STOCHASTIC MODELLING AND FORECASTING METAL PRICE DYNAMICS

A specialization project submitted in partial fulfilment of the requirements for the award of the degree of

MASTER OF SCIENCE IN DATA SCIENCE

Submitted by

ANNIE MAJELLA V

(22-PDS-006)

Under the guidance of

DR. P. MANIKANDAN

ASSISTANT PROFESSOR

DEPARTMENT OF DATA SCIENCE

LOYOLA COLLEGE

CHENNAI 600 034

APRIL 2024

CERTIFICATE

This is to certify that the project entitled **"STOCHASTIC MODELLING AND FORECASTING METAL PRICE DYNAMICS"** submitted to the Loyola College, Chennai in partial fulfilment of the requirements for the award of the **Degree of MASTER OF SCIENCE IN DATA SCIENCE** is a record of original project work done by **Ms Annie Majella** during the period 2023 - 2024 of her study in the Department of Data Science, Loyola College, Chennai under my supervision and guidance and the project has not formed the basis for the award of any Degree / Diploma / Associateship / Fellowship or other similar title of any candidate of any University.

Date:

Head of the Department	**Signature of the Guide**
Dr. T. Rajaretnam	**Dr. P. Manikandan**
M.Sc., M.Tech., Ph.D	M.Sc., M. Phil., Ph.D.

