 13 columns

Fig 14: Determining identical data

If we observe carefully, we can see that there is no similar data that we have generated after our calculation.

8. Checking for the availability of null values

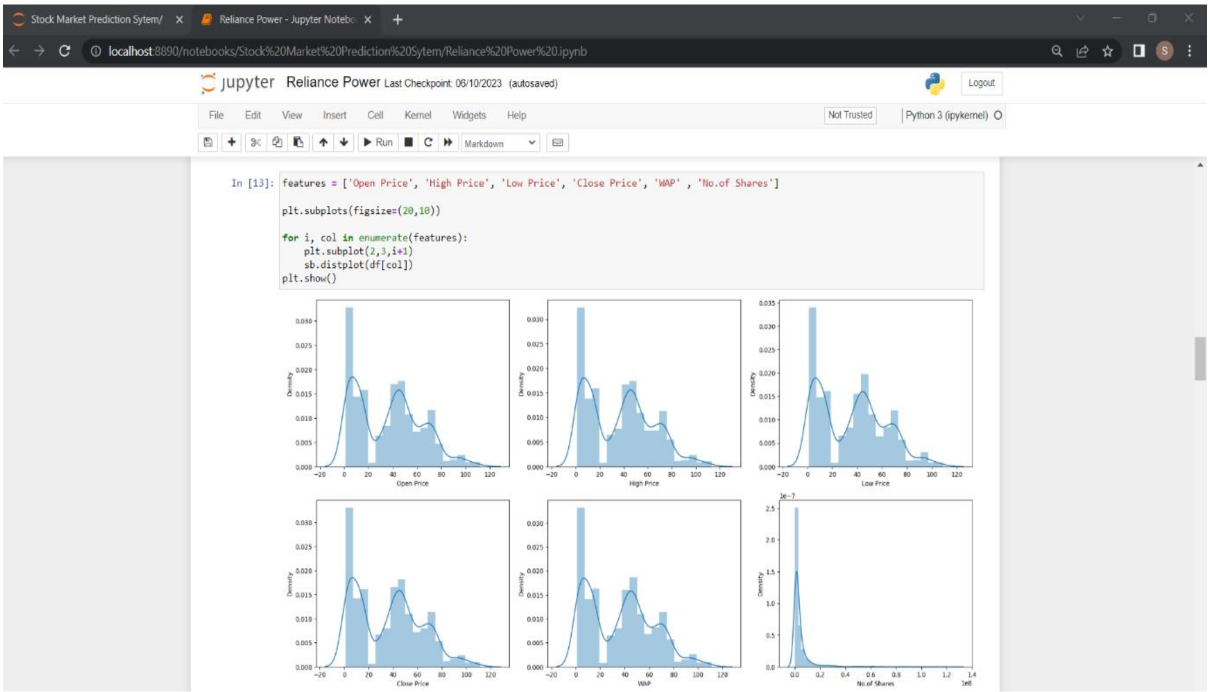

Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [10]: df.isnull().sum()

Out[10]: Date                         0
         Open Price                   0
         High Price                   0
         Low Price                    0
         Close Price                  0
         WAP                          0
         No.of Shares                 0
         No. of Trades                0
         Total Turnover (Rs.)         0
         Deliverable Quantity         8
         % Deli. Qty to Traded Qty    8
         Spread High-Low              0
         Spread Close-Open            0
         dtype: int64
```

Fig 15: Confirming Null values

This implies that there are no null values in the data set provided.

9. Creating SubPlots

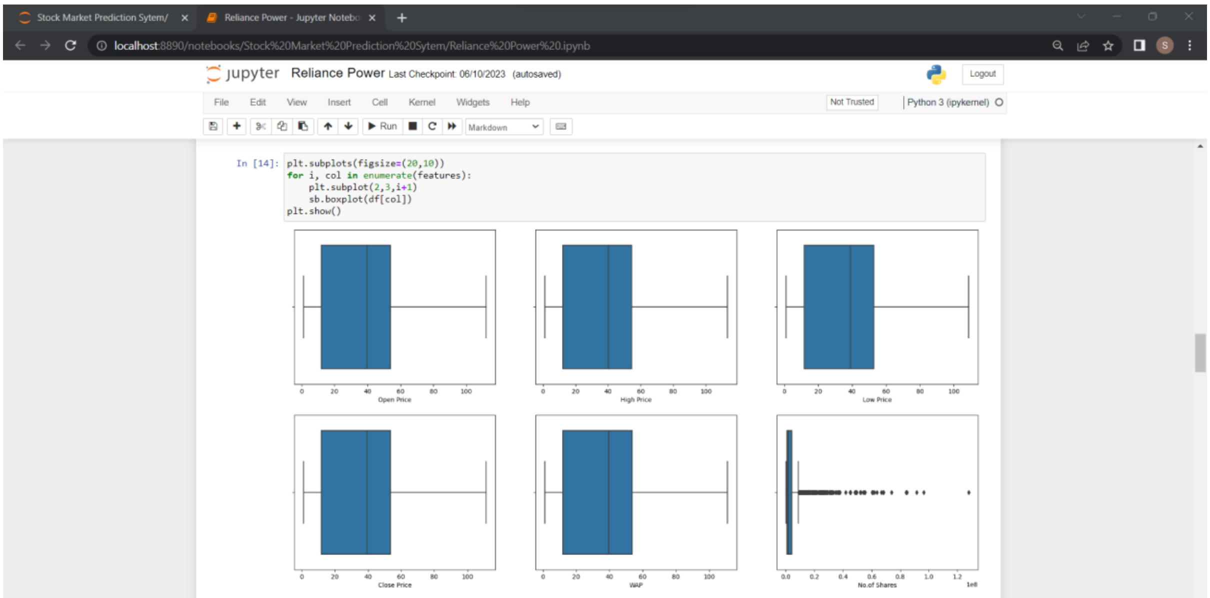

Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
Density
Open Price
High Price
Low Price
Close Price
WAP
No.of Shares
1e-7
1e8

```
In [13]: features = ['Open Price', 'High Price', 'Low Price', 'Close Price', 'WAP' , 'No.of Shares']

         plt.subplots(figsize=(20,10))

