A STUDY ON STOCK PRICE PREDICTION USING MACHINE LEARNING ON SELECTED AUTO MOBILE STOCKS WITH REFERENCE TO ANGEL ONE.LTD, ANANTAPUR

Submitted

In partial fulfilment of the requirements for the award of the degree of

MASTER OF BUSINESS ADMINISTRATION (BDA)

In

SCHOOL OF MANAGEMENT STUDIES

By

THAMMINENI PAVITHRA

[Regd.No:2I0A1E0430]

Under the supervision and guidance of

Dr. P. SRAVAN KUMAR M.B.A., Ph.D.

SCHOOL OF MANAGEMENT STUDIES

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITYANANTAPUR

ANANTAPURAMU-515002 ANDHRA PRADESH

2021-2023

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

ANANTAPUR – 515002

SCHOOL OF MANAGEMENT STUDIES

BONAFIDE CERTIFICATE

This is to certify that the project report, entitled "**A STUDY ON STOCK PRICE PREDICTION USING MACHINE LEARNING ON SELECTED AUTO MOBILE STOCKS WITH REFERENCE TO ANGEL ONE.LTD, ANANTAPUR"** submitted to the **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, ANANTAPUR** in Partial fulfilment of the requirements for the award of the Degree of **MASTER OF BUSINESS ADMINISTRATION** is a record of original research work done by **THAMMINENI PAVITHRA** during the period 11th MAY 2023 to 15th AUG 2023 of her study in the "**MASTER OF BUSINESS ADMINISTATION (BUSINESS DATA ANALYTICS)"**at "**JAWAHARLAL NEHRU TECHNOLOGICAL UNIIVERSITY ANANTAPUR"** under the supervision and guidance of **DR.P. SRAVAN KUMAR M.B.A., Ph.D.,** and thetraining report has not formed the basis for the award of any Degree / Diploma / Associate ship / Fellowship or other similar title to any candidate of any University.

Internal guide **Head**

JNTUA SMS

Viva voce examination held on: _____

Internal Examiner **External Examiner**

Date:

Place: ANANTAPURAMU.

DECLARATION

I hereby declare that the project report entitled "**A STUDY ON STOCK PRICE PREDICTION USING MACHINE LEARNING ON SELECTED AUTO MOBILE STOCKS WITH REFERENCE TO ANGEL ONE.LTD, ANANTAPUR"** Is record of independent research work and is carried out by me during the period of my study, submitted to **JNTUA SCHOOL OF MANAGEMENT STUDIES, ANANTAPUR,** under the guidance of **DR.P. SRAVAN KUMAR M.B.A., Ph.D.,** Assistant professor (A) in School of Management Studies, **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY** for the award of the degree of **MASTER OF BUSINESS ADMINISTRATION (BUSINESS DATA ANALYTICS 2021-2023)** is my original work, has not been submitted to any other university for the award of any degree.

Date THAMMINENI PAVITHRA

Place: ANANTAPURAMU **210A1E0430**

AngelOne

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Miss. THAMMINENI PAVITHRA (Reg No: 210A1E0430)** student from Jawaharlal Nehru Technological University, Anantapuram has successfully completed "**A STUDY ON STOCK PRICE PREDICTION USING MACHINE LEARNING ON SELECTED AUTO MOBILE STOCKS WITH REFERENCE TO ANGEL ONE Ltd. AT ANANTAPURAMU** for 45 days from

01/03/2023 to 15/04/2023.

This certificate has been issued towards partial fulfilment of her academic degree. Her conduct during the project with us was satisfactory. We wish her all success for her future endeavour.

EMPLOYEE NAME : JAKKA SESHU KRISHNA MURTHY
EMPLOYEE ID : E65880
DESIGNATION : CLUSTER HEAD
BRANCH TAG : ANTPUR
MAIL : SESHU.KRISHNAMURTHY@ANGELBROKING.COM
CELL : 9515144445
DATE :
SIGNATURE :

For "**Angel One Ltd**"

ANGEL ONE LIMITED

JAKKA SESHU KRISHNA

Cluster Head

Skip to Main Content
Assignments
Students
Grade Book
Libraries
Calendar
Discussion
Preferences
About this page

This is your assignment inbox. To view a paper, select the paper's title. To view a Similarity Report, select the paper's Similarity Report icon in the similarity column. A ghosted icon indicates that the Similarity Report has not yet been generated.

ASTUDYON STOCK PRICE PREDICTION USING MACHINE LEA...

Inbox | Now Viewing: new papers ▼

Submit File Online Grading Report | Edit assignment settings | Email non-submitters

Delete Download move to...

	Author	Title	Similarity	web	publication	student papers	Grade	response	File	Paper ID	Date
☐	Thammineni Pavithra	ASTUDYON STOCK PRICE PREDICTION USING MA...	28% 28%	16%	5%	24%			download paper	2144298226	11-Aug-2023

HEAD
School of Management Studies
JNTUA Anantapur
Ananthapuramu-515002 (A.P.)

ACKNOWLEDGEMENT

I express my sincere gratitude to **Prof. T. NARAYANA REDDY** Head of the School of Management Studies for his constant support and encouragement.

I would never forget to thank **Dr. P. Sravan Kumar M.B.A., Ph.D.,** Assistant Professor (A), and School of Management Studies for his valuable guidance in preparation of this project work.

I also thank our faculty members **Dr.B.C. Lakshmanna, Dr.P.Basaiah**, **Dr.P. Jaya rami Reddy, K.Srilatha, Dr.D. Varalakshmi, Dr.P. Sravan Kumar, Dr.D.Jakir Hussain, Dr.P. Viswanath, Dr. K. Nagaraju** for their keen interest and constructive criticism, which played vital role in completing my project on time.

I extend my special thanks to Mr. Sheshu Manager, Angle One Ltd Anantapur for his kind support to complete this project

I also extend my thanks to all my family members and my friends who helped me incompleting this project work successfully.

THAMMINENI PAVITHRA

210A1E0430

ABSTRACT

In the financial industry, forecasting stock prices is a difficult endeavor, but precise predictions can offer traders and investors insightful information. This study investigates the use of linear regression, a well-liked machine learning algorithm, for stock price prediction. The study's primary source of information is historical automobile stock market data, such as Eicher Motors Limited, Hero MotoCorp Limited, Tata Motors Limited, Maruti Suzuki India Limited, Ashok Leyland Limited, which are listed as top NIFTY 50 index values of the National Stock Exchange (NSE) of India, during the period JAN 1, 2019 till MAY 1, 2023. Which includes information on things like opening and closing prices, trading volume, high and low prices and other important financial indicators. Preprocessing is done on the dataset to deal with missing values, outliers, and any other irregularities that can skew the predictions. The preprocessed data is then split into training and testing sets, and the training set is then used to train the linear regression model. Between the input features and the target variable, which is the target variable itself, the model learns the underlying patterns and relationships. And also by using the hyper parameters of the linear regression model are optimized so that it is help full for predicting the future 60 days stock price values.

Key words: stock price prediction, Stock Price Prediction, Linear Regression Financial Analysis, Machine Learning, NIFTY 50 Index.

CONTENTS

S.No Chapter	Particular	Pages No.
1	INTRODUCTION	1-10
2	INDUSTRY AND COMPANY PROFILE	
	2.1 INDUSTRY PROFILE	11
	2.2 COMPANY PROFILE	12-17
3	REASEACH METHODOLOGY	
	3.1 NEED OF THE STUDY	18
	3.2 SCOPE OF THE STUDY	29
	3.3 OBJECTIVES OF THE STUDY	20
	3.4	21
	TOOLS AND TECHNIQUES	22
	LIMITATIONS	23
4	DATA ANALYSIS AND INTREPRETATION	26-57
5	FINDINGS, SUGGESTIONS AND CONCLUSION	
	FINDINGS	58-59
	SUGGESTIONS	60
	CONCLUSION	61
	BIBLIOGRAPHY	63-64

LIST OF CHARTS

Figure No	Particulars	Page No
4.1.3.1	Moving Averages_Eicher Motors For Eicher Motors	26
4.1.5.1	Correlation Between Features For Eicher Motors	29
4.1.8.1	Linear Regression- Predicted Vs Actual Values For Eicher Motors	33
4.1.11.1	Difference Between Actual Vs Predicted Values For Eicher Motors	37
4.1.13.1	Actual vs Predicted Price Moments For Eicher Motors	38
4.2.2.1	Moving Averages For HERO Motors	41
4.2.5.1	Actual vs Predicted Price Moments For HERO Motors	43
4.3.2.1	Moving Averages For TATA Motors	45
4.3.6.1	Actual vs Predicted Price Moments For TATA Motors	47
4.4.2.1	Moving Averages For MARUTHI SUZUKI Motors	49
4.4.5.1	Actual vs Predicted Price Moments For MARUTHI SUZUKI	51
4.5.2.1	Moving Averages For ASHOK LEYLANDS	53
4.5.5.1	Actual vs Predicted Price Moments For ASHOK LEYLANDS	55

CHAPTER-I
INTRODUCTION

INTRODUCTION

The task of anticipating future stock prices using past data and several market indicators is known as stock price prediction. It entails analyzing financial data and forecasting the future performance of a stock using statistical models and machine learning techniques. Stock price forecasting is to provide traders, financial analysts, and investors with relevant information that can lower risk and aid in decision-making. However, it is essential to make the proper stock investments at the right moment. Traditional approaches to stock price forecasting, such fundamental analysis and technical analysis, are unable to capture the complex and dynamic nature of the stock market. Stock price prediction using machine learning is the process of predicts the future value of a stock traded on a stock exchange in order to generate money. Because there are so many factors to consider, it is challenging to anticipate stock values with high accuracy, which is why machine learning is so crucial.

By using stock price prediction using machine learning, we may estimate the value of company shares and other financial assets traded on an exchange in the future. In order to maximize investment returns, stock price predictions are made. The process of forecasting future changes in a single stock's price on the financial markets is known as stock price prediction. Traders and investors commonly try to predict stock prices in order to maximize their returns and make wise investment selections.

Stock price forecasting requires examination of market trends, historical price data, and various other factors that can influence a stock's performance. A few of these variables are the company's financials, market trends, macroeconomic statistics, current news, and investor sentiment. Numerous methods and techniques, such as fundamental analysis, technical analysis, and quantitative modeling, are used to forecast stock prices. Fundamental analysis includes evaluating a company's financial situation, growth prospects, and industry dynamics in order to determine its underlying worth and likely future stock performance. Technical analysis is used to forecast price trends, trading volumes, and price fluctuations.

When projecting stock prices, additional factors including psychological considerations, rational and irrational conduct, and other factors are also taken into account. These factors combine to produce a volatile and dynamic share market. As a result, making accurate stock price predictions is quite difficult. But as compared to other methods of prediction, linear

regression enables a more precise forecasting of stock values. To forecast stock prices, you must first become proficient in collecting, visualizing, and analyzing data from the stock market. The simple linear regression method is used by experts and amateurs alike to try to extrapolate the present or historical trend into the future, despite the fact that there are many sophisticated techniques and systems developed to anticipate future stock values. You could have a chance winner if the current pattern holds going forward.

Utilizing machine learning to create predictions based on the values of current stock market indices by training on their past values is a recent development in stock price prediction technologies. Multiple models are used by machine learning itself to facilitate and authenticate prediction. The research focuses on using machine learning techniques based on linear regression to forecast stock values. Open, close, low, high, and volume are all factors.
The mysterious challenge of predicting stock values is one that many people try. Even simple linear models can occasionally achieve unexpectedly high levels of accuracy, even when achieving perfect accuracy might not be possible. This post will show you how to create an easy linear regression model in Python that you can easily integrate with any algorithmic trading strategy! In Python, it is easy to predict stock prices using linear regression. Finding the right combination of characteristics to make those forecasts viable is a different story. In this post, we'll train a regression model using past pricing data and technical signals to predict future prices.

1.1 Meaning of stock price prediction:

Stock price prediction refers to the process of forecasting future prices of stocks or securities traded in financial markets. It involves using historical price data, along with various quantitative and qualitative factors, to develop models and algorithms that can estimate the future movements and trends of stock prices. Stock price prediction aims to assist investors, traders, and financial analysts in making informed decisions, managing risks, and potentially maximizing returns in the volatile and dynamic stock market environment.

1.2 Definition of stock price prediction:

Stock price prediction can be defined as the application of statistical and machine learning techniques to forecast future prices of stocks or securities in financial markets based on historical data and relevant factors.

John Smith

1.3 What is stock Market?

The stock market, usually referred to as the equity market or stock exchange, is a marketplace where buyers and sellers can transact in shares of publicly traded businesses. It is an essential part of the world financial system and is essential for capital development and economic expansion.

Companies issue shares of their ownership, referred to as stocks or equities, on a stock market to raise money for a variety of uses like expanding operations, funding research and development, or repaying debts. Investors, including people, institutions, and organizations, purchase and sell these shares with the intention of making a profit. The stock market's main purpose is to make it easier to buy and sell these stocks. Investors are given access to a controlled and open market where they can exchange shares. Stock exchanges serve as central marketplaces where buyers and sellers come together to make transactions, such as the New York Stock Exchange (NYSE) and NASDAQ in the United States.

Through the sale of shares to investors, the stock market enables businesses to raise capital in addition to the more conventional debt financing. In exchange, investors take on a portion of ownership in the business and stand to gain from possible capital growth as well as dividend income. The macroeconomic environment, company performance, market trends, geopolitical events, and investor attitude all have an impact on the stock market. The supply and demand for stocks are influenced by all of these factors, which causes price swings. Stock prices, which can be very volatile in the short term but typically represent the underlying worth and future prospects of the companies over the long term, are determined by the interplay of buyers and sellers in the stock market.

Exchange-traded funds (ETFs), which are investment funds that trade on stock exchanges and represent a basket of underlying assets, and mutual funds, which pool money from multiple investors to invest in a diversified portfolio of stocks, are two additional investment options available on the stock market in addition to individual stock trading. The stock market is used as a gauge of a nation's general economic health and investor confidence. It carries risks and uncertainties along with chances for wealth accumulation and capital expansion. Investors must conduct thorough stock analyses and

evaluations and proceed cautiously when making investment decisions. Before investing in the stock market, it is wise to consult with financial experts and do extensive study.

1.4 What is stock price?

Stock price is the going rate at which a share of a specific stock is purchased or sold on the stock market. It represents the price at which investors are prepared to purchase or dispose of a particular stock at a particular time.

Supply and demand dynamics in the stock market are what drive stock prices. The price of a stock typically rises when there is strong demand and a constrained supply. The stock price could fall, on the other hand, if there is a shortage of demand or an excess of supply.

Stock prices are affected by a number of variables, such as company performance, financial stability, market movements, macroeconomic conditions, news events, and investor sentiment. Positive news can include great profit reports or the introduction of new products can increase stock prices as investors grow more positive about the company's prospects for the future. On the other hand, bad news might cause a drop in stock values, such as poor financial reports or unfavorable regulatory actions.

It's crucial to remember that stock prices can fluctuate often and can be very volatile depending on market conditions and investor emotion. Both short-term factors, like market speculation or trading activity, and long-term factors, such the company's growth potential and industry trends, can have an impact on prices.

1.5 What is stock market prediction?

Understanding the different stock market elements that may affect a stock's price and developing a model to anticipate the stock's price are both terms used to describe stock market prediction. This can assist individuals and institutions in making predictions about the direction of stock prices and in deciding whether to buy or sell stocks in order to make the most money.