Submitted for the Viva-Voce Examination held on ________________

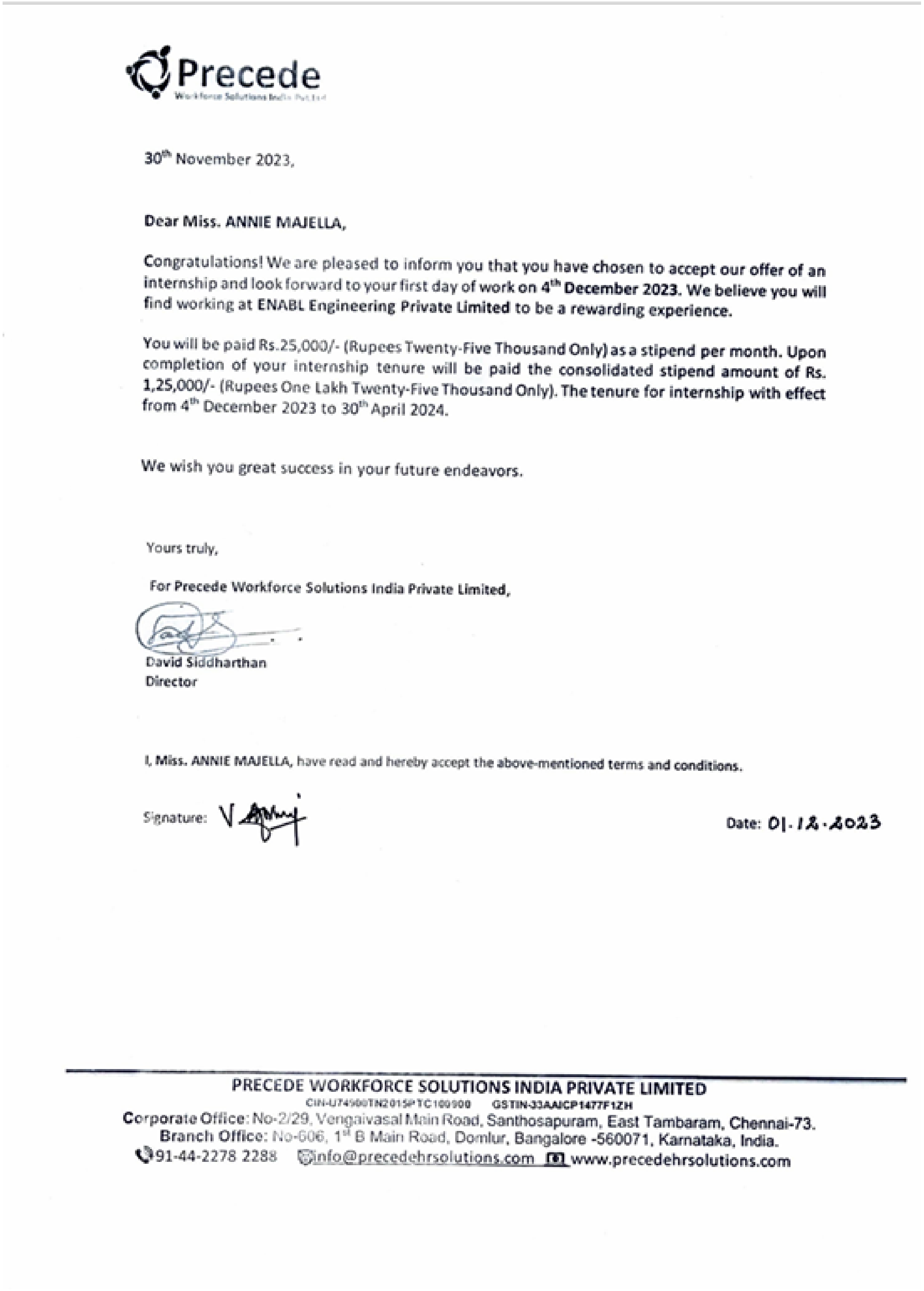
1. Examiner **2. Examiner**

DECLARATION

I, **ANNIE MAJELLA V**, hereby declare that the project entitled “STOCHASTIC MODELLING AND FORECASTING METAL PRICE DYNAMICS" submitted to the Loyola College, Chennai in partial fulfilment of the requirements for the award for the Degree of **M.Sc. DATA SCIENCE** is a record of original and independent project work done by me during 2023 – 2024 under the Supervision and Guidance of **Dr. P. Manikandan** MSc., M.Phil., Ph.D., Assistant Professor, Department of Data Science, Loyola College, Chennai and it has not formed the basis for the award of any Degree / Diploma / Associateship / Fellowship or other similar title to any candidate in any University.

Date: **Signature of the candidate:**

ANNIE MAJELLA V



Precede
Workforce Solutions India Pvt Ltd

30th November 2023,

Dear Miss. ANNIE MAJELLA,

Congratulations! We are pleased to inform you that you have chosen to accept our offer of an internship and look forward to your first day of work on **4th December 2023. We believe you will** find working at **ENABL Engineering Private Limited** to be a rewarding experience.

You will be paid Rs.25,000/- (Rupees Twenty-Five Thousand Only) as a stipend per month. Upon completion of your internship tenure will be paid the consolidated stipend amount of Rs. 1,25,000/- (Rupees One Lakh Twenty-Five Thousand Only). The tenure for internship with effect from 4th December 2023 to 30th April 2024.

We wish you great success in your future endeavors.

Yours truly,

For Precede Workforce Solutions India Private Limited,

[illegible]

David Siddharthan
Director

I, Miss. ANNIE MAJELLA, have read and hereby accept the above-mentioned terms and conditions.

Signature: [illegible] Date: 01.12.2023

PRECEDE WORKFORCE SOLUTIONS INDIA PRIVATE LIMITED
CIN-U74900TN2015PTC100900 GSTIN-33AAICP1477F1ZH
Corporate Office: No-2/29, Vengaivasal Main Road, Santhosapuram, East Tambaram, Chennai-73.
Branch Office: No-606, 1st B Main Road, Domlur, Bangalore -560071, Karnataka, India.
91-44-2278 2288 info@precedehrsolutions.com www.precedehrsolutions.com

ACKNOWLEDGEMENT

Six months of effort and hard work for the successful completion of project work has come to an end. I am extremely delighted and grateful for the motivation, guidance and assistance that I have received during the course of my project work. This project has been completed successfully only due to such guidance and assistance for which I express my sincere gratitude.

I thank **Rev. Dr. A. Louis Arockiaraj SJ** the Principal, Loyola college and **Rev. Dr. B. Jeyaraj SJ** the Secretary, for providing me the opportunity to pursue in this prestigious institution and gain work space experience for the future purposes.

I wish to express my sincere gratitude to **Dr. T. Rajaretnam**, Head of the Department of Data Science, Loyola College, Chennai, for his constant words of wisdom and motivation.

I sincerely thank **Dr. P. Manikandan**, Assistant Professor, Department of Data Science, Loyola College, Chennai, my internal project guide for his continuous guidance, support and encouragement throughout the completion of the project.