         for i, col in enumerate(features):
             plt.subplot(2,3,i+1)
             sb.distplot(df[col])
         plt.show()
```

Fig 16: Subplot for the Open, High, Low, Close, WAP, No.shares

In the distribution plot of OHLC data, we can see two peaks which means the data has varied significantly in two regions. And the No. of shares data is left-skewed.

10. Creating BoxPlot


Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
0 20 40 60 80 100
Open Price
High Price
Low Price
Close Price
WAP
0.0 0.2 0.4 0.6 0.8 1.0 1.2
No.of Shares
1e8

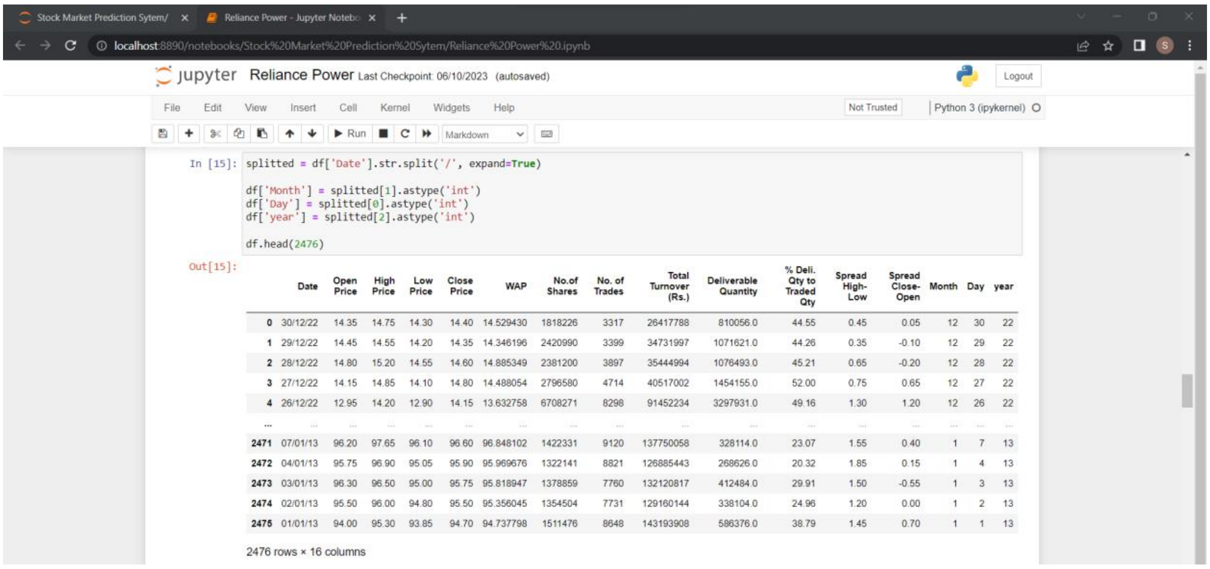
```
In [14]: plt.subplots(figsize=(20,10))
         for i, col in enumerate(features):
             plt.subplot(2,3,i+1)
             sb.boxplot(df[col])
         plt.show()
```

Fig 17: BoxPlot for the Open, High, Low, Close, WAP, No.shares

From the above boxplots, we can conclude that only volume data contains outliers in it but the data in the rest of the columns are free from any outlier.

11. Feature Engineering

Feature Engineering helps to derive some valuable features from the existing ones. These extra features sometimes help in increasing the performance of the model significantly and certainly help to gain deeper insights into the data.


Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [15]: splitted = df['Date'].str.split('/', expand=True)

         df['Month'] = splitted[1].astype('int')
         df['Day'] = splitted[0].astype('int')
         df['year'] = splitted[2].astype('int')

         df.head(2476)
```

Out[15]:

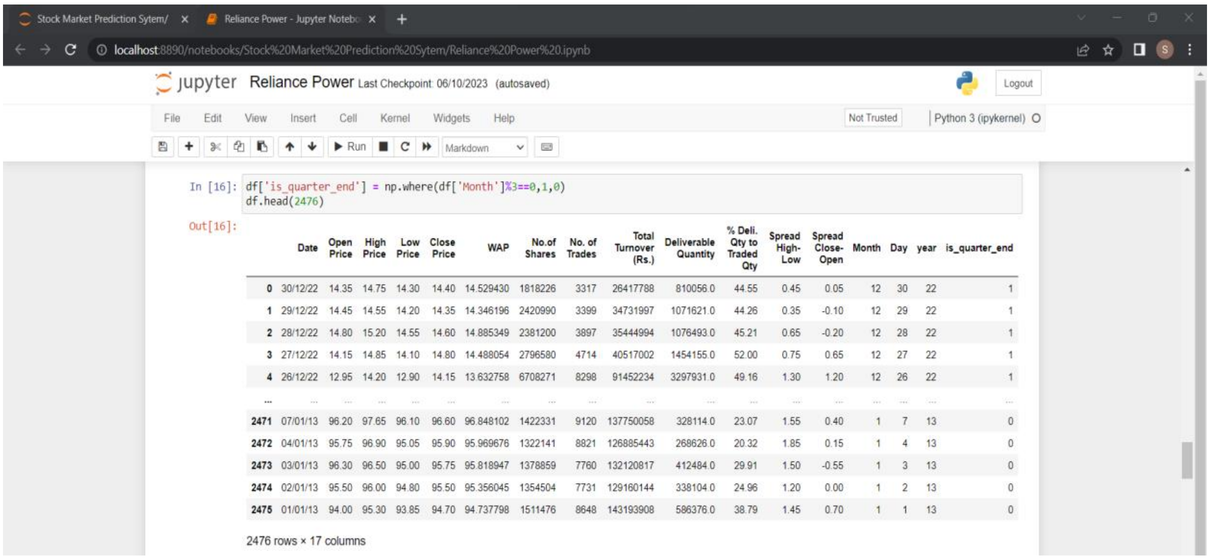
	Date	Open Price	High Price	Low Price	Close Price	WAP	No.of Shares	No. of Trades	Total Turnover (Rs.)	Deliverable Quantity	% Deli. Qty to Traded Qty	Spread High-Low	Spread Close-Open	Month	Day	year
0	30/12/22	14.35	14.75	14.30	14.40	14.529430	1818226	3317	26417788	810056.0	44.55	0.45	0.05	12	30	22
1	29/12/22	14.45	14.55	14.20	14.35	14.346196	2420990	3399	34731997	1071621.0	44.26	0.35	-0.10	12	29	22
2	28/12/22	14.80	15.20	14.55	14.60	14.885349	2381200	3897	35444994	1076493.0	45.21	0.65	-0.20	12	28	22
3	27/12/22	14.15	14.85	14.10	14.80	14.488054	2796580	4714	40517002	1454155.0	52.00	0.75	0.65	12	27	22
4	26/12/22	12.95	14.20	12.90	14.15	13.632758	6708271	8298	91452234	3297931.0	49.16	1.30	1.20	12	26	22
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2471	07/01/13	96.20	97.65	96.10	96.60	96.848102	1422331	9120	137750058	328114.0	23.07	1.55	0.40	1	7	13
2472	04/01/13	95.75	96.90	95.05	95.90	95.969676	1322141	8821	126885443	268626.0	20.32	1.85	0.15	1	4	13
2473	03/01/13	96.30	96.50	95.00	95.75	95.818947	1378859	7760	132120817	412484.0	29.91	1.50	-0.55	1	3	13
2474	02/01/13	95.50	96.00	94.80	95.50	95.356045	1354504	7731	129160144	338104.0	24.96	1.20	0.00	1	2	13
2475	01/01/13	94.00	95.30	93.85	94.70	94.737798	1511476	8648	143193908	586376.0	38.79	1.45	0.70	1	1	13

2476 rows × 16 columns

Fig 18: Deriving the data of Month, Day, Year

Now we have three more columns namely ‘day’, ‘month’ and ‘year’ all these three have been derived from the ‘Date’ column which was initially provided in the data.