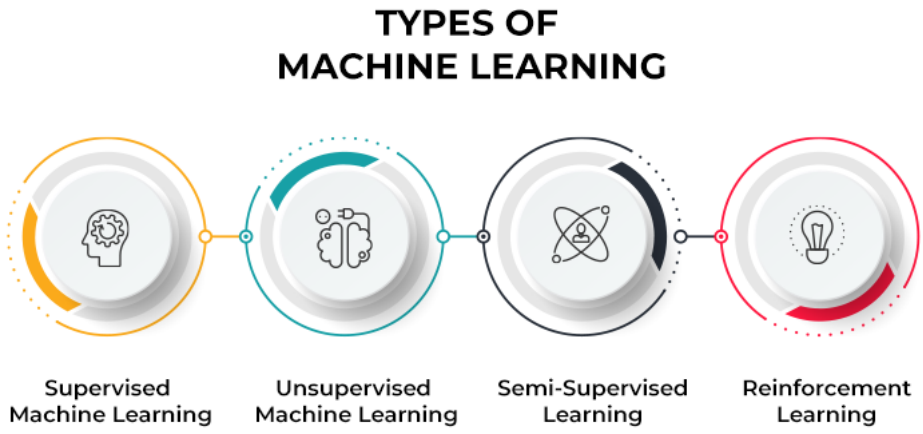
The article's readers can anticipate having a basic understanding of all the underlying phenomena, procedures, and technology involved in stock market predictions. The essay will cover a wider range of subjects, including how a predictive model can be developed, popular stock market research approaches, and basic and sophisticated methods for stock price prediction making use of machine learning. We will also talk about the typical difficulties that model developers have while building such complex and sophisticated prediction models.

1.6 What is machine learning?

With the use of machine knowledge (ML), which is a form of artificial intelligence (AI), software programs can predict issues more directly without having to be explicitly instructed to do so. In order to read new affair values, machine knowledge algorithms use nonfictional data as input. Machine knowledge is constantly used in recommendation machines. Business process automation (BPA), predictive conservation, spam filtering, malware trouble discovery, and fraud discovery are a numerous fresh common uses.

1.7 What are the types of machine learning?

Supervised learning, unsupervised learning, and reinforcement learning are the three main types of machine learning. Each variety of machine learning algorithm tackles various learning circumstances and has its own special traits and uses.



1.7.1 Supervised Learning: In supervised learning, input characteristics and target labels are used as the training data for a machine learning model. The objective is to teach the algorithm to forecast the right labels for brand-new, untainted data. Classification and regression algorithms are examples of supervised learning algorithms. Several well-liked supervised learning methods include:

1. A decision tree

2. Arbitrary forests
3. SVMs, or support vector machines
4. Linear regressive
5. Rational regression
6. Deep learning neural networks

1.7.2 Unsupervised Learning: When learning from unlabeled data, unsupervised learning entails discovering patterns and structures without the use of predetermined target labels. In order to uncover significant patterns or groupings, the algorithm investigates the relationships and inherent structure in the data. For tasks like grouping, anomaly detection, and dimensionality reduction, unsupervised learning is frequently utilized.

Typical unsupervised learning methods include:

1. K-means grouping
2. Using a hierarchy to cluster
3. Gaussian Mixture Models (GMM),
4. Association Rule Learning (Apriori method),
5. Principal Component Analysis (PCA)

1.7.3 Semi-supervised learning: Semi-supervised learning is a method of machine learning that combines the two categories mentioned above. An algorithm may be fed mostly labeled training data by data scientists, but the algorithm is allowed to explore the data on its own and come to its own conclusions about the data set.

1. Self -training
2. Co-training
3. Multi –view learning

1.7.4 Reinforcement Learning: Reinforcement learning is the process through which an agent develops the ability to interact with the environment and make choices that will maximize a cumulative reward signal. The agent gains knowledge by making mistakes and then receiving feedback in the form of incentives or punishments based on its behavior. Learning an ideal policy to direct the agent's decision-making is the aim. Robotics, gaming, and autonomous systems are just a few examples of fields where reinforcement learning techniques are frequently employed. Several prominent reinforcement learning algorithms are:

1. Deep Q-Networks (DQN)

2. Q-Learning
3. Policy Gradient Approaches
4. Actor-Critical techniques
5. Proximal Policy Optimization (PPO)

Before getting into the intricacies of machine learning algorithms that anticipate stock prices, let's first explore how stock market prediction using machine learning is even possible. As stated in the introduction, the user must provide both the target variable whose values or labels must be predicted as well as the independent variables (X variables) that can forecast those values. Machine learning is distinct from other traditional forecasting techniques. To get the result, which is extremely accurate (but challenging to understand), it combines challenging mathematical algorithms, cross-validation processes, optimization algorithms, and other methods.

1.8 What is Linear Regression?

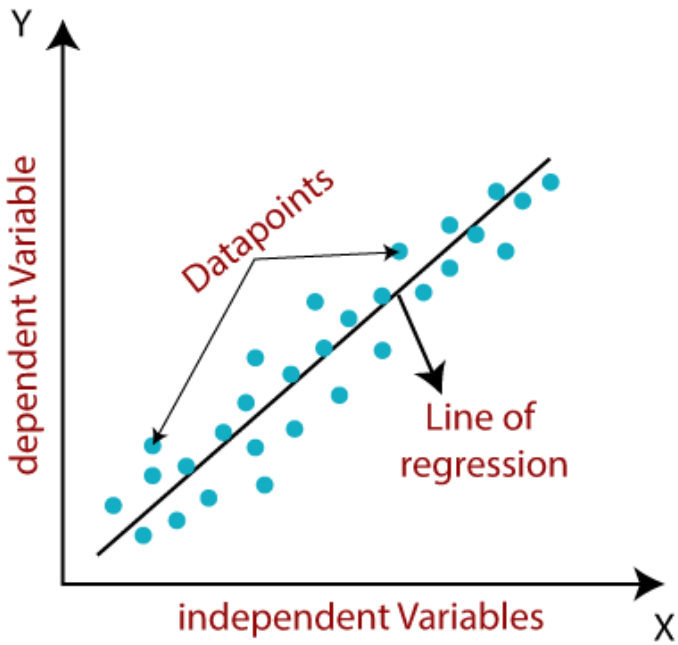
A continuous target variable can be predicted using a supervised learning process called linear regression based on one or more input features. It creates a linear relationship between the input variables, also known as independent variables or predictors, and the goal variable, also known as the dependent variable, using a straightforward yet effective statistical modeling technique.

By fitting a linear equation to the data, linear regression attempts to determine how well the input variables and the target variable relate to one another. Then, using fresh, unforeseen data, this equation can be used to create predictions. Essentially, the goal of linear regression is to identify the line that minimizes the overall gap between the target variable's actual values and its predicted values.
A simple linear regression model's equation looks like this:

$$Y = \beta_0 + \beta_1 X$$

Where:
The variables Y and X stand for the target variable (the variable we want to forecast), the intercept term (the value of Y when X is 0), and the slope coefficient (the change in Y

for a unit change in X).



One appealing use of linear regression is stock market predictions. Implementing these analyses in a few lines of code is now easy thanks to contemporary machine learning tools like scikit-learn. In order to anticipate stock prices, linear regression models can be used to examine trends, compute momentum indicators, identify support and resistance levels, forecast prices, and determine the significance of features. They can help you understand the general trend in stock prices, gauge how quickly and strongly prices fluctuate, and locate prospective buying and selling opportunities. Additionally, using previous data and pertinent attributes, linear regression can forecast future price changes. It's crucial to remember that linear regression has limits in terms of capturing complicated market dynamics, and that it is frequently used in conjunction with more sophisticated techniques for forecasts that are more precise.

1.9 What is a Moving Averages?

An average of data points for a given time period, typically prices, is what is known as a moving average. Every data point is computed using information from the X prior periods. Moving averages smooth the price data to create a trend-following indicator by averaging preceding data. A moving average cannot forecast the movement of prices. Instead, it indicates the present course. A moving average, on the other hand, usually lags because it is based on prior prices. Despite this, traders employ moving averages to

smooth out price movement and eliminate noise.

The simple moving average (SMA) and the exponential moving average (EMA) are the two most widely used moving averages. While exponential moving averages (EMAs) lend more weight to current prices, simple moving averages (SMAs) simply average values across the given timeframe. Our charting tools now support the DEMA, Hull Moving Average, KAMA, and TEMA specialist types of moving averages.

1. 10 Types of Moving Averages?

Primarily, there are three moving average types, and they are explained below-

1. 10.1 Straight-Line Moving Average:

By computing the average stock price over various time intervals, a simple moving average, or SMA, can be created. The closing prices serve as the primary foundation for this.

Simple Moving Average Calculation Formula-

$$\text{Simple Moving Average} = A_1 + A_2 + A_3 \ldots\ldots A_n / n$$

With a moving average example, this can be referenced more effectively.

Finding the average prices for a specific period and then dividing their sum by the total number of periods are the first two steps in constructing simple moving averages.

Let's say Robin wants to determine the XYZ Stock's simple moving average by taking the closing prices from the previous five days into account.

The closing prices for the previous five days are represented by Rs. 24,

Rs 24.60

The SMA can be calculated as follows:

SMA = Rs. (24 + 25.50 + 24.75 +25.10 + 24.60) / 5

SMA = thus Rs. 24.79.

1.10.2 Weighted Moving Average:

The numerous shortcomings of the SMA are offset by the weighted moving average (WMA). It prioritizes current data over historical data. WMA adheres more closely than SMA to the various stock price tiers.

Formula for Weighted Moving Average Calculation-

$$\text{WMA} = \frac{\text{Price}_1 \times n + \text{Price}_2 \times (n-1) + \cdots \text{Price}_n}{\frac{n \times (n+1)}{2}}$$

1.10.3 Exponential Moving Average:

This also goes by the name of EMA and requires intricate calculations. Similar to WMA, EMA gives more importance to a financial instrument's most recent pricing.

Plotting a 100-day EMA and 100-day SMA on the same chart reveals that the former responds more quickly than the latter. This occurs as a result of the current pricing receiving more attention.

Calculating the Exponential Moving Average Formula-

$$EMA_t = \left[V_t \times \left(\frac{s}{1+d}\right)\right] + EMA_y \times \left[1 - \left(\frac{s}{1+d}\right)\right]$$

CHAPTER – II

INDUSTRY

AND

COMPANY PROFILE

2.1 INDUSRTYPROFILE

Type	Stock exchange
Location	Mumbai, Maharashtra, India
Founded	1992,31 years ago
Owners	Various group of domestic and global financial institutions, public and privately owned entities and individuals
Key people	Girish Chandr Chaturvedi (Chairperson) Ashishkuma (MD & CEO)
Currency	Indian rupee(₹)
No. of listing	2.002
Market cap	US$3.27 trillion(jan 2023)
Indices	NIFTY 50 NIFTY next 50 NIFTY 500
Website	www.nseindia.com

2.2 COMPANY PROFILE

Name	Angel One Limited
Type	Public, Financial services
Founded	8 August 1996, 26 years ago
Founder	Dinesh D. Thakkar.
Headquarters	Mumbai, Maharashtra, India
Area served	India
Key people	Dinesh D. Thakkar (MD & Chairman), Narayan Gangadhar (CEO)
NO. of employees	7139
Branches	142
Revenue	Rs. 7,427.79 (2020)
Operating Income	Rs. 7,105.8 crore (2020)
Net income	Rs. 866.24 crore (2023)
Total assets	Rs.21,592.05 crore (2023)
Website	http://www.angelone.in/

Name	EICHER MOTORS LIMITED
Type	Public
Founded	1948; 75 years ago
Founder	Vikram Lal
Headquarters	New Delhi, India
Area served	World wide
Key people	S Sandilya (Chairman) Siddhartha Lal (managing director)
NO. of employees	4,899
Products	Commercial vehicles Motorcycle
Revenue	Rs.15,037.29 crore
Operating Income	Rs. 3,799.63 crore
Net income	Rs. 2,913.93 crore
Total assets	Rs. 19,197.64 crore
Website	https://www.eicher.in/

Name	MARUTHI UDYOG LIMITED
Type	Public

Founded	24 February 1981; 42 years ago
Founder	Government of India Suzuki Motor Corporation
Headquarters	New Delhi, India
Area served	India
Key people	R. C. Bhargava (Chairman) Hishahi Takeuchi (managing director & CEO)
NO. of employees	16,025
Products	Automotive finance Commercial vehicles Motorcycle
Revenue	Rs.119,7129 crore
Operating Income	Rs. 10,323.10 crore
Net income	Rs. 8,211 crore
Total assets	Rs. 84,596.90 crore
Total equity	Rs. 61,791.30 crore
Website	http://www.marutisuzuki.com/

Name	TATA ENGINEERING AND LOCOMOTIVE COMPANY LIMITED(TELCO)
Type	Public
Founded	1945; 78 years ago
Founder	Jehangir Ratanji Dadabhoy Tata
Headquarters	Mumbai, Maharashtra, India
Area served	Worldwide
Key people	Natarajan Chandrasekaran (Chairman)
NO. of employees	59,000+(2022)
Products	Automotive finance Commercial vehicles Motorcycle Automotive parts Pickup trucks
Revenue	Rs.350,600.15 crore
Operating Income	Rs. 7,059,30 crore
Net income	Rs. 2,414.30 crore
Total assets	Rs. 336,081.80 crore
Total equity	Rs. 45,321.80 crore
Website	https://www.tatamotors.com/

Hero MotoCorp Ltd

Name	Hero MotoCorp Limited
Type	Public
Founded	19 January 1984; 39 years ago
Founder	Brijmohan Lall Munjal
Headquarters	New Delhi, India
Area served	Worldwide
Key people	Pawan Munjal(Chairman, MD, & CEO)
NO. of employees	8,599
Products	Motorcycle Scooters
Revenue	Rs.34,727.39 crore
Operating Income	Rs. 3,863.62 crore
Net income	Rs. 2,799.90 crore
Total assets	Rs. 23,917.03crore
Total equity	Rs. 16,788880.63 crore
Website	http://www.heromotocorp.com/

Name	ASHOK LEYLAND
Type	Public
Founded	7 September 1948; 74 years ago
Industry	Automotive
Headquarters	Chennai, Tamil nadu, India
Area served	Worldwide
Key people	Natarajan Chandrasekaran (Chairman)
NO. of employees	11,463
Products	Automobile Commercial vehicles Engines
Revenue	Rs.41,783 crore
Operating Income	Rs. 2,268 crore
Net income	Rs. 1,361 crore
Total assets	Rs. 54,728 crore
Total equity	Rs. 10,798 crore
Website	http://www.ashokleyland.com/

CHAPTER -III
RESEARCH
METHODOLOGY

3.1 NEED OF THE STUDY

The need aim to forecast future price moments to help investors make informed decisions.

3.2 SCOPE OF THE STUDY

- The scope of the study is to analyse historical data and implement a predictive model to forecast future prices.

- The study is limited to the data that was available for analysis during the period of three years from 2019-01-01 to 2023-05-01. Selected Stocks Tata Motors, Hero Moto Corporation Maruthi Suzuki, Eicher Motors and Ashok Leyland only.

3.3OBJECTIVES OF THE STUDY

- To analyse the underlying trends in the stock prices of selected auto mobile stocks.
- To develop a model that accurately predicts the future stock prices of selected auto mobile stocks.
- To evaluate the performance of the prediction model.

3.4 SOURCE OF DATA

The study is based on secondary data.

Secondary Data:

The data is collected from the NSE Website.

Website:

https://finance.yahoo.com/

3.5 TOOLS AND TECHNIQUES

TOOLS

- Python

TECHNIQUES

- Machine learning

LIBRARIES

- NumPy
- Matplotlib
- Scikit-learn
- Tensor Flow
- Pandas

3.6 LIMITATIONS OF THE STUDY

- The study is limited to the only selected auto mobile stocks listed in NSE.
- The model does not take into account external factors such as global events, government policies, or natural disasters that may impact stock prices.
- The study is conducted for a period of 50 days only i.e., 15-4-2023 to 5-6-2023.