I extend my gratitude towards all the faculty members of the Department of Data Science for their direct and indirect assistance for the successful completion of the project.

I am thankful to my external guide **Mr. Prassan Cuman Nagendra Babu**, Digital solutions-Lead, ENABL A/S, for mentoring me throughout the project.

I extend my gratitude and thanks towards my family and friends for their goodwill and support in the successful completion of the project.

I would like to thank all Administrative, Technical and Library Staff of Loyola College, Chennai for their assistance and help during the period of my study.

ANNIE MAJELLA V

ABSTRACT

The volatility and unpredictability of metal prices pose significant challenges for stakeholders in the metal industry, including producers, traders, and investors. In this study, we explore the application of Monte Carlo simulation as the final model for stochastic modelling and forecasting of metal price dynamics. Historical metal price data spanning a significant time period is collected from Yahoo Finance using the yfinance library. This data is then meticulously pre-processed to handle missing values, outliers, and inconsistencies, ensuring the integrity of the subsequent analysis.

The core of our methodology lies in the Monte Carlo simulation, which generates probabilistic forecasts of future metal prices. This simulation approach provides a robust framework for capturing the inherent uncertainty and complexity of metal price dynamics, allowing stakeholders to make informed decisions in a volatile market environment. Furthermore, the Monte Carlo simulation enables us to assess the uncertainty associated with the predictions, providing valuable insights into the range of possible outcomes and risk factors.

In addition to the Monte Carlo simulation, we also leverage other forecasting models such as ARIMA, Prophet, and LSTM for comparison and evaluation purposes. The performance metrics of these models such as Mean Absolute Error (MAE) are used for internal validation and benchmarking against the Monte Carlo simulation results.

The evaluation metrics, including MAE and scores from the comparison models, are utilized to assess the performance of the Monte Carlo simulation and validate its effectiveness in predicting metal price dynamics. The results and insights derived from this comprehensive analysis provide stakeholders with actionable information to make strategic decisions regarding pricing strategies, inventory management, and risk mitigation approaches in the ever-evolving metal market landscape.

KEYWORDS: machine learning, time series, historical data, ARIMA, LSTM, PROPHET, MONTE CARLO, decision-making, efficiency, predictability, metal price prediction, LME.

LIST OF ABBREVIATIONS

S.No	ACRONYM	ABBREVIATION
1.	LME	London Metal Exchange
2.	AR	Auto Regressive
3.	MA	Moving Average
4.	ARIMA	Auto Regressive Integrated Moving Average
5.	LSTM	Long Short Term Memory
6.	CSV	Comma Separated Value
7.	MSE	Mean Squared Error
8.	MAE	Mean Absolute Error
9.	RMSE	Root Mean Squared Error
10.	EDA	Exploratory Data Analysis
11.	ACF	Auto Correlation Function
12.	PACF	Partial Auto Correlation Function

LIST OF FIGURES

LIST OF TABLES

INDEX

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

The ability to predict and forecast metal prices is of paramount importance for various industries, including manufacturing, construction, and finance. By harnessing the power of data-driven techniques and advanced statistical models, we aim to provide valuable insights into the future trends of key metal commodities. In recent years, there has been growing interest in the potential of machine learning algorithms to improve the procurement process.

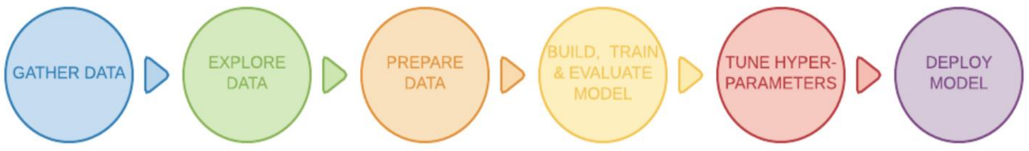


Fig.1.0 Data Science Pipeline

Source: developers.google

This project aims to explore the potential of machine learning algorithms in predicting the metal stock price and utilizes the Django web framework for building a user-friendly interface, allowing users to input parameters and obtain real-time forecasts. Data acquisition is facilitated by the yfinance library for fetching historical price data from Yahoo Finance. The modelling and simulation tasks are performed using Python, with libraries such as pandas, numpy, and Bokeh for data manipulation, analysis, and visualization. The successful implementation of this project holds significant implications for stakeholders involved in metal trading, investment, and risk management. By providing timely and accurate forecasts of metal prices, informed decision-making processes can be facilitated, leading to enhanced operational efficiency and strategic planning across various industries.