12. Determining whether it is a Quarter end or not using Boolean values


Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter
Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [16]: df['is_quarter_end'] = np.where(df['Month']%3==0,1,0)
         df.head(2476)
```

Out[16]:

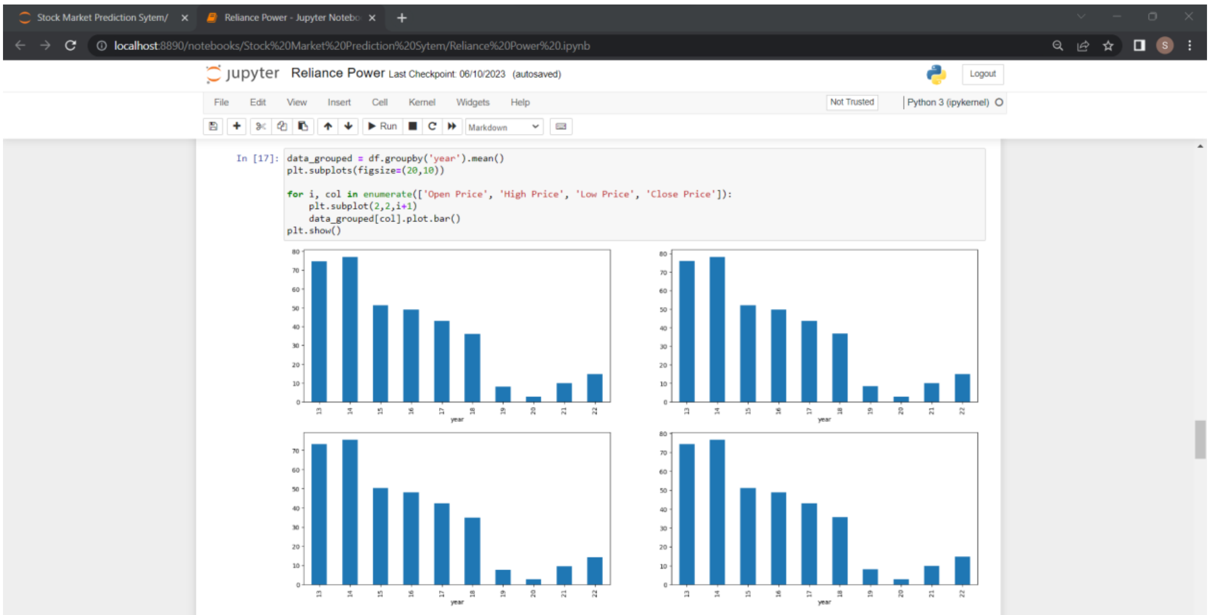
	Date	Open Price	High Price	Low Price	Close Price	WAP	No.of Shares	No. of Trades	Total Turnover (Rs.)	Deliverable Quantity	% Deli. Qty to Traded Qty	Spread High-Low	Spread Close-Open	Month	Day	year	is_quarter_end
0	30/12/22	14.35	14.75	14.30	14.40	14.529430	1818226	3317	26417788	810056.0	44.55	0.45	0.05	12	30	22	1
1	29/12/22	14.45	14.55	14.20	14.35	14.346196	2420990	3399	34731997	1071621.0	44.26	0.35	-0.10	12	29	22	1
2	28/12/22	14.80	15.20	14.55	14.60	14.885349	2381200	3897	35444994	1076493.0	45.21	0.65	-0.20	12	28	22	1
3	27/12/22	14.15	14.85	14.10	14.80	14.488054	2796580	4714	40517002	1454155.0	52.00	0.75	0.65	12	27	22	1
4	26/12/22	12.95	14.20	12.90	14.15	13.632758	6708271	8298	91452234	3297931.0	49.16	1.30	1.20	12	26	22	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2471	07/01/13	96.20	97.65	96.10	96.60	96.848102	1422331	9120	137750058	328114.0	23.07	1.55	0.40	1	7	13	0
2472	04/01/13	95.75	96.90	95.05	95.90	95.969676	1322141	8821	126885443	268626.0	20.32	1.85	0.15	1	4	13	0
2473	03/01/13	96.30	96.50	95.00	95.75	95.818947	1378859	7760	132120817	412484.0	29.91	1.50	-0.55	1	3	13	0
2474	02/01/13	95.50	96.00	94.80	95.50	95.356045	1354504	7731	129160144	338104.0	24.96	1.20	0.00	1	2	13	0
2475	01/01/13	94.00	95.30	93.85	94.70	94.737798	1511476	8648	143193908	586376.0	38.79	1.45	0.70	1	1	13	0

2476 rows × 17 columns

Fig 19: Determining Quarter End

A quarter is defined as a group of three months. Every company prepares its quarterly results and publishes them publicly so, that people can analyse the company’s performance. These quarterly results affect the stock prices heavily which is why we have added this feature because this can be a helpful feature for the learning model.

13. Creating Bar graph

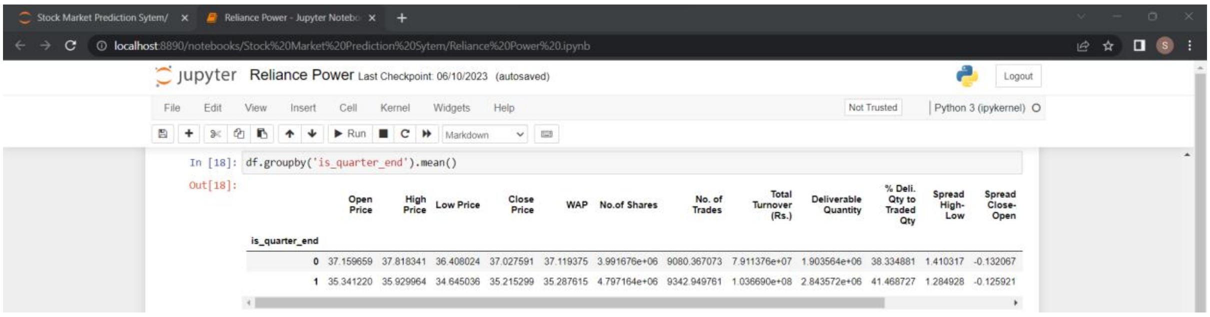

Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter
Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
80
70
60
50
40
30
20
10
0
13
14
15
16
17
18
19
20
21
22
year

```
In [17]: data_grouped = df.groupby('year').mean()
         plt.subplots(figsize=(20,10))

         for i, col in enumerate(['Open Price', 'High Price', 'Low Price', 'Close Price']):
             plt.subplot(2,2,i+1)
             data_grouped[col].plot.bar()
         plt.show()
```

Fig 20: Bar Chart for the Reliance Data

From the above bar graph, we can conclude that the stock prices increased by 50% from year 2021 to that in 2022.

14. Calculating Quarter End


Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [18]: df.groupby('is_quarter_end').mean()
Out[18]:
```

is_quarter_end	Open Price	High Price	Low Price	Close Price	WAP	No.of Shares	No. of Trades	Total Turnover (Rs.)	Deliverable Quantity	% Deli. Qty to Traded Qty	Spread High-Low	Spread Close-Open
0	37.159659	37.818341	36.408024	37.027591	37.119375	3.991676e+06	9080.367073	7.911376e+07	1.903564e+06	38.334881	1.410317	-0.132067
1	35.341220	35.929964	34.645036	35.215299	35.287615	4.797164e+06	9342.949761	1.036690e+08	2.843572e+06	41.468727	1.284928	-0.125921

Fig 21: Price Differences of Quarter End

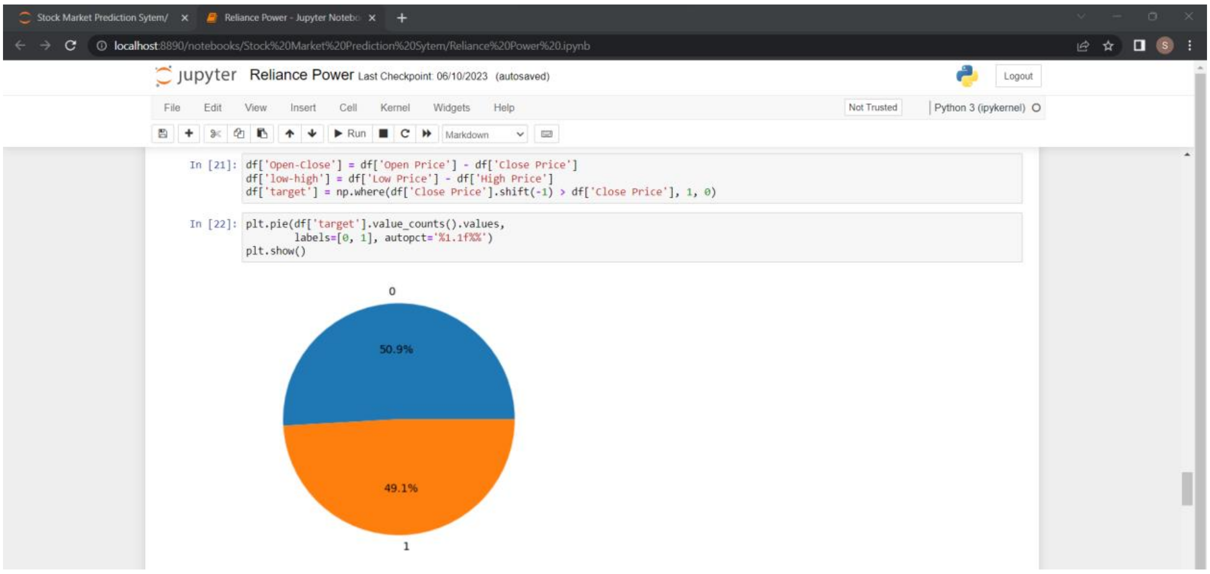
Here are some of the important observations of the above-grouped data:

Prices are higher in the months which are quarter end as compared to that of the non-quarter end months.