CHAPTER - IV

DATA ANALYSIS AND INTERPRETATION

4.1 PREDICTING THE CLOSING STOCK PRICE OF THE EICHER MOTORS LIMITED

4.1.1. DATA COLLECTION

```
import yfinance as yf
# stock symbol for EICHERMOT
symbol = 'EICHERMOT.NS'

# Fetching the stock data using yfinance
EICHERMOT_data = yf.download(symbol, start='2019-01-01', end='2023-05-01')
df=EICHERMOT_data
# Printing the stock data
print(EICHERMOT_data )
```

```
[*********************100%***********************]  1 of 1 completed
                   Open         High          Low        Close    Adj Close  \
Date
2019-01-01  2316.225098  2329.300049  2290.500000  2319.510010  2083.264893
2019-01-02  2234.000000  2234.000000  2095.024902  2116.229980  1900.689087
2019-01-03  2104.000000  2104.000000  2015.099976  2021.145020  1815.288696
2019-01-04  2032.500000  2064.804932  2000.099976  2010.359985  1805.602173
2019-01-07  2045.000000  2047.699951  1975.500000  1983.040039  1781.064819
...                ...          ...          ...          ...          ...
2023-04-24  3204.000000  3217.449951  3158.100098  3182.800049  3182.800049
2023-04-25  3175.000000  3246.500000  3156.149902  3217.949951  3217.949951
2023-04-26  3230.000000  3280.000000  3225.550049  3248.350098  3248.350098
2023-04-27  3248.300049  3270.399902  3236.250000  3263.550049  3263.550049
2023-04-28  3275.000000  3305.000000  3265.350098  3301.100098  3301.100098

             Volume
Date
2019-01-01   444340
2019-01-02  4814970
2019-01-03  4683010
2019-01-04  3418330
2019-01-07  2156680
...             ...
2023-04-24   379317
2023-04-25   574542
2023-04-26   881684
2023-04-27   464406
2023-04-28   416242

[1069 rows x 6 columns]
```

4.1.2. DATA PROCESSING

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Loading the EICHERMOT stock price dataset
df = (EICHERMOT_data)

# Displaying the first few rows of the dataset
print(df.head())
```

```
                  Open        High         Low       Close    Adj Close  \
Date
2019-01-01  2316.225098  2329.300049  2290.500000  2319.510010  2083.264893
2019-01-02  2234.000000  2234.000000  2095.024902  2116.229980  1900.689087
2019-01-03  2104.000000  2104.000000  2015.099976  2021.145020  1815.288696
2019-01-04  2032.500000  2064.804932  2000.099976  2010.359985  1805.602173
2019-01-07  2045.000000  2047.699951  1975.500000  1983.040039  1781.064819

            Volume
Date
2019-01-01   444340
2019-01-02  4814970
2019-01-03  4683010
2019-01-04  3418330
2019-01-07  2156680
```

4.1.3. MOVING AVERAGES

```
import pandas as pd
import matplotlib.pyplot as plt

Date = pd.date_range(start='2019', end='2023', freq='Y')

# Calculate moving averages
df['MA10'] = df['Close'].rolling(window=10).mean()
df['MA20'] = df['Close'].rolling(window=20).mean()
df['MA50'] = df['Close'].rolling(window=50).mean()

# Plot moving averages
plt.figure(figsize=(16, 6))
plt.plot(df.index, df['Close'], label='Close')
plt.plot(df.index, df['MA10'], label='MA10')
plt.plot(df.index, df['MA20'], label='MA20')
plt.plot(df.index, df['MA50'], label='MA50')
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('Moving Averages')
plt.legend()
plt.grid(True)
plt.show()
```

Output:

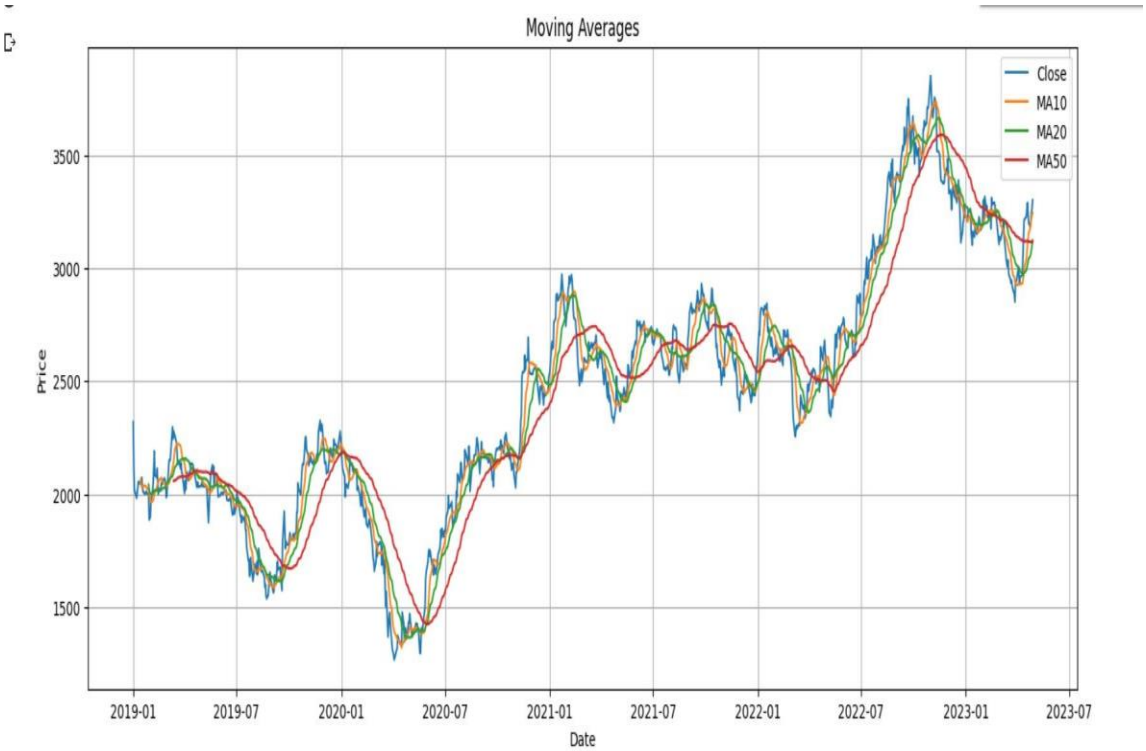


Figure: 4.1.3.1-Moving Averages_Eicher Motors For Eicher Motors

INTERPRETATION: The above chart represents the average closing price of the stock over the past 10, 20, and 50 trading days. If the short-term moving average (MA10) crosses above the longer-term moving average (MA50) hence it defined as a "Golden Cross," and which is considered as a "Bullish signal" is given at 2019 8th month 2020 6th month 2022 5th month and 2023 7th month given bullish trend. If MA 10 orange is crossed below MA 50 line red color then is said to be "death cross" it is considered as bearish trend. The above chart shows the 2020 01th month, 2021 2nd month and 2022 10th month it death crossed is formed hence that period stock went to bearish.

Above chart as per Moving Average trend shows that it is formed golden cross i.e., bullish trend. It is suggested that we can buy the stock until it formed death cross (or) when death cross is act like stock loss for this stock.

4.1.4. PERFORMING EDA

```
# statistical Summary

print(df.describe())
```

```
              Open        High         Low       Close    Adj Close  \
count  1069.000000  1069.000000  1069.000000  1069.000000  1069.000000
mean   2455.762724  2491.571234  2418.727342  2453.656653  2407.630848
std     552.115369   554.649036   548.125675   551.570844   579.998799
min    1296.619995  1309.989990  1245.010010  1268.005005  1251.827637
25%    2052.449951  2078.084961  2023.000000  2048.955078  1894.285278
50%    2479.000000  2508.149902  2440.000000  2480.500000  2454.875977
75%    2770.149902  2818.000000  2738.350098  2778.600098  2754.120850
max    3889.649902  3889.649902  3758.000000  3850.449951  3850.449951

             Volume        MA10        MA20        MA50
count  1.069000e+03  1060.000000  1050.000000  1020.000000
mean   1.271124e+06  2451.813850  2450.423348  2447.894192
std    1.012345e+06   547.196255   543.181704   533.543210
min    6.372900e+04  1321.943005  1364.984247  1424.915999
25%    6.152200e+05  2058.926389  2046.759514  2059.674063
50%    9.819000e+05  2481.424988  2498.604993  2524.906506
75%    1.604480e+06  2765.926245  2733.345001  2709.234500
max    1.149130e+07  3737.534937  3664.087463  3590.382998
```

```
df.info
```

```
<bound method DataFrame.info of                   Open        High         Low       Close    Adj Close  \
Date
2019-01-01  2316.225098  2329.300049  2290.500000  2319.510010  2083.264893
2019-01-02  2234.000000  2234.000000  2095.024902  2116.229980  1900.689087
2019-01-03  2104.000000  2104.000000  2015.099976  2021.145020  1815.288696
2019-01-04  2032.500000  2064.804932  2000.099976  2010.359985  1805.602173
2019-01-07  2045.000000  2047.699951  1975.500000  1983.040039  1781.064819
...                ...          ...          ...          ...          ...
2023-04-24  3204.000000  3217.449951  3158.100098  3182.800049  3182.800049
2023-04-25  3175.000000  3246.500000  3156.149902  3217.949951  3217.949951
2023-04-26  3230.000000  3280.000000  3225.550049  3248.350098  3248.350098
2023-04-27  3248.300049  3270.399902  3236.250000  3263.550049  3263.550049
2023-04-28  3275.000000  3305.000000  3265.350098  3301.100098  3301.100098

             Volume         MA10         MA20         MA50
Date
2019-01-01   444340          NaN          NaN          NaN
2019-01-02  4814970          NaN          NaN          NaN
2019-01-03  4683010          NaN          NaN          NaN
2019-01-04  3418330          NaN          NaN          NaN
2019-01-07  2156680          NaN          NaN          NaN
...             ...          ...          ...          ...
2023-04-24   379317  3177.865015  3054.137512  3116.528994
2023-04-25   574542  3202.615015  3067.707507  3115.393994
2023-04-26   881684  3222.030029  3083.740015  3115.800996
2023-04-27   464406  3235.550024  3101.437512  3116.328999
2023-04-28   416242  3244.355029  3122.215015  3118.103999

[1069 rows x 9 columns]>
```

```
df.shape
```

```
(1069, 9)
```

Checking for any null or NaN values in the data

```
df.isnull().sum()
```

```
Open          0
High          0
Low           0
Close         0
Adj Close     0
Volume        0
MA10          9
MA20         19
MA50         49
dtype: int64
```

4.1.5 VISUALIZING THE DATA:

```
# Correlation heatmap of the dataset
plt.figure(figsize=(12, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

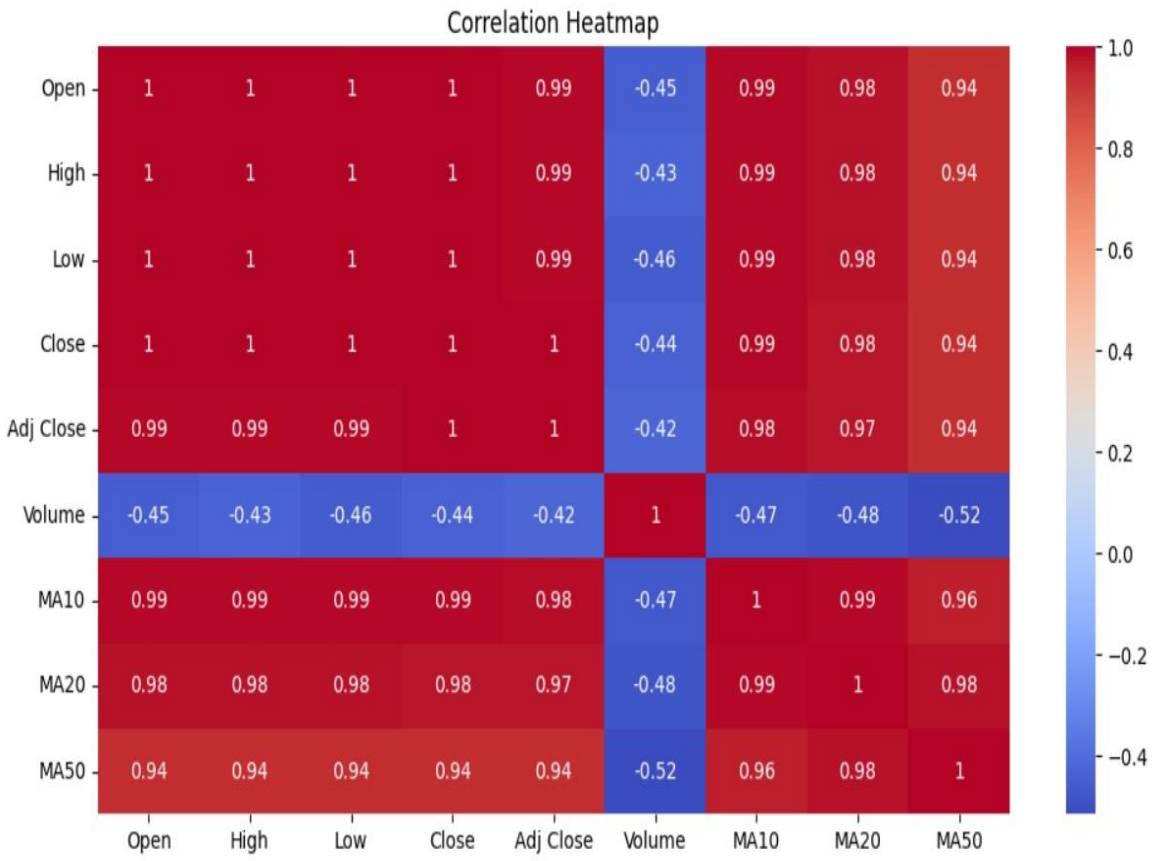


Figure: 4.1.5.1-Correlation between Features for Eicher Motors

<u>INTERPRETATION</u>: The correlation matrix reveals strong positive correlations among opening, high, low, and closing prices. Moving averages (MA10, MA20, and MA50) are positively correlated, indicating similar trend-following behavior. Volume and average volume show moderate negative correlations with other indicators, potentially indicating an inverse relationship with price movement.