1.2 Essential Background Information

Time series forecasting

Time series forecasting is a statistical technique used to predict future values of a variable based on its past behaviour. It involves analysing a sequence of data points measured over time, and then using this information to make informed predictions about future values of the variable. Time series forecasting is widely used in various fields, including finance, economics, weather forecasting, and inventory management, among others.

The primary objective of time series forecasting is to identify patterns and trends in the data that can be used to make accurate predictions about future values of the variable. Various statistical and machine learning techniques are used for time series forecasting, including ARIMA, exponential smoothing, and neural networks.

One of the critical challenges in time series forecasting is accounting for the various factors that can influence the variable being predicted. These factors may include seasonality, trends, cyclical fluctuations, and random events. Failure to account for these factors can lead to inaccurate predictions.

Time series forecasting is a powerful tool for predicting future values of a variable based on its past behaviour. By analysing historical data and identifying patterns and trends, time series forecasting can help businesses and organizations make informed decisions and optimize their operations.

1.2.1 Domain of Research

Time series forecasting is a domain of research that involves analysing and predicting patterns and trends in time series data. Time series data is a type of data that is collected over time, with observations made at regular intervals. Examples of time series data include stock prices, weather data, sales data, and sensor data.

Time series forecasting involves using statistical and machine learning techniques to analysis and model time series data and make predictions about future values. The goal of time series forecasting is to understand and predict patterns in the data, so that businesses and organizations can make informed decisions based on this information. Some of the key techniques used in time series forecasting include autoregressive integrated moving average (ARIMA) models, exponential smoothing models, and deep learning models such as recurrent neural networks (RNNs) and long short-term memory (LSTM) networks. Time series forecasting is used in a wide range of applications, including finance, marketing, healthcare and transportation.