The WAP of trades is lower in the months which are quarter end.

15. Creating Pie-Chart

Above we have added some more columns which will help in the training of our model. We have added the target feature which is a signal whether to buy or not we will train our model to predict this only. But before proceeding let's check whether the target is balanced or not using a pie chart.

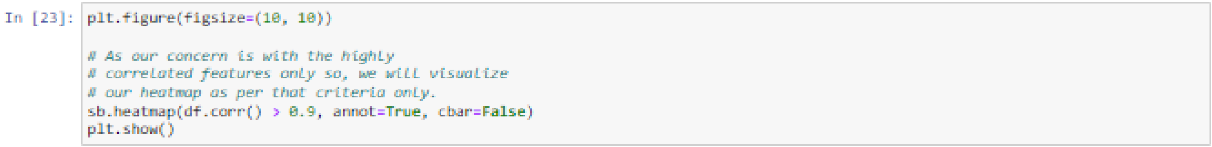

Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
0
50.9%
49.1%
1

```
In [21]: df['Open-Close'] = df['Open Price'] - df['Close Price']
         df['low-high'] = df['Low Price'] - df['High Price']
         df['target'] = np.where(df['Close Price'].shift(-1) > df['Close Price'], 1, 0)

In [22]: plt.pie(df['target'].value_counts().values,
                labels=[0, 1], autopct='%1.1f%%')
         plt.show()
```

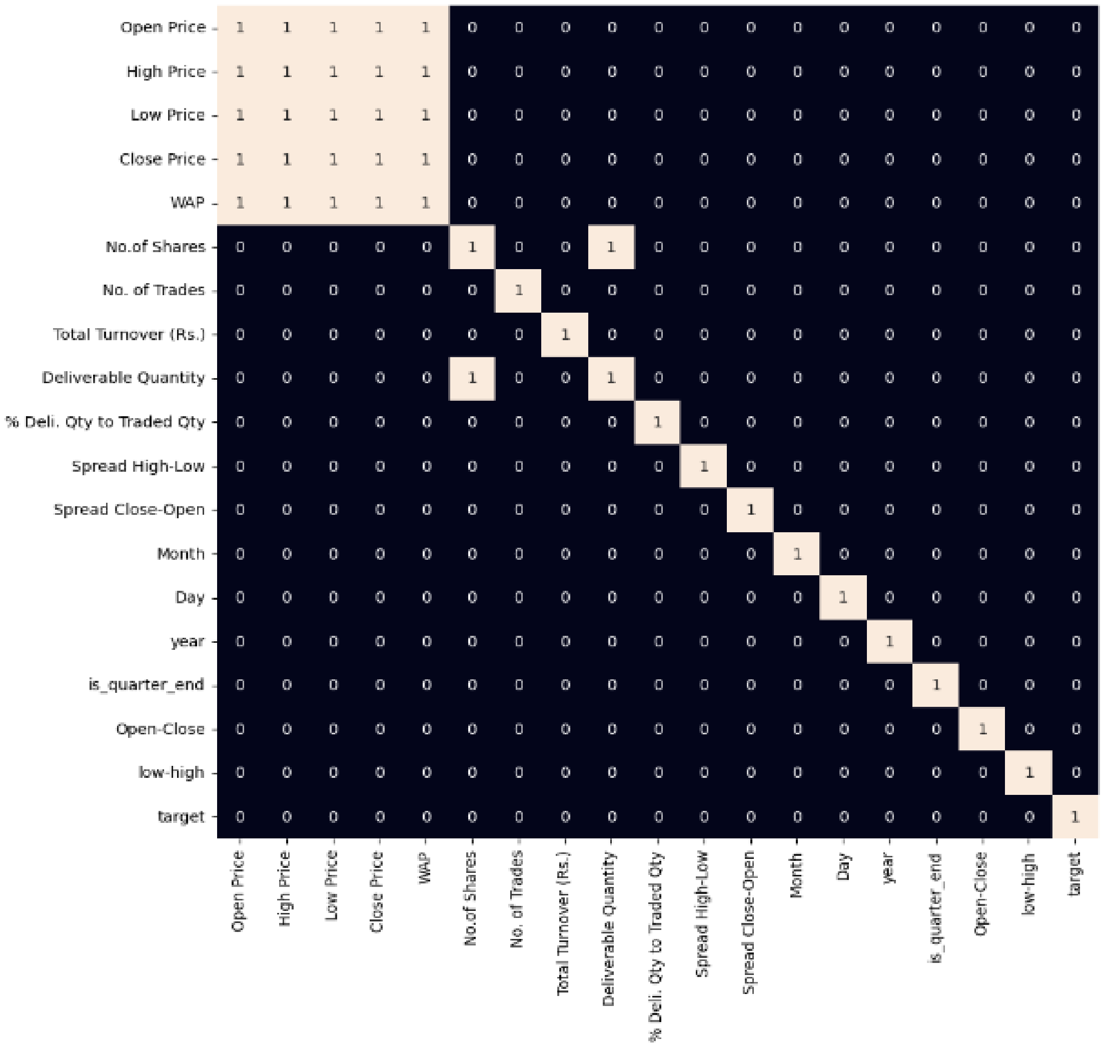
Fig 22: Creating Pie Chart for the Reliance Data

When we add features to our dataset, we have to ensure that there are no highly correlated features as they do not help in the learning process of the algorithm.

16. Creating Heatmap



```
In [23]: plt.figure(figsize=(10, 10))

         # As our concern is with the highly
         # correlated features only so, we will visualize
         # our heatmap as per that criteria only.
         sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)
         plt.show()
```

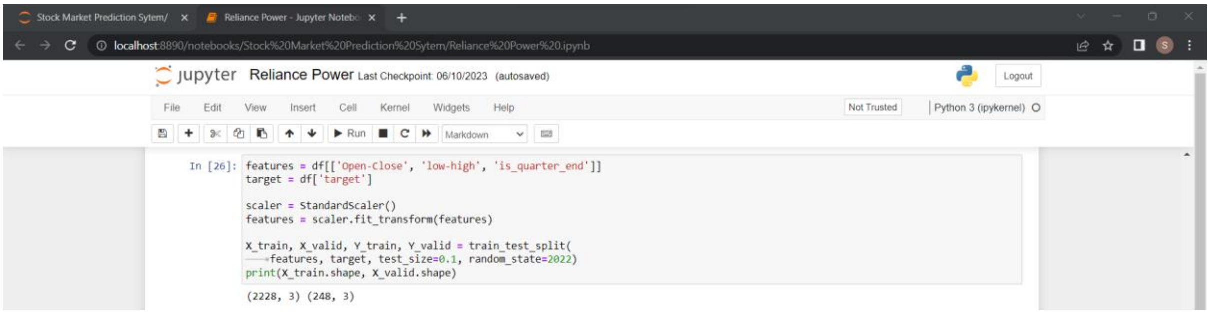


	Open Price	High Price	Low Price	Close Price	WAP	No.of Shares	No. of Trades	Total Turnover (Rs.)	Deliverable Quantity	% Deli. Qty to Traded Qty	Spread High-Low	Spread Close-Open	Month	Day	year	is_quarter_end	Open-Close	low-high	target
Open Price	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
High Price	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Low Price	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Close Price	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WAP	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
No.of Shares	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
No. of Trades	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Total Turnover (Rs.)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Deliverable Quantity	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
% Deli. Qty to Traded Qty	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Spread High-Low	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Spread Close-Open	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Month	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Day	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
year	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
is_quarter_end	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Open-Close	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
low-high	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
target	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Fig 23: Heat map determining various values

From the above heatmap, we can say that there is a high correlation between OHLC that is pretty obvious, and the added features are not highly correlated with each other or previously provided features which means that we are good to go and build our model.