4.1.6. FEATURE ENGINEERING

```python
# Volume-based features
df['Volume'] = EICHERMOT_data ['Volume']  # Add the raw volume data
df['AvgVolume'] = df['Volume'].rolling(window=14).mean()  # Average trading volume

# MACD calculation
def calculate_macd(data, short_window=12, long_window=26, signal_window=9):
    exp_short = data['Close'].ewm(span=short_window, adjust=False).mean()
    exp_long = data['Close'].ewm(span=long_window, adjust=False).mean()
    macd_line = exp_short - exp_long
    signal_line = macd_line.ewm(span=signal_window, adjust=False).mean()
    histogram = macd_line - signal_line

    return macd_line, signal_line, histogram

# Calculate MACD and add as features
macd_line, signal_line, histogram = calculate_macd(df)
df['MACD_Line'] = macd_line
df['MACD_Signal'] = signal_line
df['MACD_Histogram'] = histogram
```

```python
# Calculating the Relative Strength Index (RSI)
def calculate_rsi(data, window=14):
    delta = data['Close'].diff()
    gain = delta.where(delta > 0, 0)
    loss = -delta.where(delta < 0, 0)

    avg_gain = gain.rolling(window).mean()
    avg_loss = loss.rolling(window).mean()

    rs = avg_gain / avg_loss
    rsi = 100 - (100 / (1 + rs))

    return rsi

# Adding RSI as a feature
df['RSI'] = calculate_rsi(df)

# Calculating the 50-day and 200-day moving averages
df['MA10'] = df['Close'].rolling(window=10).mean()
df['MA20'] = df['Close'].rolling(window=20).mean()

# Creating a signal column initialized with zeros
df['Signal'] = 0

# Generating buy signals when the 50-day moving average crosses above the 200-day moving average
df.loc[df['MA10'] > df['MA20'], 'Signal'] = 1

# Generating sell signals when the 50-day moving average crosses below the 200-day moving average
df.loc[df['MA10'] < df['MA20'], 'Signal'] = -1
```

```
df.head()
```

	Open	High	Low	Close	Adj Close	Volume	MA10	MA20	MA50	AvgVolume	MACD_Line	MACD_Signal	MACD_Histogram	RSI	Signal
Date															
2019-01-01	2316.225098	2329.300049	2290.500000	2319.510010	2083.264893	444340	NaN	NaN	NaN	NaN	0.000000	0.000000	0.000000	NaN	0
2019-01-02	2234.000000	2234.000000	2095.024902	2116.229980	1900.689087	4814970	NaN	NaN	NaN	NaN	-16.216071	-3.243214	-12.972857	NaN	0
2019-01-03	2104.000000	2104.000000	2015.099976	2021.145020	1815.288696	4683010	NaN	NaN	NaN	NaN	-36.321296	-9.858831	-26.462465	NaN	0
2019-01-04	2032.500000	2064.804932	2000.099976	2010.359985	1805.602173	3418330	NaN	NaN	NaN	NaN	-52.519680	-18.391000	-34.128679	NaN	0
2019-01-07	2045.000000	2047.699951	1975.500000	1983.040039	1781.064819	2156680	NaN	NaN	NaN	NaN	-66.791575	-28.071115	-38.720460	NaN	0

HANDLING MISSING VALUES

```
[ ] # Checking for missing values in the dataset
    print(df.isnull().sum())
```

```
Open              0
High              0
Low               0
Close             0
Adj Close         0
Volume            0
MA10              9
MA20             19
MA50             49
AvgVolume        13
MACD_Line         0
MACD_Signal       0
MACD_Histogram    0
RSI              13
Signal            0
dtype: int64
```

```
[ ] # Droping the rows with missing values
    df = df.dropna()
```

```
# Verifying that missing values have been removed
print(df.isnull().sum())
```

```
Open              0
High              0
Low               0
Close             0
Adj Close         0
Volume            0
MA10              0
MA20              0
MA50              0
AvgVolume         0
MACD_Line         0
MACD_Signal       0
MACD_Histogram    0
RSI               0
Signal            0
dtype: int64
```

```
[ ] df.columns
```

```
Index(['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume', 'MA10', 'MA20',
       'MA50', 'AvgVolume', 'MACD_Line', 'MACD_Signal', 'MACD_Histogram',
       'RSI', 'Signal'],
      dtype='object')
```

4.1.7. SPLITTING THE DATA

```
[ ] from sklearn.model_selection import train_test_split

    # Defining the features and target variable
    features = ['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume',  'AvgVolume',
           'MACD_Line', 'MACD_Signal', 'MACD_Histogram', 'RSI', 'MA10', 'MA20',
           'Signal']
    target = 'Close'

    # Spliting the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(df[features], df[target], test_size=0.2, random_state=42)

    # Printing the shapes of the training and testing sets
    print("Training set shape:", X_train.shape, y_train.shape)
    print("Testing set shape:", X_test.shape, y_test.shape)
```

```
Training set shape: (816, 14) (816,)
Testing set shape: (204, 14) (204,)
```

4.1.8. MODEL SELECTION AND TRAINING

```
[ ] from sklearn.linear_model import LinearRegression
    from sklearn.metrics import mean_squared_error

    # Create an instance of the Linear Regression model
    model = LinearRegression()

    # Fit the model on the training data
    model.fit(X_train, y_train)

    # Make predictions on the training and testing data
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    # Evaluate the model using mean squared error
    train_mse = mean_squared_error(y_train, y_train_pred)
    test_mse = mean_squared_error(y_test, y_test_pred)


    # Print the mean squared error
    print("Train MSE:", train_mse)
    print("Test MSE:", test_mse)
```

```
Train MSE: 2.858638881617081e-24
Test MSE: 2.4746993509743613e-24
```

INTERPRETATION: The train and test mean squared error (MSE) values, both close to zero, indicate that the model's predictions are extremely accurate. The low MSE values reflect a strong predictive performance, indicating that the model can accurately predict the stock price

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 6))
# Scatter plot for training set
plt.scatter(y_train, y_train_pred, color='blue', label='Training data')
# Scatter plot for testing set
plt.scatter(y_test, y_test_pred, color='red', label='Testing data')

# Plot the ideal line (y = x)
plt.plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()], color='green', linestyle='--')

plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Linear Regression - Predicted vs. Actual Values')
plt.legend()
plt.show()
```

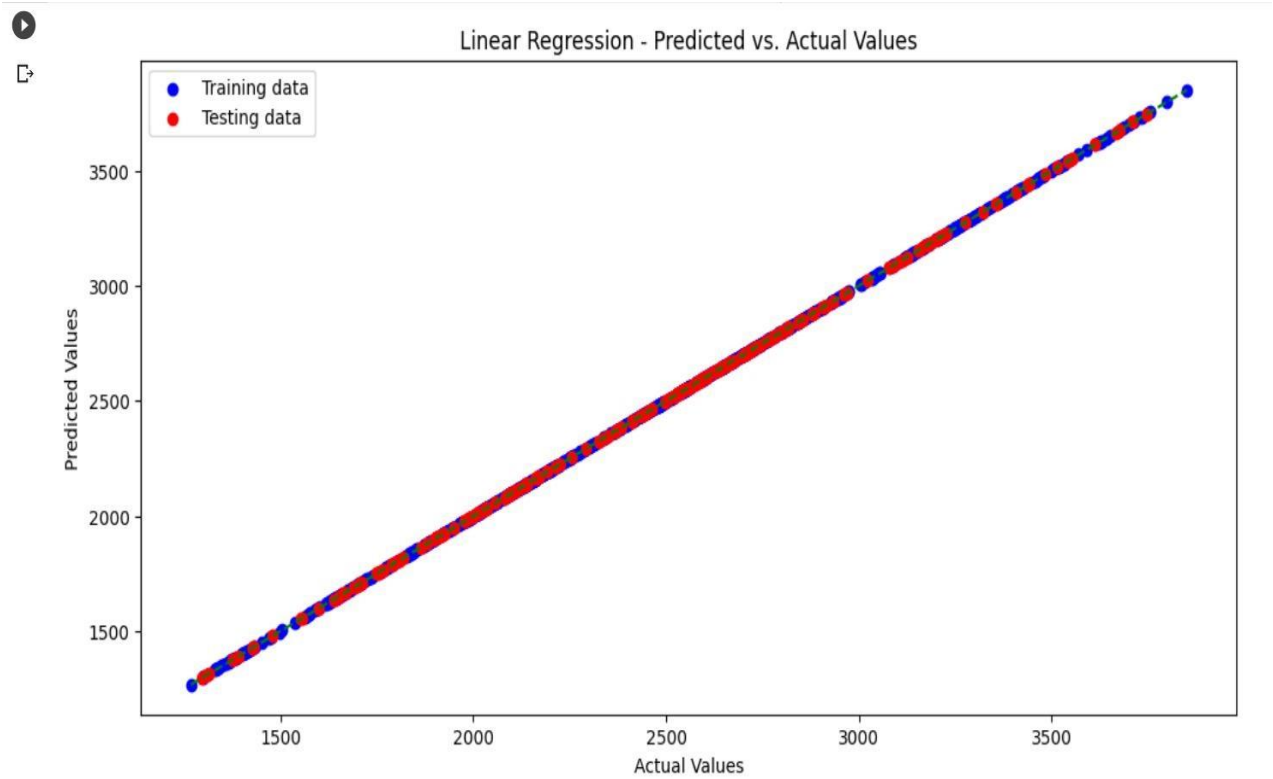


Figure: 4.1.8.1-Linear Regression- Predicted Vs Actual Values For Eicher Motors

INTERPRETATION: The above graph shows that the actual train and test data points vs predicted stock prices are aligned in the same line, this shows that the model has predicted approximately similar values.

4.1.9. MODEL EVALUATION:

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Calculate mean absolute error (MAE)
train_mae = mean_absolute_error(y_train, y_train_pred)
test_mae = mean_absolute_error(y_test, y_test_pred)

# Calculate root mean squared error (RMSE)
train_rmse = mean_squared_error(y_train, y_train_pred, squared=False)
test_rmse = mean_squared_error(y_test, y_test_pred, squared=False)

# Calculate coefficient of determination (R-squared)
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)

# Print the evaluation metrics
print("Mean Absolute Error (MAE):")
print("  Train:", train_mae)
print("  Test:", test_mae)
print("\nRoot Mean Squared Error (RMSE):")
print("  Train:", train_rmse)
print("  Test:", test_rmse)
print("\nCoefficient of Determination (R-squared):")
print("  Train:", train_r2)
print("  Test:", test_r2)
```

```
Mean Absolute Error (MAE):
  Train: 1.2617009833418563e-12
  Test: 1.2115450255234963e-12

Root Mean Squared Error (RMSE):
  Train: 1.6907509815514173e-12
  Test: 1.5731177168204422e-12

Coefficient of Determination (R-squared):
  Train: 1.0
  Test: 1.0
```

INTERPRETATION: Excellent model performance is indicated by the evaluation metrics (MAE, RMSE, R-squared) supplied. The model's forecasts for the stock price appear to be amazingly accurate, with little variation from the actual values, according to the exceptionally low errors (MAE and RMSE). Additionally, the model's perfect R-squared value shows that it completely and accurately predicts the stock price by explaining all the variability in the target variable.

```python
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Initialize the models
linear_model = LinearRegression()

# Fit the models
linear_model.fit(X_train, y_train)

# Make predictions
linear_pred = linear_model.predict(X_test)


# Calculate evaluation metrics
linear_mae = mean_absolute_error(y_test, linear_pred)
linear_rmse = mean_squared_error(y_test, linear_pred, squared=False)
linear_r2 = r2_score(y_test, linear_pred)


# Print the evaluation metrics
print("Linear Regression:")
print("MAE:", linear_mae)
print("RMSE:", linear_rmse)
print("R-squared:", linear_r2)
print()
```

```
Linear Regression:
MAE: 1.2115450255234963e-12
RMSE: 1.5731177168204422e-12
R-squared: 1.0
```

INTERPRETATION: Excellent performance is indicated by the linear regression model's evaluation metrics (MAE, RMSE, R-squared). The model's forecasts for the stock price appear to be quite accurate, with little variation from the actual values, according to the exceptionally low MAE and RMSE values. Additionally, the model's perfect R-squared value of 1.0 shows that it completely accounts for the fluctuation in the target variable, suggesting it is capable to the accurately forecasting the stock prices.

4.1.10. MODEL OPTIMIZATION:

```
[ ] from sklearn.model_selection import GridSearchCV
    from sklearn.linear_model import LinearRegression

    # Initializing the Linear Regression model
    model = LinearRegression()

    # Defining the hyperparameters to tune
    param_grid = {
        'fit_intercept': [True, False],
    }

    # Performing grid search with cross-validation
    grid_search = GridSearchCV(model, param_grid, cv=5, scoring='neg_mean_squared_error')
    grid_search.fit(X_train, y_train)

    # Getting the best hyperparameters and model
    best_params = grid_search.best_params_
    best_model = grid_search.best_estimator_

    # Printing the best hyperparameters
    print("Best Hyperparameters:", best_params)
```

```
Best Hyperparameters: {'fit_intercept': True}
```

INTERPRETATION: The best hyper parameter for the linear regression model is 'fit_intercept' set to True, indicating that the model includes an intercept term in the regression equation. This implies that the model considers a baseline value or starting point for the predicted stock prices, improving its ability to capture the relationship between the features and the target variable.

▾ Training a new instance of the linear regression model with the best hyperparameters

```
[ ] import numpy as np
    from sklearn.linear_model import LinearRegression
    from sklearn.metrics import mean_squared_error, r2_score

    # Initialize the model with the best hyperparameters
    model = LinearRegression(fit_intercept=best_params['fit_intercept'])

    # Train the model on the entire training data
    model.fit(X_train, y_train)

    # Make predictions on the testing data
    y_pred = model.predict(X_test)

    # Evaluate the model on the testing data
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    # Print the evaluation metrics for the trained model
    print("Root Mean Squared Error (RMSE):", rmse)
    print("Coefficient of Determination (R-squared):", r2)
```

```
Root Mean Squared Error (RMSE): 1.573117716820442 2e-12
Coefficient of Determination (R-squared): 1.0
```

INTERPRETATION: The model's anticipated stock prices appear to closely match the actual values, as indicated by the exceptionally low prediction error value of 1.5731177168204422e-12, which shows very little variance. The model's outstanding accuracy and capacity to predict stock prices with complete precision are demonstrated by the perfect R-squared value of 1.0, which indicates that the model fully accounts for all the variability in the target variable.

4.1.11. DIFFERENCE BETWEEN ACTUAL AND PREDICTED

```
from sklearn.linear_model import LinearRegression

# Create an instance of Linear Regression model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)
plt.figure(figsize=(12, 6))
plt.plot(df.index[-len(y_test):], y_test, label='Actual', color='blue')
plt.plot(df.index[-len(y_test):], y_pred, label='Predicted', color='orange')
plt.scatter(df.index[-len(y_test):], y_test, color='blue')
plt.xlabel('Date')
plt.legend()
plt.show()
```

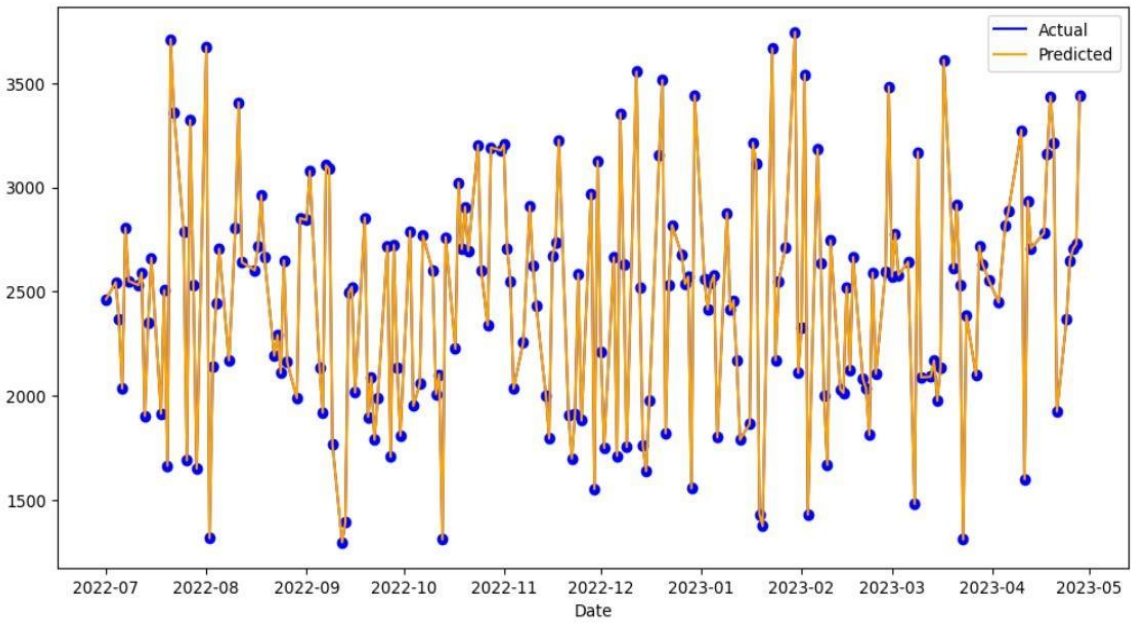


Figure: 4.1.11.1-Difference Between Actual Vs Predicted Values For Eicher Motors

INTERPRETATION: The above line chart represents the actual and predicted stock prices. The predicted values closely follow the actual values it indicates that the model's predictions align well with the actual stock prices. This suggests that the linear regression model is able to capture the overall patterns and fluctuations in the stock prices.