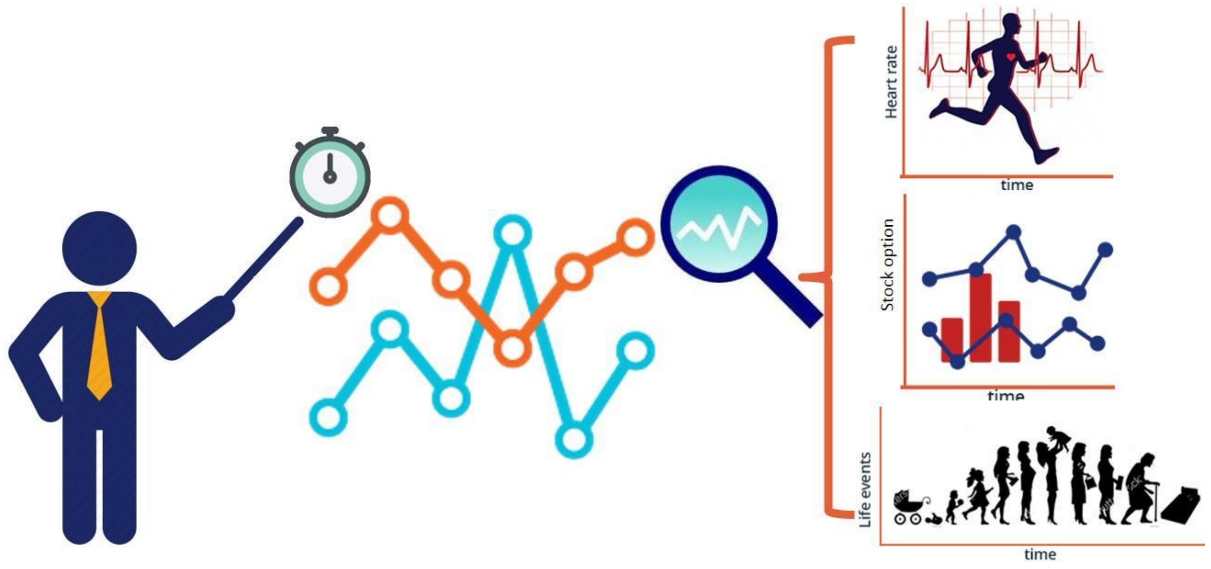


Fig.1.2.1 Domain of time series

1.2.2 Motive behind Chosen Domain

The time series domain is often chosen for analysis when the data is collected over time and the order of observations is important. Time series data has a natural temporal ordering and may exhibit patterns and trends over time, such as seasonality, cyclicality, or trends. In many fields such as finance, economics, meteorology, and engineering, time series data analysis is crucial for understanding and predicting future behaviour. Time series models can be used to model and forecast future values of a variable based on its past behaviour. This makes it particularly useful for tasks such as predicting future stock prices, sales, weather patterns, and more. Time series models can also be used for anomaly detection, trend analysis, and seasonal forecasting. Overall, the time series domain provides a rich source of data with a wide range of applications in many different fields.

1.3. Scope of the project

The scope of metal price prediction project using time series forecasting would be to predict future demand for goods and services based on historical data. This would allow organizations to optimize their procurement process by ordering the right quantities at the right time, minimizing waste and reducing costs. Time series forecasting can also help identify patterns and trends in procurement data, allowing organizations to make informed decisions about when to buy and how much to buy.

Additionally, by using machine learning algorithms, e-procurement systems can continuously learn from past data and improve their forecasting accuracy over time. Overall, the scope of an e-procurement project using time series forecasting is to improve efficiency, reduce costs, and make more informed procurement decisions.

1.4. Problem Statement

The problem statement of this project is to develop an efficient time series forecasting model to predict the future demand for metals in a procurement system. E-Procurement is an electronic procurement system that enables organizations to streamline their procurement processes by automating their procurement activities. E-Procurement systems help organizations to reduce the time and cost associated with procurement activities, improve transparency and accountability, and enhance the overall efficiency of the procurement process. However, to achieve these benefits, it is crucial to accurately forecast the future demand for metals. The demand for metals in a procurement system can vary significantly over time, and it is influenced by various factors, such as changes in market conditions, customer behaviour, and economic factors. Therefore, accurate forecasting of demand is essential to ensure that the procurement system is adequately equipped to meet the future demand for metals.

The current approach to forecasting demand in the E-Procurement system relies on historical data and basic statistical methods, such as moving averages and exponential smoothing. However, these methods are not always accurate, and they do not take into account the complexity of the procurement system and the various factors that can influence demand. Therefore, the problem statement of this project is to develop a more sophisticated time series forecasting model that can accurately predict future prices of metals in the procurement system.

The model should be able to take into account the various factors that can influence demand, such as changes in market conditions and economic factors, and it should be able to adapt to changing conditions over time. The successful development of an accurate time series forecasting model would enable organizations to better plan their procurement activities, reduce costs, and improve overall efficiency.

Objectives

- Develop and implement a robust Monte Carlo simulation model to generate probabilistic forecasts of future metal prices, taking into account the inherent uncertainty and complexity of the market dynamics.
- Evaluate and compare the performance of Monte Carlo simulation model with the traditional time series forecasting models such as ARIMA, Prophet, and LSTM in predicting metal price dynamics, using metrics such as Mean Absolute Error (MAE) and

Root Mean Squared Error (RMSE) for validation.

- Explore the seasonal patterns and trends in historical metal price data using advanced time series analysis techniques, including decomposition and trend analysis, to gain insights into the underlying factors driving price fluctuations.
- The project aims to provide decision-makers with accurate and timely information about the procurement process. This can help in making informed decisions about procurement planning, inventory management, and supplier relationship management.

CHAPTER-2

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 Introduction

The literature review section of this study aims to provide a comprehensive overview of existing research, methodologies, and findings related to stochastic modelling and forecasting of metal price dynamics. The volatility and unpredictability of metal prices have been a subject of extensive study in the fields of economics, finance, and data science, as stakeholders in the metal industry seek effective strategies for pricing, risk management, and decision-making. This literature review will explore key themes and topics such as time series analysis, forecasting models, Monte Carlo simulation, ARIMA, LSTM, Prophet, and their applications in predicting metal price movements. By synthesizing and analyzing existing literature, this study aims to build upon the current knowledge base and identify gaps, challenges, and opportunities for further research in the field of metal price forecasting.