17. Data Splitting and Normalization


Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter
Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [26]: features = df[['Open-Close', 'low-high', 'is_quarter_end']]
         target = df['target']

         scaler = StandardScaler()
         features = scaler.fit_transform(features)

         X_train, X_valid, Y_train, Y_valid = train_test_split(
             features, target, test_size=0.1, random_state=2022)
         print(X_train.shape, X_valid.shape)

         (2228, 3) (248, 3)
```

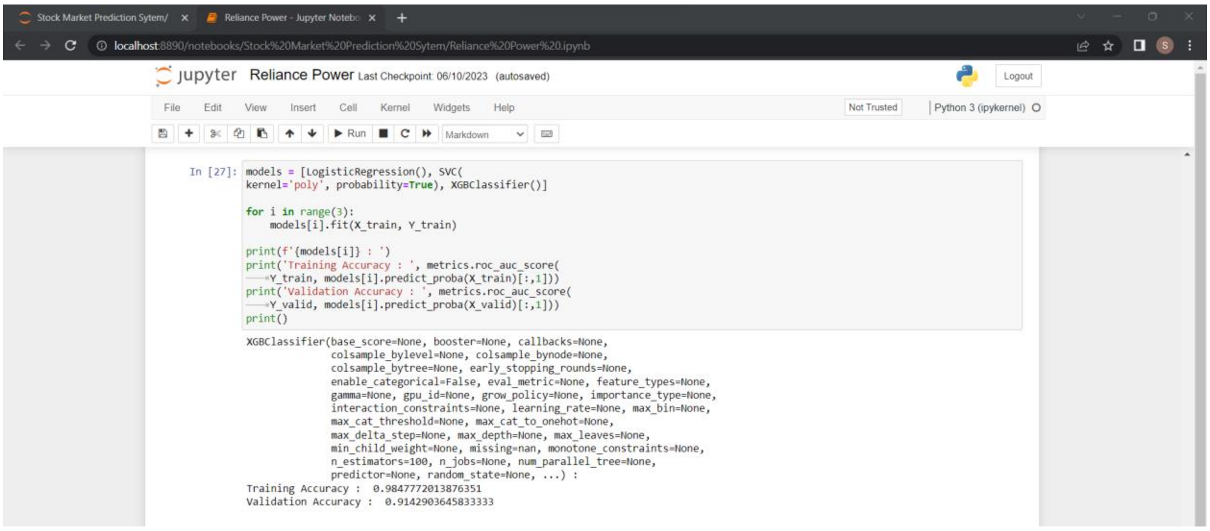
Fig 24: Data Splitting and Normalization

After selecting the features to train the model on we should normalize the data because normalized data leads to stable and fast training of the model. After that whole data has been split into two parts with a 90/10 ratio so, that we can evaluate the performance of our model on unseen data.

18. Model Development and Evaluation

Now is the time to train some state-of-the-art machine learning models (Logistic Regression, Support Vector Machine, XGBClassifier), and then based on their performance on the training and validation data we will choose which ML model is serving the purpose at hand better.

For the evaluation metric, we will use the ROC-AUC curve but why this is because instead of predicting the hard probability that is 0 or 1, we would like it to predict soft probabilities that are continuous values between 0 to 1. And with soft probabilities, the ROC-AUC curve is generally used to measure the accuracy of the predictions.


Stock Market Prediction Sytem/
Reliance Power - Jupyter Notebo
localhost:8890/notebooks/Stock%20Market%20Prediction%20Sytem/Reliance%20Power%20.ipynb
jupyter
Reliance Power Last Checkpoint: 06/10/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [27]: models = [LogisticRegression(), SVC(
         kernel='poly', probability=True), XGBClassifier()]

         for i in range(3):
             models[i].fit(X_train, Y_train)

         print(f'{models[i]} : ')
         print('Training Accuracy : ', metrics.roc_auc_score(
             Y_train, models[i].predict_proba(X_train)[:,1]))
         print('Validation Accuracy : ', metrics.roc_auc_score(
             Y_valid, models[i].predict_proba(X_valid)[:,1]))
         print()

         XGBClassifier(base_score=None, booster=None, callbacks=None,
                       colsample_bylevel=None, colsample_bynode=None,
                       colsample_bytree=None, early_stopping_rounds=None,
                       enable_categorical=False, eval_metric=None, feature_types=None,
                       gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
                       interaction_constraints=None, learning_rate=None, max_bin=None,
                       max_cat_threshold=None, max_cat_to_onehot=None,
                       max_delta_step=None, max_depth=None, max_leaves=None,
                       min_child_weight=None, missing=nan, monotone_constraints=None,
                       n_estimators=100, n_jobs=None, num_parallel_tree=None,
                       predictor=None, random_state=None, ...) :
         Training Accuracy :  0.9847772013876351
         Validation Accuracy :  0.9142903645833333
```

Fig 25: Training Accuracy and Validation Accuracy Value after pridiction

Among the three models, we have trained XGBClassifier has the highest performance but it is pruned to overfitting as the difference between the training and the validation accuracy is too high. But in the case of the Logistic Regression, this is not the case.