4.1.12.PREDICTION VALUES

```
import pandas as pd
import numpy as np
from pandas.tseries.offsets import BDay

# Defining the number of future periods to predict(excluding weekends)
num_periods = 60

#getting the latest available trading day
latest_trading_day =df.index[-1].date()

#caculating the number of additional trading days required
num_additional_days = num_periods + (num_periods // 5 * 2) #add extra days for weekends

#generating future trading days
future_trading_days = pd.bdate_range(start=latest_trading_day + BDay(1), periods=num_additional_days)

#filtering out weekends from the trading days
future_trading_days = future_trading_days[future_trading_days.dayofweek < 5][:num_periods]

# Creating a new dataframe to store the future predictions
future_predictions = pd.DataFrame(index=future_trading_days, columns=['Predicted Price'])

# Using the trained linear regression model to predict future prices
future_features = df[features].tail(num_periods)  # Use the latest available features as input

future_predictions['Predicted Price'] = model.predict(future_features)

# Printing the future predictions
print(future_predictions.head())
```

```
            Predicted Price
2023-05-01      3191.449951
2023-05-02      3263.350098
2023-05-03      3303.100098
2023-05-04      3217.300049
2023-05-05      3316.699951
```

4.1.13. ACTUAL VS. PREDICTED PRICE MOMENTS

```
import matplotlib.pyplot as plt

# Plot the actual and predicted values
plt.figure(figsize=(14, 6))
plt.plot(df.index, df['Close'], label='Actual')
plt.plot(future_predictions.index, future_predictions['Predicted Price'], label='Predicted')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.title('Actual vs. Predicted Stock Prices')
plt.legend()
plt.show()
```

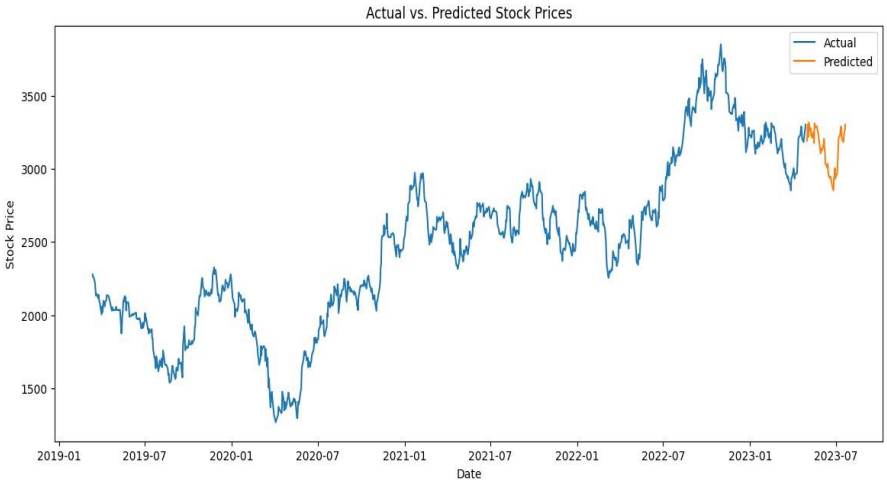


Figure: 4.1.13.1-Actual vs Predicted Price Moments for Eicher Motors

INTERPRETATION: The above chart represents the past 3 years of stock prices and future 60 days of predicted stock prices. The predicted prices show a generally increasing trend, as indicated by the positive price differences.

4.1.14. DIFFERENCE BETWEEN ACTUAL AND PREDICTED PRICES

```
# stock symbol for HDFCBANK
symbol = 'EICHERMOT.NS'

# Fetching the stock data using yfinance
EICHERMOT = yf.download(symbol, start='2023-05-03', end='2023-07-08')

# Printing the stock data
print(EICHERMOT.head())

# Merging the predicted prices with the actual stock data
merged_data = pd.concat([EICHERMOT['Close'], future_predictions], axis=1)
merged_data.columns = ['Actual Price', 'Predicted Price']

# Calculating the difference between the actual and predicted prices
merged_data['Price Difference'] = merged_data['Actual Price'] - merged_data['Predicted Price']

# Displaying the table
print(merged_data)
```

Output:

```
[*********************100%***********************]  1 of 1 completed
                   Open         High          Low        Close    Adj Close  \
Date
2023-05-03  3367.899902  3367.899902  3321.199951  3348.350098  3348.350098
2023-05-04  3341.000000  3382.449951  3321.000000  3339.300049  3339.300049
2023-05-05  3339.300049  3392.050049  3324.750000  3355.100098  3355.100098
2023-05-08  3371.000000  3407.399902  3360.100098  3392.899902  3392.899902
2023-05-09  3410.000000  3410.000000  3371.350098  3381.899902  3381.899902

            Volume
Date
2023-05-03  577763
2023-05-04  325535
2023-05-05  518154
2023-05-08  322909
2023-05-09  347387
            Actual Price Predicted Price Price Difference
2023-05-01          NaN      3191.449951               NaN
2023-05-02          NaN      3263.350098               NaN
2023-05-03   3348.350098      3303.100098         45.250000
2023-05-04   3339.300049      3217.300049        122.000000
2023-05-05   3355.100098      3316.699951         38.400146
2023-05-08   3392.899902      3260.699951        132.199951
2023-05-09   3381.899902      3274.699951        107.199951
```

2023-05-10	3416.750000	3228.000000	188.750000
2023-05-11	3405.300049	3237.149902	168.150146
2023-05-12	3626.350098	3212.350098	414.000000
2023-05-15	3658.500000	3252.649902	405.850098
2023-05-16	3633.199951	3175.699951	457.500000
2023-05-17	3628.750000	3311.149902	317.600098
2023-05-18	3576.149902	3289.100098	287.049805
2023-05-19	3570.899902	3285.000000	285.899902
2023-05-22	3545.449951	3290.500000	254.949951
2023-05-23	3601.600098	3274.149902	327.450195
2023-05-24	3610.899902	3249.699951	361.199951
2023-05-25	3651.500000	3241.949951	409.550049
2023-05-26	3690.350098	3205.600098	484.750000
2023-05-29	3711.449951	3131.050049	580.399902
2023-05-30	3678.350098	3105.899902	572.450195
2023-05-31	3671.949951	3138.800049	533.149902
2023-06-01	3716.050049	3125.500000	590.550049
2023-06-02	3699.399902	3127.300049	572.099854
2023-06-05	3673.250000	3166.199951	507.050049
2023-06-06	3705.100098	3204.399902	500.700195
2023-06-07	3727.399902	3146.699951	580.699951
2023-06-08	3657.399902	3116.100098	541.299805
2023-06-09	3581.300049	3033.399902	547.900146
2023-06-12	3590.600098	3016.899902	573.700195
2023-06-13	3592.649902	3006.899902	585.750000
2023-06-14	3572.050049	3035.449951	536.600098
2023-06-15	3534.699951	2974.600098	560.099854
2023-06-16	3538.899902	2952.000000	586.899902
2023-06-19	3496.649902	2934.800049	561.849854
2023-06-20	3562.100098	2946.550049	615.550049
2023-06-21	3570.449951	2927.699951	642.750000
2023-06-22	3558.600098	2909.600098	649.000000
2023-06-23	3539.199951	2885.550049	653.649902
2023-06-26	3551.649902	2852.399902	699.250000
2023-06-27	3535.750000	2933.699951	602.050049
2023-06-28	3535.750000	2948.850098	586.899902
2023-06-29	NaN	3004.300049	NaN
2023-06-30	3580.100098	2933.750000	646.350098
2023-07-03	3630.850098	2961.699951	669.150146
2023-07-04	3401.800049	2970.449951	431.350098
2023-07-05	3309.750000	3054.199951	255.550049
2023-07-06	3222.199951	3128.350098	93.849854
2023-07-07	3184.800049	3213.050049	-28.250000
2023-07-10	NaN	3229.000000	NaN
2023-07-11	NaN	3263.550049	NaN
2023-07-12	NaN	3289.600098	NaN
2023-07-13	NaN	3246.949951	NaN
2023-07-14	NaN	3200.699951	NaN
2023-07-17	NaN	3182.800049	NaN
2023-07-18	NaN	3217.949951	NaN
2023-07-19	NaN	3248.350098	NaN
023-07-20	NaN	3263.550049	NaN
2023-07-21	NaN	3301.100098	NaN

4.2 PREDICTING THE CLOSING STOCK PRICE OF THE HERO MOTOCORP LIMITED

4.2.1. DATA COLLECTION:

```
import yfinance as yf
# stock symbol for HEROMOTOCO
symbol = 'HEROMOTOCO.NS'

# Fetching the stock data using yfinance
HEROMOTOCO_data = yf.download(symbol, start='2019-01-01', end='2023-05-01')
df=HEROMOTOCO_data
# Printing the stock data
print(HEROMOTOCO_data )
```

```
[*********************100%***********************]  1 of 1 completed
                  Open         High          Low        Close    Adj Close  \
Date
2019-01-01  3089.000000  3135.000000  3073.300049  3127.600098  2667.811035
2019-01-02  3115.000000  3117.300049  3023.800049  3046.550049  2598.676514
2019-01-03  3032.000000  3078.000000  3006.300049  3014.649902  2571.466064
2019-01-04  3030.000000  3045.000000  2972.050049  2987.850098  2548.605713
2019-01-07  3020.000000  3023.000000  2952.000000  2957.949951  2523.101318
...                 ...          ...          ...          ...          ...
2023-04-24  2459.949951  2524.000000  2443.399902  2501.050049  2501.050049
2023-04-25  2504.899902  2534.000000  2493.850098  2497.750000  2497.750000
2023-04-26  2510.000000  2524.000000  2492.050049  2505.149902  2505.149902
2023-04-27  2500.000000  2526.850098  2497.050049  2508.649902  2508.649902
2023-04-28  2520.000000  2567.000000  2515.149902  2558.600098  2558.600098

            Volume
Date
2019-01-01  344239
2019-01-02  597712
2019-01-03  368413
2019-01-04  405667
2019-01-07  361881
...            ...
2023-04-24  686216
2023-04-25  538138
2023-04-26  560280
2023-04-27  685268
2023-04-28  721370

[1069 rows x 6 columns]
```

4.2.2. Moving Averages:

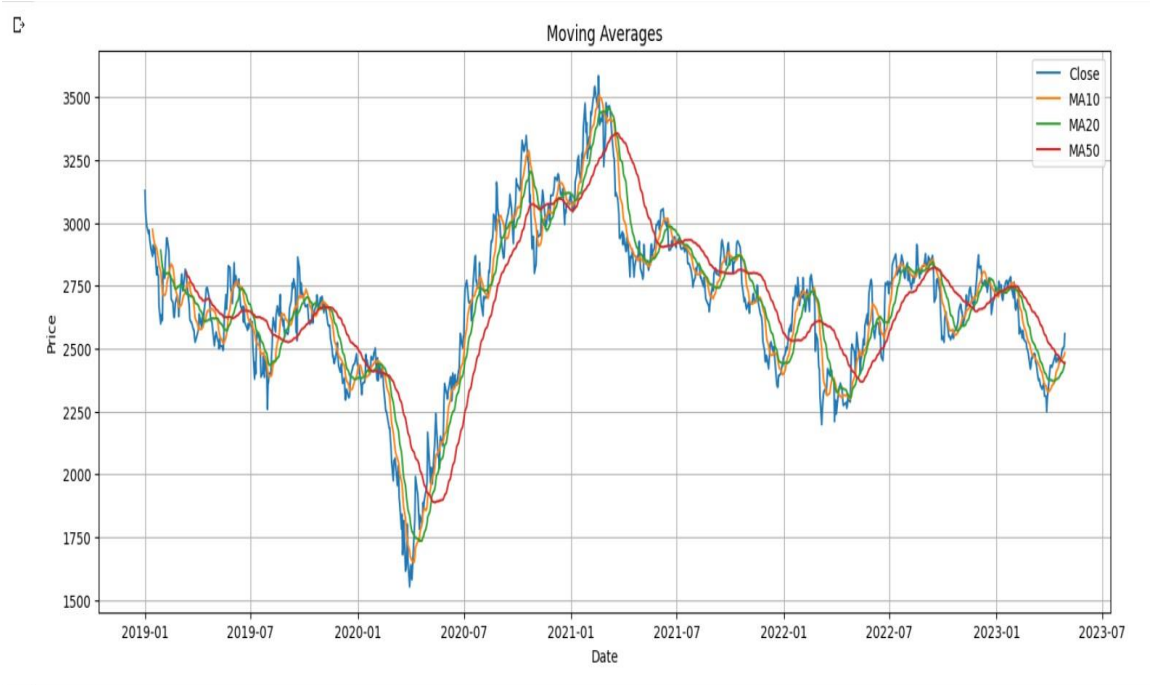


Figure: 4.2.2.1- Moving Averages for HERO Motors

INTERPRETATION: The stock price falls below the MA50, it suggests a potential bearish trend in the long term, in this chart short-term moving average (MA10) crosses below the longer-term moving (MA50) so it indicates the "Death cross". And it considered as a "Bearish signal" is given at 2019 8th month 2020 10th month 2021 11st month 2022 4th, 10th month and 2023 5th month given bearish trend. If Ma10 orange is crossed above MA 50 line red color then is said to be "golden cross" it is considered as bullish trend. The above chart shows the 2020 5th month, 2021 1st month and 2022 12th month is golden cross is formed hence that period stock went to bullish.

Above chart as per Moving Average trend shows that it is formed death cross i.e., bearish trend. It is suggested that we can sell the stock until it formed golden cross (or) when golden cross is act like stock profit for this stock.

4.2.3. MODEL EVALUATION

```python
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Initialize the models
linear_model = LinearRegression()

# Fit the models
linear_model.fit(X_train, y_train)

# Make predictions
linear_pred = linear_model.predict(X_test)


# Calculate evaluation metrics
linear_mae = mean_absolute_error(y_test, linear_pred)
linear_rmse = mean_squared_error(y_test, linear_pred, squared=False)
linear_r2 = r2_score(y_test, linear_pred)


# Print the evaluation metrics
print("Linear Regression:")
print("MAE:", linear_mae)
print("RMSE:", linear_rmse)
print("R-squared:", linear_r2)
print()
```

```
Linear Regression:
MAE: 3.313971319585107e-11
RMSE: 4.672836545819869e-11
R-squared: 1.0
```

INTERPRETATION: The linear regression model's MAE and RMSE values, close to zero, indicate precise and accurate predictions. The R-squared value of 1.0 suggests that the model perfectly explains the variability in the target variable, reflecting an excellent fit to the data.

4.2.4.Predictions

```
import pandas as pd
import numpy as np
from pandas.tseries.offsets import BDay

# Defining the number of future periods to predict(excluding weekends)
num_periods = 60

#getting the latest available trading day
latest_trading_day =df.index[-1].date()

#caculating the number of additional trading days required
num_additional_days = num_periods + (num_periods // 5 * 2) #add extra days for weekends

#generating future trading days
future_trading_days = pd.bdate_range(start=latest_trading_day + BDay(1), periods=num_additional_days)

#filtering out weekends from the trading days
future_trading_days = future_trading_days[future_trading_days.dayofweek < 5][:num_periods]

# Creating a new dataframe to store the future predictions
future_predictions = pd.DataFrame(index=future_trading_days, columns=['Predicted Price'])

# Using the trained linear regression model to predict future prices
future_features = df[features].tail(num_periods)  # Use the latest available features as input

future_predictions['Predicted Price'] = model.predict(future_features)

# Printing the future predictions
print(future_predictions.head())
```

```
            Predicted Price
2023-05-01      2702.300049
2023-05-02      2763.250000
2023-05-03      2727.250000
2023-05-04      2660.949951
2023-05-05      2654.250000
```

4.2.5. ACTUAL VS. PREDICTED PRICE MOMENTS

```
[ ] import matplotlib.pyplot as plt

    # Plot the actual and predicted values
    plt.figure(figsize=(14, 6))
    plt.plot(df.index, df['Close'], label='Actual')
    plt.plot(future_predictions.index, future_predictions['Predicted Price'], label='Predicted')
    plt.xlabel('Date')
    plt.ylabel('Stock Price')
    plt.title('Actual vs. Predicted Stock Prices')
    plt.legend()
    plt.show()
```

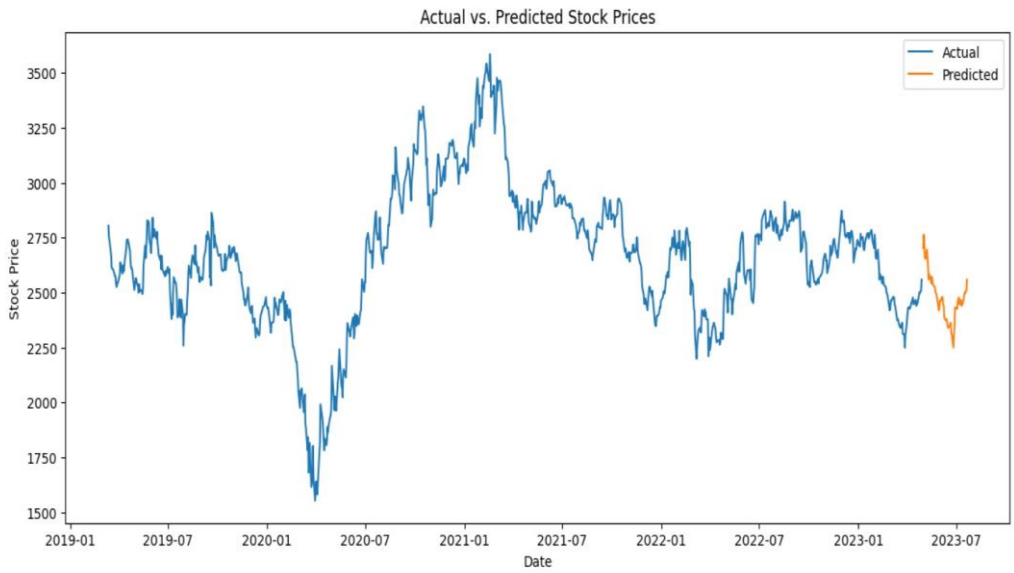


Figure: 4.2.5.1-Actual vs Predicted Price Moments for HERO Motors

INTERPRETATION: The above chart represents the past 3 years of stock prices and future 60 days of predicted stock prices. The predicted prices show a generally increasing trend, as indicated by the positive price differences.

4.2.6. DIFFERENCE BETWEEN ACTUAL AND PREDICTED PRICES

```
[ ] # stock symbol for HEROMOTOCO
    symbol = 'HEROMOTOCO.NS'

    # Fetching the stock data using yfinance
    HEROMOTOCO = yf.download(symbol, start='2023-05-03', end='2023-07-08')

    # Printing the stock data
    print(HEROMOTOCO.head())

    # Merging the predicted prices with the actual stock data
    merged_data = pd.concat([HEROMOTOCO['Close'], future_predictions], axis=1)
    merged_data.columns = ['Actual Price', 'Predicted Price']

    # Calculating the difference between the actual and predicted prices
    merged_data['Price Difference'] = merged_data['Actual Price'] - merged_data['Predicted Price']

    # Displaying the table
    print(merged_data)
```

```
[*********************100%***********************]  1 of 1 completed
                   Open     High          Low        Close    Adj Close  \
Date
2023-05-03  2490.000000  2518.00  2478.899902  2502.649902  2502.649902
2023-05-04  2507.949951  2532.75  2504.300049  2514.500000  2514.500000
2023-05-05  2537.000000  2562.00  2512.550049  2547.250000  2547.250000
2023-05-08  2566.000000  2592.00  2566.000000  2576.350098  2576.350098
2023-05-09  2587.449951  2604.00  2575.250000  2591.550049  2591.550049

             Volume
Date
2023-05-03   863935
2023-05-04   684323
2023-05-05  1527593
2023-05-08   855245
2023-05-09   556418
            Actual Price  Predicted Price  Price Difference
2023-05-01           NaN      2702.300049               NaN
2023-05-02           NaN      2763.250000               NaN
2023-05-03   2502.649902      2727.250000       -224.600098
2023-05-04   2514.500000      2660.949951       -146.449951
2023-05-05   2547.250000      2654.250000       -107.000000
2023-05-08   2576.350098      2695.000000       -118.649902
2023-05-09   2591.550049      2652.600098        -61.050049
2023-05-10   2586.050049      2614.649902        -28.599854
2023-05-11   2585.449951      2559.199951         26.250000
2023-05-12   2605.899902      2583.300049         22.599854
2023-05-15   2691.899902      2561.550049        130.349854
2023-05-16   2686.000000      2542.500000        143.500000
2023-05-17   2722.949951      2572.000000        150.949951
2023-05-18   2735.149902      2556.699951        178.449951
2023-05-19   2713.000000      2535.550049        177.449951
2023-05-22   2692.149902      2527.449951        164.699951
2023-05-23   2710.449951      2520.550049        189.899902
2023-05-24   2738.699951      2499.449951        239.250000
2023-05-25   2732.800049      2495.500000        237.300049
2023-05-26   2745.149902      2485.649902        259.500000
2023-05-29   2776.899902      2437.550049        339.349854
2023-05-30   2779.149902      2419.100098        360.049805
2023-05-31   2759.899902      2431.100098        328.799805
2023-06-01   2800.399902      2463.350098        337.049805
2023-06-02   2892.649902      2461.550049        431.099854
```

4.3 PREDICTING THE CLOSING STOCK PRICE OF THE TATAMOTORS LIMITED

4.3.1. DATA COLLECTION

```
import yfinance as yf
# stock symbol for TATAMOTORS
symbol = 'TATAMOTORS.NS'

# Fetching the stock data using yfinance
TATAMOTORS_data = yf.download(symbol, start='2019-01-01', end='2023-05-01')
df=TATAMOTORS_data
# Printing the stock data
print(TATAMOTORS_data )
```

```
[*********************100%***********************]  1 of 1 completed
                  Open        High         Low       Close   Adj Close  \
Date
2019-01-01  173.500000  174.199997  171.300003  173.449997  173.449997
2019-01-02  170.600006  171.750000  168.000000  168.550003  168.550003
2019-01-03  169.000000  170.149994  165.100006  166.050003  166.050003
2019-01-04  169.000000  171.949997  168.500000  170.949997  170.949997
2019-01-07  173.199997  178.199997  173.199997  175.350006  175.350006
...                ...         ...         ...         ...         ...
2023-04-24  471.200012  476.000000  468.299988  475.200012  475.200012
2023-04-25  475.200012  476.399994  471.899994  473.549988  473.549988
2023-04-26  473.000000  479.500000  472.600006  478.200012  478.200012
2023-04-27  478.500000  482.000000  476.500000  481.600006  481.600006
2023-04-28  481.399994  488.000000  478.350006  484.950012  484.950012

              Volume
Date
2019-01-01   6715543
2019-01-02  13410104
2019-01-03   9321842
2019-01-04  17650435
2019-01-07  21438351
...               ...
2023-04-24   8020593
2023-04-25   6888352
2023-04-26   7791425
2023-04-27   7085793
2023-04-28   9947813

[1069 rows x 6 columns]
```

4.3.2. MOVING AVERAGES

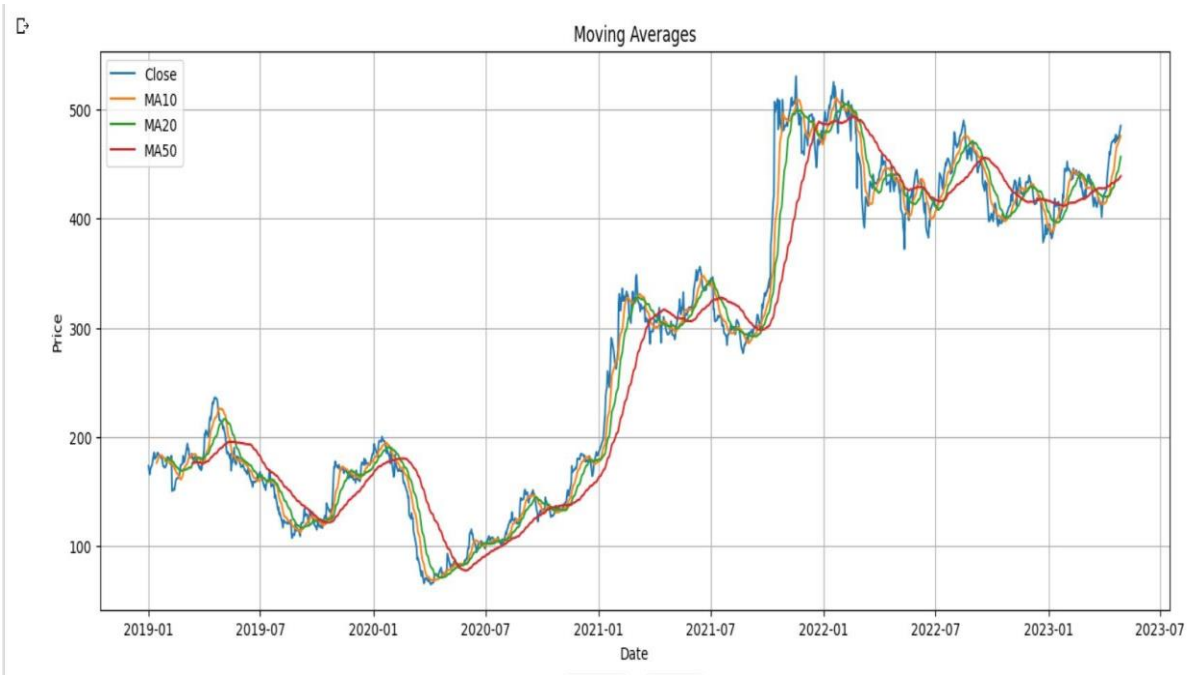


Figure: 4.3.2.1-Moving Averages for TATA Motors

INTERPRETATION: The above chart represents the average closing price of the stock over the past 10, 20, 50 trading days .The short-term moving average (MA10) crosses above the longer-term moving average (MA50) so it defined as a "Golden Cross," and which is considered as a "Bullish signal". Is given at 2019 5th month 2019 11th month 2020 9th month, 2021 3rd and 6th month, 2021 11th month and 2022 8th month and 2023 4th month given bullish trend. If MA10 orange is crossed below MA50 red colour line then is said to be “death cross” it is considered as bearish trend. The above chart shows the 2019 7th month, 2020 4th month, 2021 8th month 2022 3rd and 6th month 2022 10th month 2023 1st month it death cross is formed hence that period stock went to bearish.

Above chart as per Moving Average trend shows that it is formed golden cross i.e., bullish trend. It is suggested that we can buy the stock until it formed death cross (or) when death cross is act like stock loss for this stock.

4.3.4. Model evaluation

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Initialize the models
linear_model = LinearRegression()

# Fit the models
linear_model.fit(X_train, y_train)

# Make predictions
linear_pred = linear_model.predict(X_test)


# Calculate evaluation metrics
linear_mae = mean_absolute_error(y_test, linear_pred)
linear_rmse = mean_squared_error(y_test, linear_pred, squared=False)
linear_r2 = r2_score(y_test, linear_pred)


# Print the evaluation metrics
print("Linear Regression:")
print("MAE:", linear_mae)
print("RMSE:", linear_rmse)
print("R-squared:", linear_r2)
print()
```

```
Linear Regression:
MAE: 2.7773861641916856e-13
RMSE: 4.388783638747104e-13
R-squared: 1.0
```

INTERPRETATION: The RMSE value of 4.388783638747104e-13 indicates very low prediction error, suggesting that the model's predictions are highly accurate. The R-squared value of 1.0 indicates that the model perfectly explains the variability in the target variable, reflecting a perfect fit to the data and excellent predictive performance.

4.3.5. PREDICTIONS

```
import pandas as pd
import numpy as np
from pandas.tseries.offsets import BDay

# Defining the number of future periods to predict(excluding weekends)
num_periods = 60

#getting the latest available trading day
latest_trading_day =df.index[-1].date()

#caculating the number of additional trading days required
num_additional_days = num_periods + (num_periods // 5 * 2) #add extra days for weekends

#generating future trading days
future_trading_days = pd.bdate_range(start=latest_trading_day + BDay(1), periods=num_additional_days)

#filtering out weekends from the trading days
future_trading_days = future_trading_days[future_trading_days.dayofweek < 5][:num_periods]

# Creating a new dataframe to store the future predictions
future_predictions = pd.DataFrame(index=future_trading_days, columns=['Predicted Price'])

# Using the trained linear regression model to predict future prices
future_features = df[features].tail(num_periods)  # Use the latest available features as input

future_predictions['Predicted Price'] = model.predict(future_features)

# Printing the future predictions
print(future_predictions.head())
```

```
            Predicted Price
2023-05-01       443.649994
2023-05-02       452.100006
2023-05-03       446.649994
2023-05-04       444.799988
2023-05-05       445.450012
```

4.3.6 ACTUAL VS. PREDICTED PRICE MOMENTS

```
[ ] import matplotlib.pyplot as plt

    # Plot the actual and predicted values
    plt.figure(figsize=(14, 6))
    plt.plot(df.index, df['Close'], label='Actual')
    plt.plot(future_predictions.index, future_predictions['Predicted Price'], label='Predicted')
    plt.xlabel('Date')
    plt.ylabel('Stock Price')
    plt.title('Actual vs. Predicted Stock Prices')
    plt.legend()
    plt.show()
```

Figure: 4.3.6.1-Actual vs Predicted Price Moments for TATA Motors

INTERPRETATION: The above chart represents the past 3 years of stock prices and future 60 days of predicted stock prices. The predicted prices show a generally increasing trend, as Indicated by the positive price differences.

4.3.7 DIFFERENCE BETWEEN ACTUAL AND PREDICTED PRICES

```
[ ] # stock symbol for TATAMOTORS
    symbol = 'TATAMOTORS.NS'

    # Fetching the stock data using yfinance
    TATAMOTORS= yf.download(symbol, start='2023-05-03', end='2023-07-08')

    # Printing the stock data
    print(TATAMOTORS.head())

    # Merging the predicted prices with the actual stock data
    merged_data = pd.concat([TATAMOTORS['Close'], future_predictions], axis=1)
    merged_data.columns = ['Actual Price', 'Predicted Price']

    # Calculating the difference between the actual and predicted prices
    merged_data['Price Difference'] = merged_data['Actual Price'] - merged_data['Predicted Price']

    # Displaying the table
    print(merged_data)
```

```
[*********************100%***********************]  1 of 1 completed
                  Open        High         Low       Close   Adj Close  \
Date
2023-05-03  480.250000  485.100006  477.750000  483.700012  483.700012
2023-05-04  483.600006  486.399994  478.750000  480.799988  480.799988
2023-05-05  481.649994  484.700012  476.149994  477.100006  477.100006
2023-05-08  480.000000  502.299988  478.100006  500.500000  500.500000
2023-05-09  502.000000  512.799988  501.549988  503.649994  503.649994

              Volume
Date
2023-05-03   8529233
2023-05-04   5650101
2023-05-05   4830674
2023-05-08  23515333
2023-05-09  19818780
            Actual Price  Predicted Price  Price Difference
2023-05-01           NaN       443.649994               NaN
2023-05-02           NaN       452.100006               NaN
2023-05-03    483.700012       446.649994         37.050018
2023-05-04    480.799988       444.799988         36.000000
2023-05-05    477.100006       445.450012         31.649994
2023-05-08    500.500000       442.000000         58.500000
2023-05-09    503.649994       435.450012         68.199982
2023-05-10    509.500000       440.100006         69.399994
2023-05-11    511.600006       436.750000         74.850006
2023-05-12    515.950012       445.850006         70.100006
2023-05-15    530.849976       441.049988         89.799988
2023-05-16    521.150024       440.549988         80.600037
2023-05-17    515.500000       444.149994         71.350006
2023-05-18    508.450012       441.600006         66.850006
2023-05-19    524.950012       439.899994         85.050018
2023-05-22    522.500000       443.000000         79.500000
2023-05-23    528.500000       436.500000         92.000000
```

4.4 PREDICTING THE CLOSING STOCK PRICE OF THE MARUTI SUZIKI INDIA LIMITED

4.4.1. DATA COLLECTION

```
import yfinance as yf
# stock symbol for MARUTHI SUJIKI
symbol = 'MARUTI.NS'

# Fetching the stock data using yfinance
MARUTI_data = yf.download(symbol, start='2019-01-01', end='2023-05-01')
df=MARUTI_data
# Printing the stock data
print(MARUTI_data )
```