19. Confusion Matrix

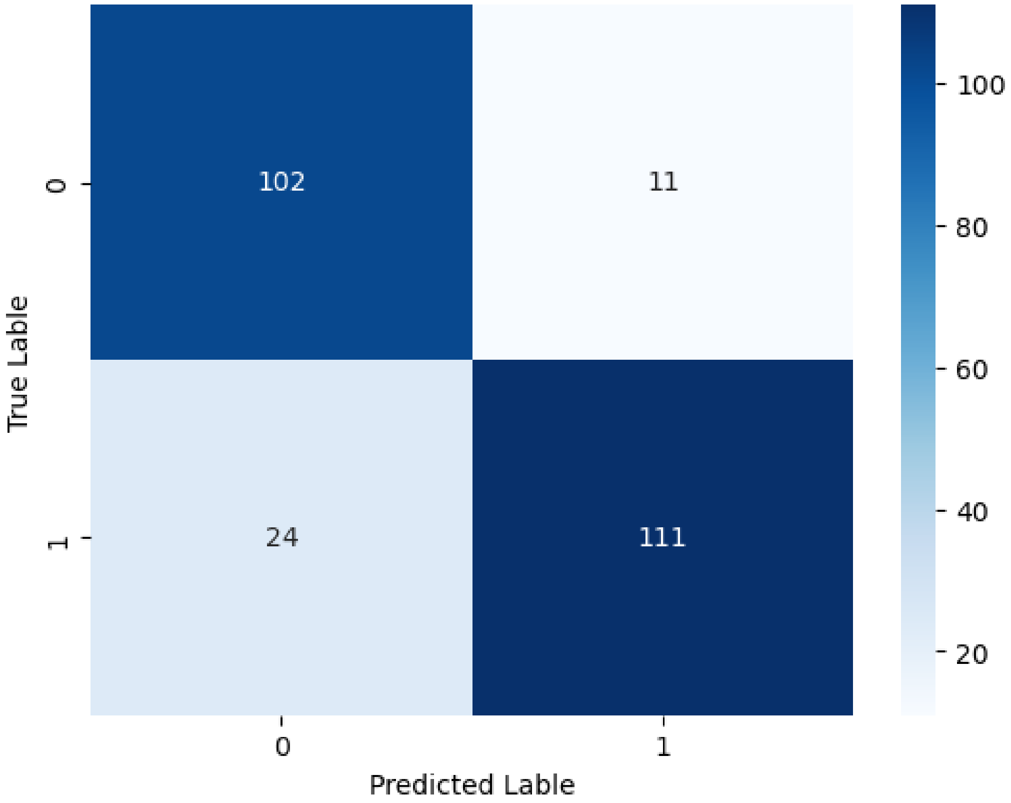
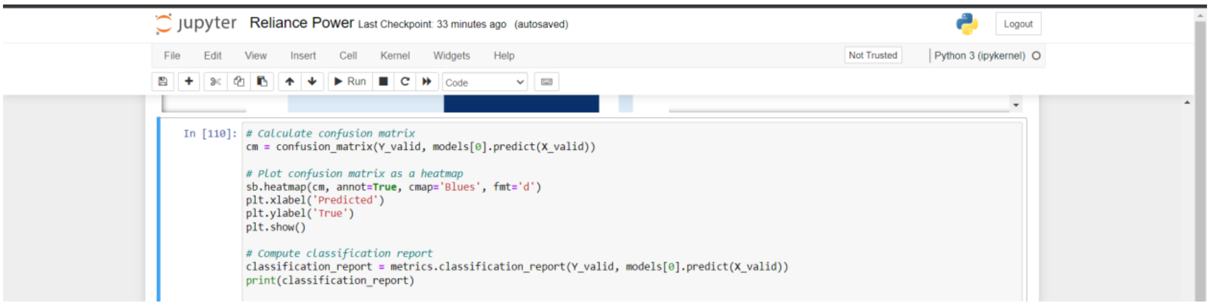
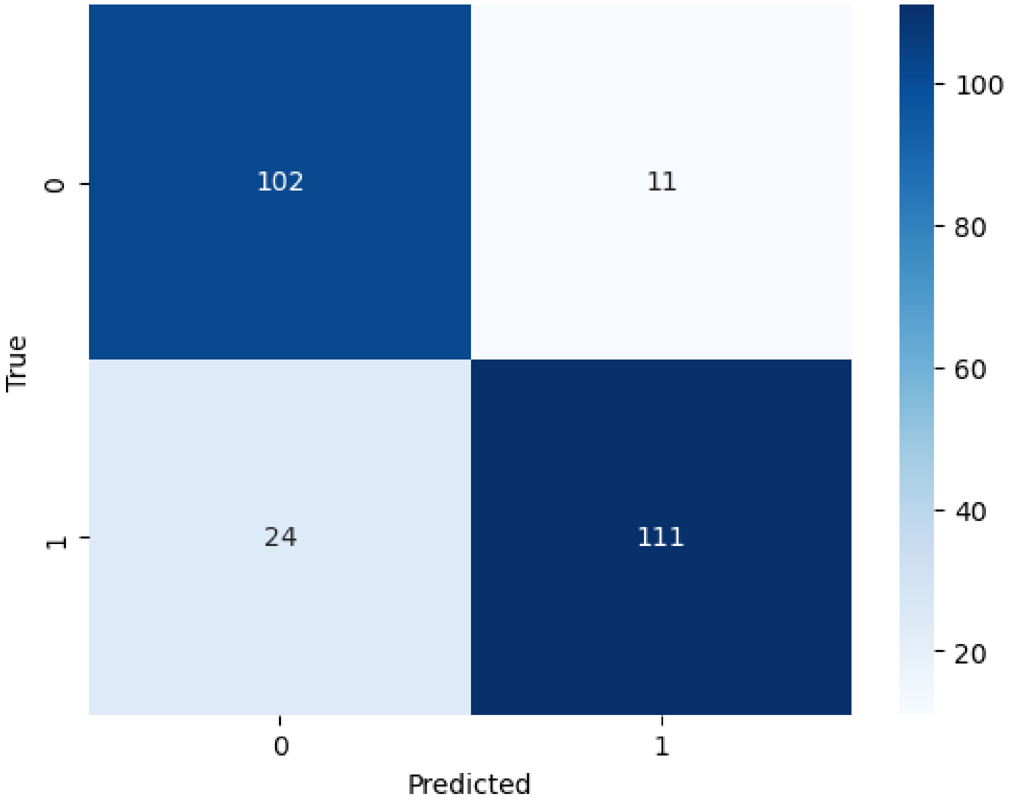


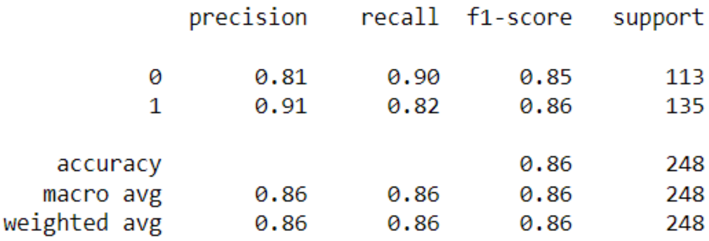
Fig 26: Creating the Confusion matrix for the Reliance Stocks

20. Calculated Confusion Matrix



```
In [110]: # Calculate confusion matrix
          cm = confusion_matrix(Y_valid, models[0].predict(X_valid))

          # Plot confusion matrix as a heatmap
          sb.heatmap(cm, annot=True, cmap='Blues', fmt='d')
          plt.xlabel('Predicted')
          plt.ylabel('True')
          plt.show()

          # Compute classification report
          classification_report = metrics.classification_report(Y_valid, models[0].predict(X_valid))
          print(classification_report)
```



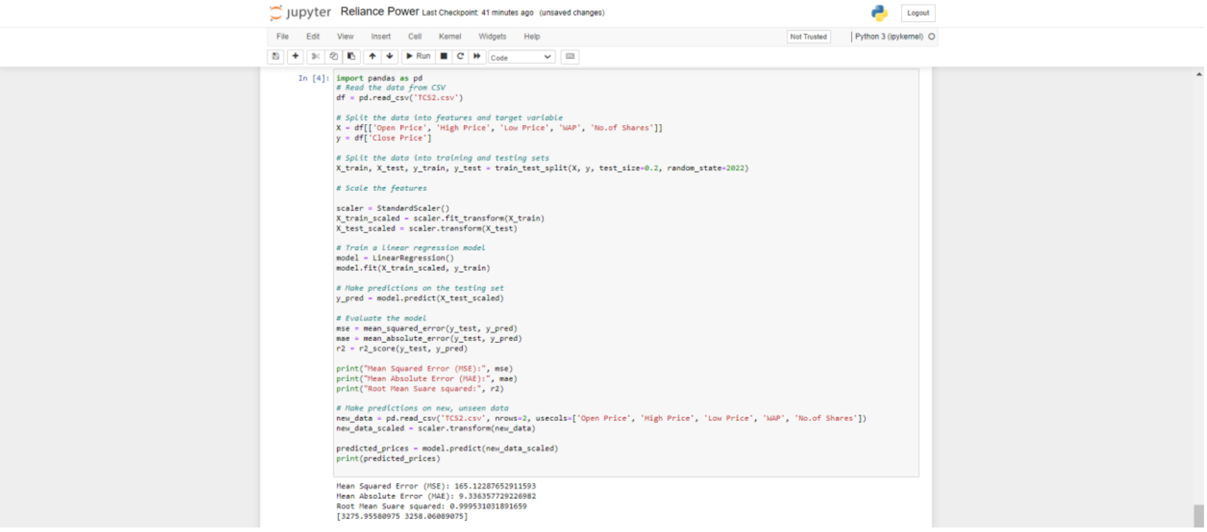


```
              precision    recall  f1-score   support

           0       0.81      0.90      0.85       113
           1       0.91      0.82      0.86       135

    accuracy                           0.86       248
   macro avg       0.86      0.86      0.86       248
weighted avg       0.86      0.86      0.86       248
```

Fig 27: Calculated Confusion matrix for the Reliance Stocks

The confusion matrix is a table that shows the performance of a classification model by comparing the predicted labels with the true labels. It provides insights into the number of true positives, true negatives, false positives, and false negatives.