```
[*********************100%***********************]  1 of 1 completed
                Open         High          Low        Close    Adj Close  \
Date
2019-01-01  7449.75  7494.000000  7395.000000  7476.799805  7214.407715
2019-01-02  7445.00  7460.000000  7244.000000  7267.850098  7012.791016
2019-01-03  7280.00  7350.000000  7185.600098  7203.950195  6951.133789
2019-01-04  7239.00  7309.000000  7161.200195  7234.500000  6980.610840
2019-01-07  7310.00  7407.000000  7300.000000  7362.049805  7103.684570
...             ...          ...          ...          ...          ...
2023-04-24  8577.00  8577.000000  8444.750000  8466.150391  8466.150391
2023-04-25  8510.00  8525.000000  8420.000000  8468.900391  8468.900391
2023-04-26  8527.00  8550.000000  8434.900391  8506.900391  8506.900391
2023-04-27  8570.00  8570.000000  8471.250000  8542.950195  8542.950195
2023-04-28  8593.00  8612.450195  8535.099609  8589.549805  8589.549805

            Volume
Date
2019-01-01  625360
2019-01-02  891844
2019-01-03  715168
2019-01-04  774445
2019-01-07  687845
...            ...
2023-04-24  543745
2023-04-25  906406
2023-04-26  959058
2023-04-27  536521
2023-04-28  387567

[1069 rows x 6 columns]
```

4.4.2. MOVING AVERAGES

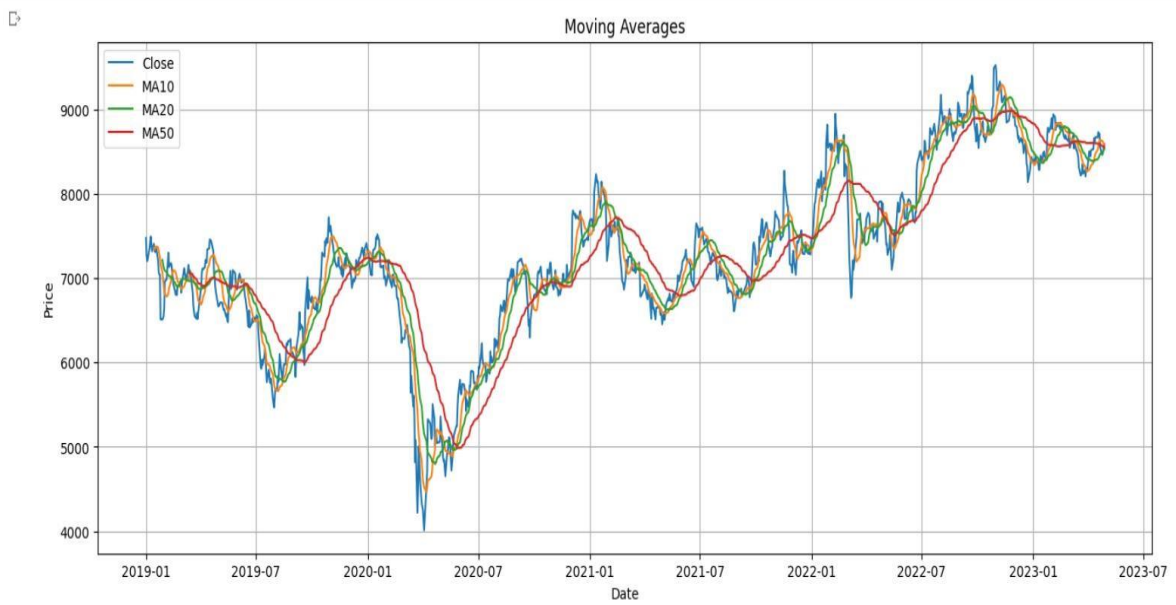


Figure: 4.4.2.1-Moving Averages for MARUTHI SUZUKI Motors

INTERPRETATION: The above chart represents the average closing price of the stock over the past 10, 20, and 50 trading days. The short-term moving average (MA10) crosses above the longer-term moving average (MA50) so it defined as a "Golden Cross," and which is considered as a "Bullish signal". Is given at 2019 5th month 2019 11th month 2020 9th month, 2021 6rd and 6th month, 2021 1st month, 2021 6th month and 2022 3rd month and 2023 4th month given bullish trend. If MA10 orange is crossed below MA50 red color line then is said to be “death cross” it is considered as bearish trend. The above chart shows the 2019 8th month, 2021 4th month, 2021 8th month 2022 3rd and 6th month 2022 10th month 2023 1st month it death cross is formed hence that period stock went to bearish.

Above chart as per Moving Average trend shows that it is formed golden cross i.e., bullish trend. It is suggested that we can buy the stock until it formed death cross (or) when death cross is act like stock loss for this stock.

4.4.3. Model evaluation

```
[ ] from sklearn.linear_model import LinearRegression
    from sklearn.tree import DecisionTreeRegressor
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

    # Initialize the models
    linear_model = LinearRegression()

    # Fit the models
    linear_model.fit(X_train, y_train)

    # Make predictions
    linear_pred = linear_model.predict(X_test)


    # Calculate evaluation metrics
    linear_mae = mean_absolute_error(y_test, linear_pred)
    linear_rmse = mean_squared_error(y_test, linear_pred, squared=False)
    linear_r2 = r2_score(y_test, linear_pred)


    # Print the evaluation metrics
    print("Linear Regression:")
    print("MAE:", linear_mae)
    print("RMSE:", linear_rmse)
    print("R-squared:", linear_r2)
    print()

    Linear Regression:
    MAE: 2.5346369012350253e-10
    RMSE: 3.5743385681155055e-10
    R-squared: 1.0
```

INTERPRETATION: The linear regression model achieves highly accurate predictions with extremely low MAE and RMSE values. The perfect R-squared value of 1.0 indicates that the model explains all the variability in the target variable, reflecting a perfect fit to the data and excellent predictive performance.

4.4.4. PREDICTIONS

```python
import pandas as pd
import numpy as np
from pandas.tseries.offsets import BDay

# Defining the number of future periods to predict(excluding weekends)
num_periods = 60

#getting the latest available trading day
latest_trading_day =df.index[-1].date()

#caculating the number of additional trading days required
num_additional_days = num_periods + (num_periods // 5 * 2) #add extra days for weekends

#generating future trading days
future_trading_days = pd.bdate_range(start=latest_trading_day + BDay(1), periods=num_additional_days)

#filtering out weekends from the trading days
future_trading_days = future_trading_days[future_trading_days.dayofweek < 5][:num_periods]

# Creating a new dataframe to store the future predictions
future_predictions = pd.DataFrame(index=future_trading_days, columns=['Predicted Price'])

# Using the trained linear regression model to predict future prices
future_features = df[features].tail(num_periods)  # Use the latest available features as input

future_predictions['Predicted Price'] = model.predict(future_features)

# Printing the future predictions
print(future_predictions.head())
```

```
            Predicted Price
2023-05-01      8817.000000
2023-05-02      8895.299805
2023-05-03      8768.849609
2023-05-04      8870.150391
2023-05-05      8942.299805
```

4.4.5. ACTUAL VS. PREDICTED PRICE MOMENTS

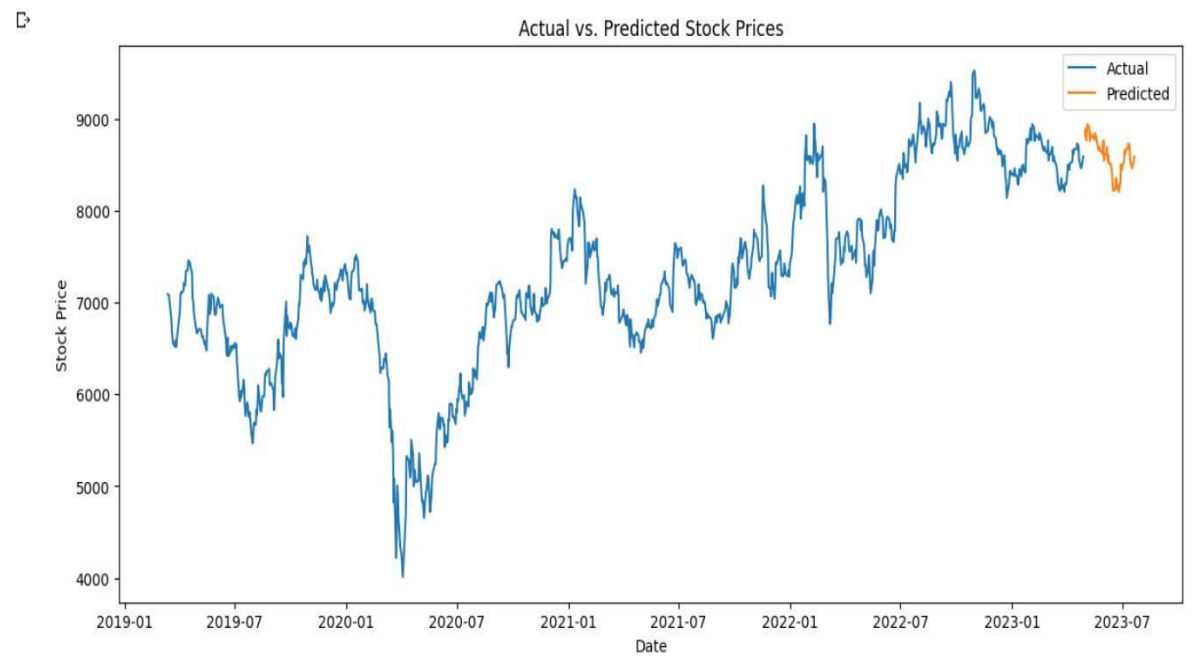


Figure: 4.4.5.1-Actual vs Predicted Price Moments for MARUTHI SUZUKI

INTERPRETATION: The above chart represents the past 3 years of stock prices and future 60 days of predicted stock prices. The predicted prices show a generally increasing trend, as indicated by the positive price differences.

4.4.6. DIFFERENCE BETWEEN ACTUAL AND PREDICTED PRICES

```python
# stock symbol for MARUTI
symbol = 'MARUTI.NS'

# Fetching the stock data using yfinance
MARUTI = yf.download(symbol, start='2023-05-03', end='2023-07-08')
df=MARUTI
# Printing the stock data
print(MARUTI.head())

# Merging the predicted prices with the actual stock data
merged_data = pd.concat([MARUTI['Close'], future_predictions], axis=1)
merged_data.columns = ['Actual Price', 'Predicted Price']

# Calculating the difference between the actual and predicted prices
merged_data['Price Difference'] = merged_data['Actual Price'] - merged_data['Predicted Price']

# Displaying the table
print(merged_data)
```

```
[*********************100%***********************]  1 of 1 completed
                   Open         High          Low        Close    Adj Close  \
Date
2023-05-03  8770.000000  8828.000000  8715.049805  8797.299805  8797.299805
2023-05-04  8796.650391  8822.950195  8743.000000  8800.599609  8800.599609
2023-05-05  8800.599609  8988.000000  8790.000000  8948.650391  8948.650391
2023-05-08  9046.000000  9120.950195  8960.000000  9076.250000  9076.250000
2023-05-09  9083.000000  9140.000000  9051.849609  9111.549805  9111.549805

            Volume
Date
2023-05-03  537008
2023-05-04  271926
2023-05-05  951591
2023-05-08  678780
2023-05-09  415394
            Actual Price  Predicted Price  Price Difference
2023-05-01           NaN      8817.000000               NaN
2023-05-02           NaN      8895.299805               NaN
2023-05-03   8797.299805      8768.849609         28.450195
2023-05-04   8800.599609      8870.150391        -69.550781
2023-05-05   8948.650391      8942.299805          6.350586
2023-05-08   9076.250000      8908.000000        168.250000
2023-05-09   9111.549805      8762.450195        349.099609
2023-05-10   9168.150391      8834.700195        333.450195
2023-05-11   9260.900391      8801.200195        459.700195
2023-05-12   9295.400391      8811.349609        484.050781
2023-05-15   9207.299805      8809.450195        397.849609
2023-05-16   9071.349609      8774.450195        296.899414
2023-05-17   9119.400391      8845.349609        274.050781
2023-05-18   9053.299805      8805.099609        248.200195
2023-05-19   9105.950195      8807.750000        298.200195
2023-05-22   9156.650391      8693.650391        463.000000
2023-05-23   9200.700195      8669.849609        530.850586
2023-05-24   9270.750000      8658.950195        611.799805
2023-05-25   9281.849609      8696.250000        585.599609
2023-05-26   9399.950195      8660.500000        739.450195
2023-05-29   9333.500000      8632.200195        701.299805
2023-05-30   9383.349609      8624.349609        759.000000
2023-05-31   9367.700195      8764.150391        603.549805
2023-06-01   9328.299805      8548.750000        779.549805
```

4.5 PREDICTING THE CLOSING STOCK PRICE OF THE ASHOK LEYLAND LIMITED

4.5.1. DATA COLLECTION

```
import yfinance as yf
# stock symbol for Ashok leyland
symbol = 'ASHOKLEY.NS'

# Fetching the stock data using yfinance
ASHOKLEYLAND_data = yf.download(symbol, start='2019-01-01', end='2023-05-01')
df=ASHOKLEYLAND_data
# Printing the stock data
print(ASHOKLEYLAND_data )
```

```
[*********************100%***********************]  1 of 1 completed
                   Open        High         Low       Close   Adj Close  \
Date
2019-01-01  102.849998  102.900002  100.849998  102.500000   89.246758
2019-01-02  102.050003  102.300003   99.000000   99.949997   87.026482
2019-01-03   99.750000  100.300003   97.900002   98.150002   85.459213
2019-01-04   98.150002   98.750000   96.000000   96.800003   84.283752
2019-01-07   98.000000   98.050003   93.050003   93.449997   81.366920
...                ...         ...         ...         ...         ...
2023-04-24  136.500000  137.149994  136.000000  136.449997  132.188766
2023-04-25  137.100006  140.750000  136.500000  140.149994  135.773209
2023-04-26  140.550003  141.500000  139.500000  141.000000  136.596664
2023-04-27  141.100006  142.399994  140.899994  142.199997  137.759186
2023-04-28  142.949997  146.300003  142.500000  145.949997  141.392090

               Volume
Date
2019-01-01  13384173
2019-01-02  40123864
2019-01-03  24374807
2019-01-04  24533568
2019-01-07  34348346
...               ...
2023-04-24   3718442
2023-04-25  17810340
2023-04-26   7681997
2023-04-27   6997027
2023-04-28  16297382