21. Calculating the Root Mean Square and Accuracy of the Model



```
In [4]: import pandas as pd
        # Read the data from CSV
        df = pd.read_csv('TCS2.csv')

        # Split the data into features and target variable
        X = df[['Open Price', 'High Price', 'Low Price', 'WAP', 'No.of Shares']]
        y = df['Close Price']

        # Split the data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2022)

        # Scale the features

        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)

        # Train a linear regression model
        model = LinearRegression()
        model.fit(X_train_scaled, y_train)

        # Make predictions on the testing set
        y_pred = model.predict(X_test_scaled)

        # Evaluate the model
        mse = mean_squared_error(y_test, y_pred)
        mae = mean_absolute_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)

        print("Mean Squared Error (MSE):", mse)
        print("Mean Absolute Error (MAE):", mae)
        print("Root Mean Suare squared:", r2)

        # Make predictions on new, unseen data
        new_data = pd.read_csv('TCS2.csv', nrows=2, usecols=['Open Price', 'High Price', 'Low Price', 'WAP', 'No.of Shares'])
        new_data_scaled = scaler.transform(new_data)

        predicted_prices = model.predict(new_data_scaled)
        print(predicted_prices)

        Mean Squared Error (MSE): 165.12287652911593
        Mean Absolute Error (MAE): 9.336357729226982
        Root Mean Suare squared: 0.999531031891659
        [3275.95580975 3258.06089075]
```

Fig 28: Value of Root Mean Square

Conclusion

The provided code snippet demonstrates the use of a linear regression model to predict the closing prices of a stock based on given features. Here's a conclusion based on the code:

Data Preparation: The code reads the data from a CSV file named 'Reliance Power.csv' using pandas and splits it into features (X) and the target variable (y). The features include 'Open Price', 'High Price', 'Low Price', 'WAP', and 'No.of Shares'.

Train-Test Split: The data is further split into training and testing sets using the train_test_split function from scikit-learn. The testing set size is set to 20% of the data, and a random state of 2022 is used for reproducibility.

Feature Scaling: The features in the training and testing sets are scaled using the StandardScaler from scikit-learn. The scaling is applied to normalize the feature values.

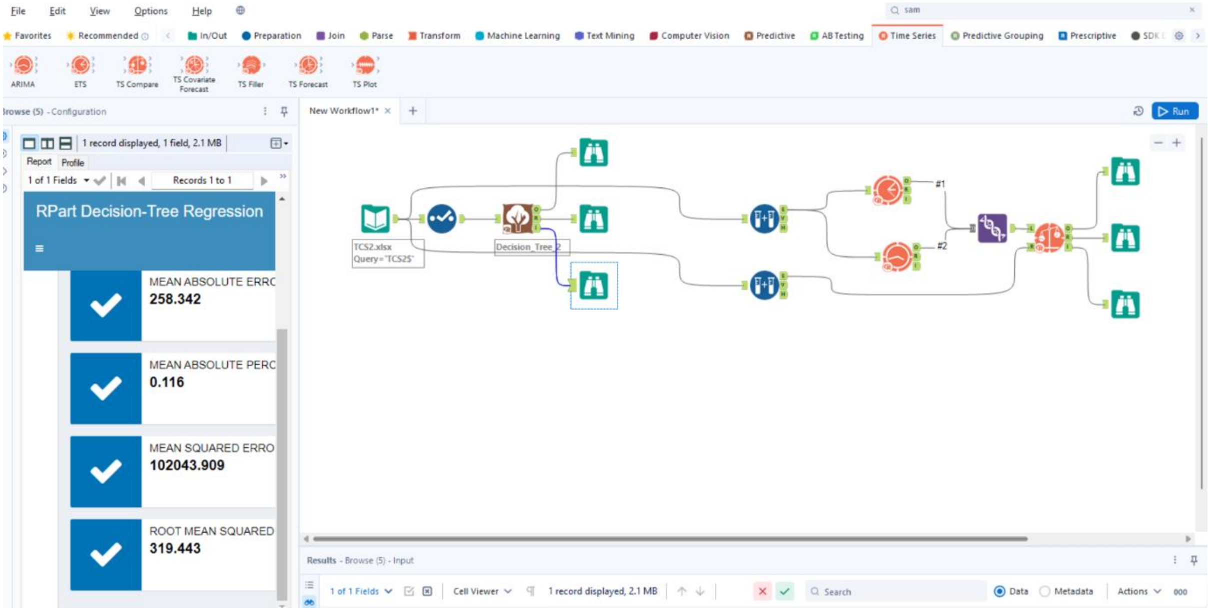
Model Training: A linear regression model is instantiated using the LinearRegression class from scikit-learn. The model is trained on the scaled training data using the fit method.

Prediction and Evaluation: Predictions are made on the scaled testing set using the trained model, and the mean squared error (MSE), mean absolute error (MAE), and Root Mean-squared (r2) scores are calculated to evaluate the model's performance. The evaluation metrics are printed to the console.

Prediction on New Data: Additionally, the code demonstrates making predictions on new, unseen data. Two rows of data are read from the 'Reliance Power.csv' file, containing the same features used for training the model. The new data is then scaled using the same scaler, and the model predicts the corresponding closing prices. The predicted prices are printed to the console.

In conclusion, the code trains a linear regression model on historical stock data and evaluates its performance using various metrics. It also shows how to use the trained model to predict the closing prices of new, unseen data.