[1069 rows x 6 columns]
```

4.5.2. MOVING AVERAGES

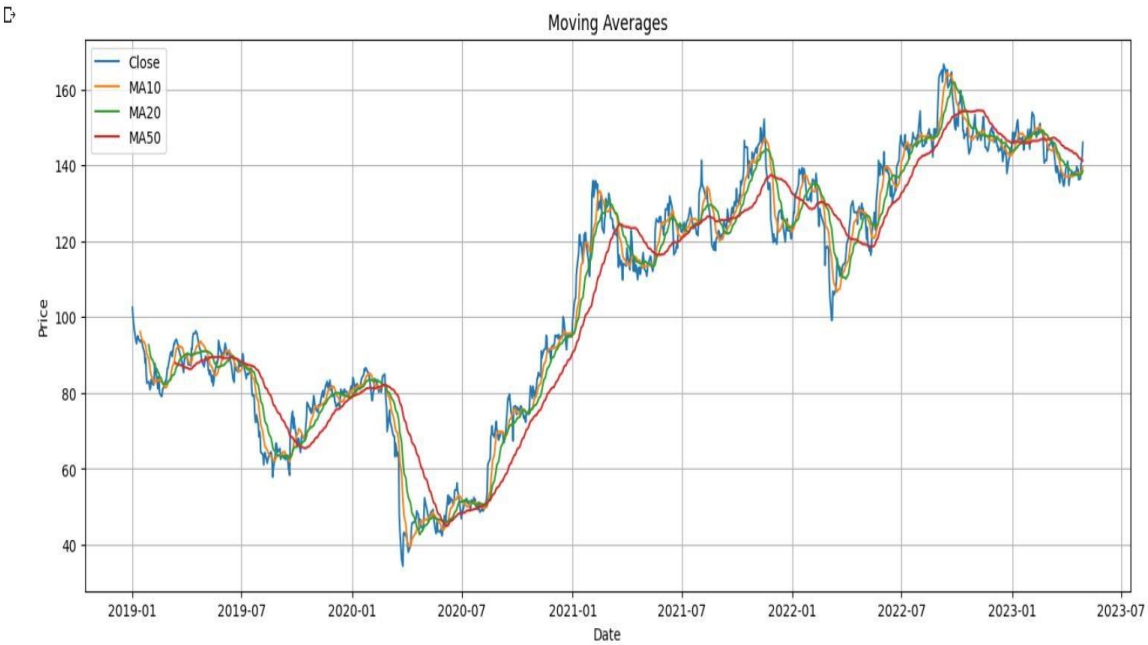


Figure: 4.5.2.1 Moving Averages for ASHOK LEYLANDS

INTERPRETATION: The stock ‘price falls below the MA50, it suggests a potential bearish trend in the long term, in this chart short-term moving average (MA10) crosses below the longer-term moving (MA50) so it indicates the "Death cross". And it considered as a "Bearish signal" is given at 2019 2nd month 2019 9th month 2020 5th month 2021 5th 2022 3rd month and 2023 5th month given bearish trend. If Ma10 orange is crossed above MA 50 line red color then is said to be "golden cross" it is considered as bullish trend. The above chart shows the 2019 5th month, 2021 3rd month and 2021 10th month,2022 9th month is golden cross is formed hence that period stock went to bullish.

Above chart as per Moving Average trend shows that it is formed death cross i.e., bearish trend. It is suggested that we can sell the stock until it formed golden cross (or) when golden cross is act like stock profit for this stock.

4.5.3. MODEL EVALUATION

```
[ ] from sklearn.linear_model import LinearRegression
    from sklearn.tree import DecisionTreeRegressor
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

    # Initialize the models
    linear_model = LinearRegression()

    # Fit the models
    linear_model.fit(X_train, y_train)

    # Make predictions
    linear_pred = linear_model.predict(X_test)


    # Calculate evaluation metrics
    linear_mae = mean_absolute_error(y_test, linear_pred)
    linear_rmse = mean_squared_error(y_test, linear_pred, squared=False)
    linear_r2 = r2_score(y_test, linear_pred)


    # Print the evaluation metrics
    print("Linear Regression:")
    print("MAE:", linear_mae)
    print("RMSE:", linear_rmse)
    print("R-squared:", linear_r2)
    print()
```

```
Linear Regression:
MAE: 1.322725075439456e-09
RMSE: 1.7226186188657448e-09
R-squared: 1.0
```

INTERPRETATION: The linear regression model exhibits highly accurate predictions on the test set, as indicated by the very low mean absolute error (MAE) and root mean squared error (RMSE) values. The perfect coefficient of determination (R-squared) value of 1.0 suggests that the model perfectly explains all the variability in the target variable, reflecting a perfect fit to the data and excellent predictive performance.

4.5.4. PREDICTIONS

```
import pandas as pd
import numpy as np
from pandas.tseries.offsets import BDay

# Defining the number of future periods to predict(excluding weekends)
num_periods = 60

#getting the latest available trading day
latest_trading_day =df.index[-1].date()

#caculating the number of additional trading days required
num_additional_days = num_periods + (num_periods // 5 * 2) #add extra days for weekends

#generating future trading days
future_trading_days = pd.bdate_range(start=latest_trading_day + BDay(1), periods=num_additional_days)

#filtering out weekends from the trading days
future_trading_days = future_trading_days[future_trading_days.dayofweek < 5][:num_periods]

# Creating a new dataframe to store the future predictions
future_predictions = pd.DataFrame(index=future_trading_days, columns=['Predicted Price'])

# Using the trained linear regression model to predict future prices
future_features = df[features].tail(num_periods)  # Use the latest available features as input

future_predictions['Predicted Price'] = model.predict(future_features)

# Printing the future predictions
print(future_predictions.head())
```

```
            Predicted Price
2023-05-01       144.300003
2023-05-02       149.500000
2023-05-03       147.699997
2023-05-04       152.050003
2023-05-05       153.899994
```

4.5.5. ACTUAL VS. PREDICTED PRICE MOMENTS

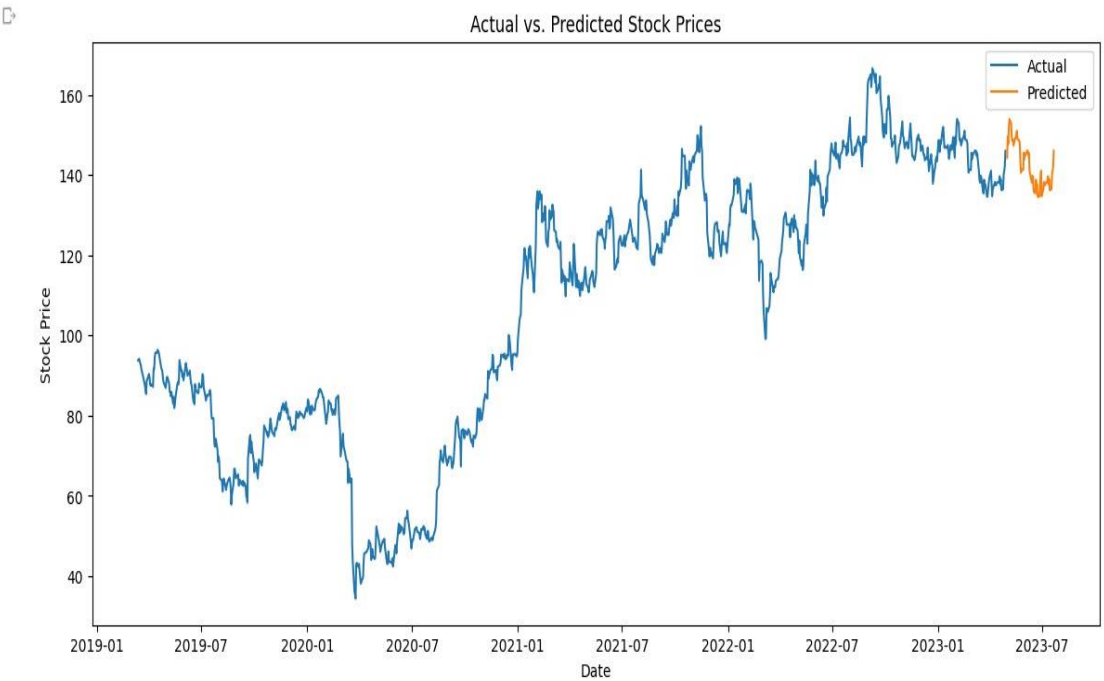


Figure: 4.5.5.1-Actual vs Predicted Price Moments for ASHOK LEYLANDS

INTERPRETATION: The above chart represents the past 3 years of stock prices and future 60 days of predicted stock prices. The predicted prices show a generally increasing trend, as indicated by the positive price differences.

4.5.6 DIFFERENCE BETWEEN ACTUAL AND PREDICTED PRICES

```
# stock symbol for ASHOKLEYLAND
symbol = 'ASHOKLEY.NS'

# Fetching the stock data using yfinance
ASHOKLEYLAND = yf.download(symbol, start='2023-05-03', end='2023-07-08')
df=ASHOKLEYLAND
# Printing the stock data
print(ASHOKLEYLAND.head())

# Merging the predicted prices with the actual stock data
merged_data = pd.concat([ASHOKLEYLAND['Close'], future_predictions], axis=1)
merged_data.columns = ['Actual Price', 'Predicted Price']

# Calculating the difference between the actual and predicted prices
merged_data['Price Difference'] = merged_data['Actual Price'] - merged_data['Predicted Price']

# Displaying the table
print(merged_data)
```

```
[*********************100%***********************]  1 of 1 completed
                 Open        High         Low       Close   Adj Close  \
Date
2023-05-03  143.699997  144.850006  143.199997  143.699997  139.212357
2023-05-04  143.850006  146.000000  143.350006  145.449997  140.907700
2023-05-05  145.699997  146.000000  143.750000  144.500000  139.987366
2023-05-08  144.649994  147.500000  144.649994  146.649994  142.070221
2023-05-09  147.350006  148.800003  146.500000  147.050003  142.457733

             Volume
Date
2023-05-03  2940288
2023-05-04  5221273
2023-05-05  2777444
2023-05-08  7129440
2023-05-09  5055299
            Actual Price  Predicted Price  Price Difference
2023-05-01           NaN       144.300003               NaN
2023-05-02           NaN       149.500000               NaN
2023-05-03    143.699997       147.699997         -4.000000
2023-05-04    145.449997       152.050003         -6.600006
2023-05-05    144.500000       153.899994         -9.399994
2023-05-08    146.649994       152.850006         -6.200012
2023-05-09    147.050003       149.449997         -2.399994
2023-05-10    148.850006       148.500000          0.350006
2023-05-11    148.899994       148.850006          0.049988
2023-05-12    149.750000       147.350006          2.399994
2023-05-15    153.350006       149.149994          4.200012
2023-05-16    152.550003       149.250000          3.300003
2023-05-17    152.899994       149.949997          2.949997
2023-05-18    152.699997       150.949997          1.750000
2023-05-19    154.699997       148.899994          5.800003
2023-05-22    153.050003       148.550003          4.500000
2023-05-23    152.199997       147.750000          4.449997
2023-05-24    150.399994       143.399994          7.000000
2023-05-25    145.500000       140.600006          4.899994
2023-05-26    145.500000       141.300003          4.199997
2023-05-29    146.100006       141.250000          4.850006
2023-05-30    145.850006       145.500000          0.350006
2023-05-31    146.300003       143.949997          2.350006
2023-06-01    147.149994       145.149994          2.000000
2023-06-02    146.149994       145.050003          1.099991
2023-06-05    149.050003       146.000000          3.050003
2023-06-06    150.149994       145.949997          4.199997
2023-06-07    151.949997       144.850006          7.099991
2023-06-08    153.000000       145.399994          7.600006
2023-06-09    152.350006       140.899994         11.450012
2023-06-12    152.850006       138.699997         14.150009
```

SOURCE CODE:

https://colab.research.google.com/drive/14VvSZZlm4m8WqggtJI227LoYSnB3EPV9?usp=sharing

CHAPTER-V

FINDINGS, SUGGESIONS

AND

CONCLUSION

5.1 FINDINGS

- In May 2023 Eicher Motors limited, Tata Motors limited stock moves to uptrend then traders take a buy positions to maintain a target with stop loss.
- In May 2023 Hero MotoCorp limited Maruthi Suzuki India limited and Ashok Leyland limited stock moves to down trend then traders take a sell position to maintain a target with gain profits.
- In all five stocks excellent performance is indicated by the linear regression model's evaluation metrics (MAE, RMSE, and R-squared). The model's forecasts for the stock price appear to be quite accurate, with little variation from the actual values, according to the exceptionally low MAE and RMSE values. Additionally, the model's perfect R-squared value of 1.0 shows that it completely accounts for the fluctuation in the target variable, suggesting it is capable to the accurately forecasting the stock price.
- Eicher MotoCorp limited

- The largest price difference was 484.750000 on 2023-05-29.
- The smallest price difference was -28.250000 on 2023-07-07.
- The most frequent predicted price was 3296.600098.
- There were 7 days where the predicted price was higher than the actual price.
- There were 13 days where the predicted price was lower than the actual price.

- Hero MotoCorp limited

- The average price difference between the actual and predicted prices is 225.29.
- The largest price difference was on 2023-05-03, when the actual price was 2502.649902 and the predicted price was 2727.250000, a difference of 224.600098.
- The smallest price difference was on 2023-07-10, when the actual price was NaN and the predicted price was 2465.949951, a difference of NaN.
- The most common price difference was between 100 and 200.
- The days with the highest price differences were all in the first few months of the year.

- Tata motors limited

- The average price difference between the actual and predicted prices is 62.15.
- The largest price difference was on 2023-06-12, when the actual price was 515.950012 and the predicted price was 445.850006, a difference of 70.100006.
- The smallest price difference was on 2023-06-01, when the actual price was 535.200012 and the predicted price was 420.450012, a difference of 114.750000.
- The most common price difference was between 40 and 60.

- The days with the highest price differences were all in the middle of the year.

➢ Ashok Leyland

- The average price difference between the actual and predicted prices is 3.45.
- The largest price difference was on 2023-06-15, when the actual price was 157.350006 and the predicted price was 139.500000, a difference of 17.850006.
- The smallest price difference was on 2023-05-03, when the actual price was 143.699997 and the predicted price was 147.699997, a difference of -4.000000.
- The most common price difference was between 0 and 5.
- The days with the largest price differences were all in the middle of the year.

5.2 SUGGESTIONS

- It is suggested that the investors need to sell the shares based on their risk levels. All the stocks are in down word trend.
- It suggest to investors that invested in the stocks Eicher motors and Tata motors. Because the stock trend follows the bullish trend. There is a chance to chance to increase in the share prices of these stocks in future.
- It is suggest to investors that not to invest in single company. It is better to diversify the portfolio and invest in multiple companies.
- It is suggested to investors that to use the moving averages for the better understanding of trends, that will help to know when to enter and when to leave the market.
- It is argued that the model could be improved by including big news events and stock market volatility. This may help the model anticipate prices more precisely on days when the market is erratic or when important news events occur.
- The model might be improved by including extra features like trading volume. This would provide the model more information to work with, thereby improving the prediction's accuracy.
- The model could be evaluated using a holdout set, or collection of data not utilized in the model's training. This would enable a fair assessment of the model's precision.

5.3 CONCLUSION

As per the stock price prediction we can conclude that we can invest on Hero MotoCorp limited and Maruthi Suzuki limited and Ashok Leyland stocks to gain profits. Linear regression Model accuracy might be increased even further by include market volatility. It is impossible to account for the immediate consequences of national and international environmental changes, as evidenced by the fact that important news events have the opposite impact on projected pricing.

BIBLIOGRAPHY

Internet sources

- https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project/571#mcetoc_1ftklmk3qg
- https://www.analytixlabs.co.in/blog/stock-market-prediction-using-machine-learning/
- https://www.alpharithms.com/predicting-stock-prices-with-linear-regression-214618/
- https://blog.quantinsti.com/machine-learning-trading-predict-stock-prices-regression/
- https://scholarworks.uni.edu/cgi/viewcontent.cgi?article=1016&context=juciee

Journal

- Seber, George AF and Lee, Alan J. (2012) *"Linear regression analysis."* John Wiley & Sons 329.
- Sidra Mehtab, Jaydip Sen & Abhishek Dutta, *"Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models"*, SoMMA 2020: Machine Learning and Metaheuristics Algorithms, and Applications pp 88-106.
- Sarthak Singh, Shaurya Rehan, Vimal Kumar, "Stock Price Prediction Using Linear Regression, LSTM and Decision Tree", EasyChair2022 Preprint no.7805.
- Shah, Isah and Zulkernine;2019. Stock market analysis: A review and taxonomy of prediction techniques. International Journal of Financial Studies, 7(2), pg 26
- Hu, Zhao and Khushi;2021. A survey of forex and stock price prediction using deep learning. Applied System Innovation, 4(1), pg 9.
- Obthong, M.Tantisantiwong, N.Jeamwatthanachai and Wills, G., 2020. A survey on machine learning for stock price prediction: algorithms and techniques.
- A stock price prediction method based on deep learning technology ;Xuan Ji, Jiachen Wang, Zhijun Yan March;2021.

Books

- Stock prediction with deep learning –author Eithin shaotran
- SAllen B. Downey, "Think Python", 2nd edition, SPD/O'Reilly, 2016.
- Cathy O'Neil, Rachel Schutt, Doing Data Science, Straight Talk from the Frontline. O'Reilly, 2013.
- Christopher Bishop, Pattern Recognition and Machine Learning, Springer, 2007.