22. Comparison With Existing Models



Browse (5) - Configuration

1 record displayed, 1 field, 2.1 MB

Report Profile

1 of 1 Fields Records 1 to 1

RPart Decision-Tree Regression

MEAN ABSOLUTE ERRC
258.342

MEAN ABSOLUTE PERC
0.116

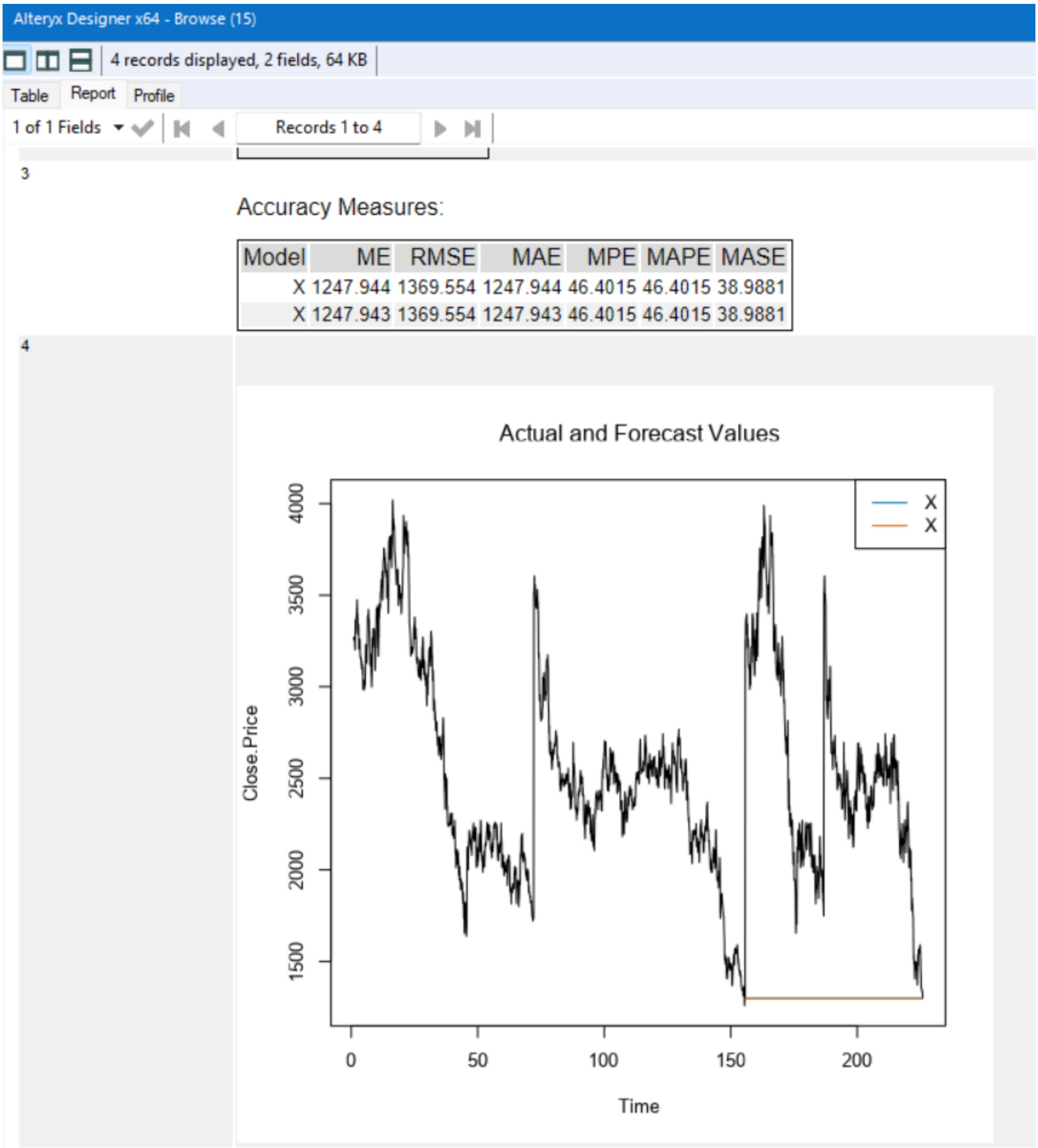
MEAN SQUARED ERRO
102043.909

ROOT MEAN SQUARED
319.443

Fig 29: System Design for Model Comparison

Using the Alteryx Software, we Designed a system which is being used to compare our Decision Tree based model with the existing model Arima and ETS.

23. RMS, MAE and ME Values of ARIMA & ETS Model


Alteryx Designer x64 - Browse (15)
4 records displayed, 2 fields, 64 KB
Table
Report
Profile
1 of 1 Fields
Records 1 to 4
3
4

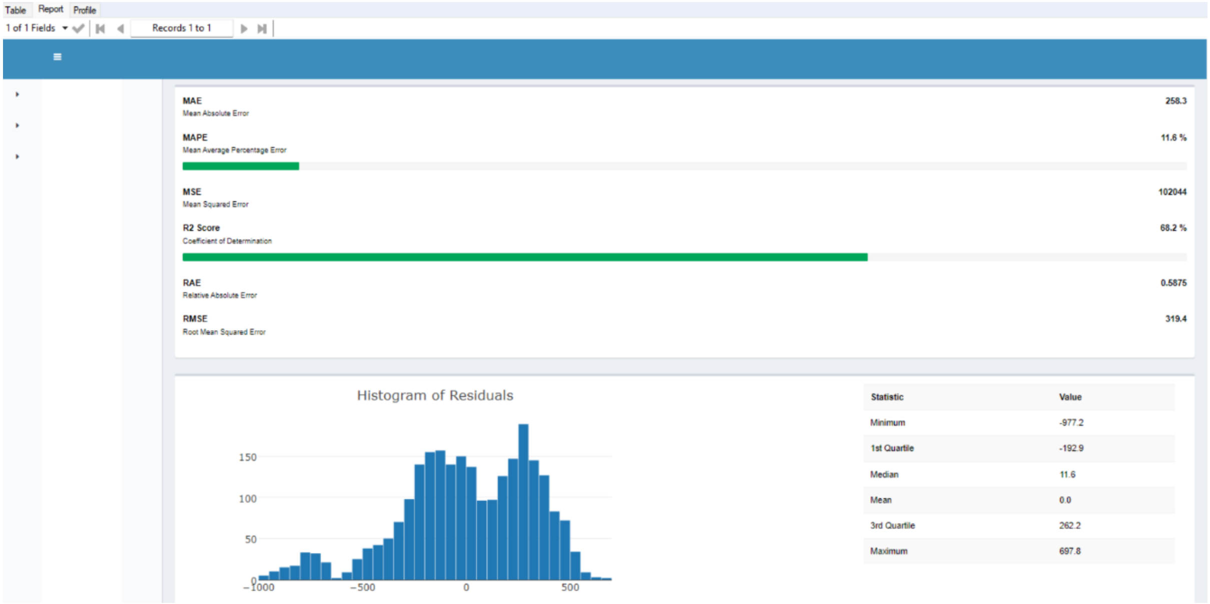
Accuracy Measures:

Model	ME	RMSE	MAE	MPE	MAPE	MASE
X	1247.944	1369.554	1247.944	46.4015	46.4015	38.9881
X	1247.943	1369.554	1247.943	46.4015	46.4015	38.9881

Fig 30: RMS, MEA & ME value of Arima and ETS Model

AS we can observe in the above output the RMS (Root Mean Square), MAE and ME value for the ARIMA and ETS model is 1369 for the given dataset on which the system is constructed.

24. RMS, MAE & ME Values of Decision Tree Model



MAE	Mean Absolute Error	258.3
MAPE	Mean Average Percentage Error	11.6 %
MSE	Mean Squared Error	102044
R2 Score	Coefficient of Determination	68.2 %
RAE	Relative Absolute Error	0.5875
RMSE	Root Mean Squared Error	319.4

Statistic	Value
Minimum	-977.2
1st Quartile	-192.9
Median	11.6
Mean	0.0
3rd Quartile	262.2
Maximum	697.8

Fig 31: RMS, MAE & ME value of Decision Tree Model

AS we can observe in the above output the RMS (Root Mean Square), MAE and ME value for the Decision Tree model is just 319 for the given dataset on which the system is constructed which very less when compared with the existing models ARIMA and E

References

[1] W. K. Wong, M. Manzur, and B. K. Chew, "How rewarding is technical analysis? Evidence from Singapore stock market," *Applied Financial Economics*, vol. 13, no. 7, pp. 543–551, Jul. 2003, doi: 10.1080/0960310022000020906.

[2] Milind Paradkar, "Machine Learning Application in Forex Markets," *quantinsti.com*, Mar. 28, 2016.

[3] M. Segal, "Tree Depth in a Forest NUS / IMS Workshop on Classification and Regression Trees."

[4] J. R. Quinlan, "Induction of Decision Trees," 1986.

[5] by J. Ross Quinlan, M. Kaufmann Publishers, and S. L. Salzberg, "Programs for Machine Learning," 1994.

[6] T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka, "Stock Market Prediction System with Modular Neural Networks."

[7] A. Abraham, B. Nath, and M. P. K, "Hybrid Intelligent Systems for Stock Market Analysis."

[8] K. H. Tsai and J. C. Wang, "External technology sourcing and innovation performance in LMT sectors: An analysis based on the Taiwanese Technological Innovation Survey," *Res Policy*, vol. 38, no. 3, pp. 518–526, Apr. 2009, doi: 10.1016/j.respol.2008.10.007.

[9] K.-H. Han, "Genetic Quantum Algorithm and its Application to Combinatorial Optimization Problem."

[10] L. Khaidem, S. Saha, and S. R. Dey, "Predicting the direction of stock market prices using random forest," Apr. 2016, [Online]. Available: http://arxiv.org/abs/1605.00003

[11] K. K. Aggarwal, Y. Singh, P. Chandra, and M. Puri, "Evaluation of various training algorithms in a neural network model for software engineering applications," *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, pp. 1–4, Jul. 2005, doi: 10.1145/1082983.1083003.

[12] Chainika Thakar, "Machine Learning for Trading," *quantinsti.com*, Aug. 16, 2021.

[13] Larry Page and Sergey Brin, "Google Finanace," *google.com*, Sep. 04, 1998.

[14] "trading algorithms work," *quora.com*.

[15] QuantInsti Blog, "Algo Trading & Quant Finance," *quantinsti.com*.

[16] andredumas, “Data Plots,Indicators,Interaction & Other Examples,” *github.com*, Sep. 28, 2016.

[17] Chainika Thakar, “Regression,” *quantinsti.com*, Apr. 18, 2023.

[18] Sushant Ratnaparkhi & Milind Paradkar, “Decision Trees,” *quantinsti.com*, Oct. 17, 2017.

[19] Eric Hammer, “Machine Learning for Quantitative Finance,” *quantinsti.com*, Apr. 28, 2017.

[20] “Financial companies use machine learning,” *www.quora.com*.

[21] Premchand Roychand, “BSE Index,” *bseindia.com*, Jul. 09, 1875.