2.2 Review on Time series forecasting models

1. A hybrid approach based on wavelet transform and deep neural networks to forecast time series data. The method achieves high accuracy in predicting future values. It consists of two main components: a wavelet transform-based feature extraction method and a deep neural network model for forecasting. The deep neural network model is designed to learn the complex patterns in the wavelet coefficients and make accurate predictions for future time steps. The hybrid method is tested on several benchmark time series datasets and compared the results to traditional time series forecasting techniques such as ARIMA and exponential smoothing. *[Qi et al., 2017]*
2. Time series forecasting is explored with a focus on deep learning approaches, particularly Long Short-Term Memory (LSTM) networks. The effectiveness of LSTMs in time series forecasting tasks compared to traditional methods like ARIMA is investigated by the authors. The advantages of LSTMs in handling complex non-linear relationships within time series data is discussed. *[Xiaoqi Chen et al., 2023]*
3. The application of time series analysis in anomaly detection is investigated in this paper. A novel method for identifying unusual patterns within time series data is proposed by the authors. Statistical techniques and machine learning algorithms are leveraged to effectively

detect anomalies that may signal potential problems or critical events. *[V. Chandola et al., 2009]*

4. The use of time series analysis in financial markets is explored in this paper. Various techniques for modeling and forecasting stock prices, exchange rates, and other financial data are examined. The importance of time series analysis in making informed investment decisions and managing financial risk is discussed. *[R. Ashley et al., 2009]*
5. The application of time series analysis in climate science is studied in this paper. Various techniques for analyzing and forecasting climate data, such as temperature, precipitation, and sea level rise are discussed. The research highlights the role of time series analysis in understanding climate change and its potential impacts. *[Wilks D.S., 2006]*
6. The use of Bayesian methods for time series analysis is investigated. The authors explore how Bayesian techniques can be applied to model uncertainty and improve forecasting accuracy. The advantages of Bayesian methods in dealing with missing data and incorporating prior knowledge into the analysis are discussed. *[Andrew Gelman et al., 2013]*
7. A new approach for analyzing and forecasting time series data based on wavelet transforms is introduced. The authors discuss how wavelets can be used to decompose time series into different frequency components, enabling better understanding of the underlying dynamics. The potential of wavelet-based methods for various time series applications is explored. *[G. P. Nason, 2008]*

8. The use of transfer learning for time series forecasting is investigated. The authors explore how knowledge gained from one time series forecasting task can be applied to improve the performance on a different but related task. The research highlights the potential of transfer learning for leveraging existing knowledge and improving forecasting accuracy in various domains. *[Bao H. et al., 2022]*

2.3 Review on LSTM, Prophet models

1. This paper introduces Prophet, (Facebook) developed open-source tool for time series forecasting. Prophet utilizes an additive model to capture trends, seasonality, and holiday effects within the data. The model's strengths in handling various time series problems, including its ease of use and interpretability of results is discussed by the authors. *[Sean J.*

Taylor and Benjamin Letham, 2017]

2. The performance of Prophet compared to other time series forecasting methods is investigated in this paper. Prophet's accuracy on various datasets and benchmark its results against traditional models like ARIMA and Exponential Smoothing is evaluated. The scenarios where Prophet excels and offers insights into its limitations The paper explores the is explored. *[Hamed Khashei et al., 2021]*

3. The application of Prophet for forecasting short-term electricity demand is examined in this study. The authors explore how Prophet's ability to capture seasonality and daily patterns proves valuable in predicting electricity usage. The research highlights the effectiveness of Prophet in applications requiring high-frequency forecasting with strong seasonal components. *[Yixuan Liu et al., 2020]*
4. The use of Prophet for forecasting tourism demand is investigated in this paer. The authors discuss how Prophet's ability to handle holidays and seasonality aligns well with the cyclical nature of tourism data. The potential of Prophet for informed decision-making in the tourism industry, such as resource allocation and marketing strategies is explored. *[Yuanyuan Feng et al., 2021]*

5. The concept of Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN) architecture is explored in this paper. LSTMs address the vanishing gradient problem that hinders traditional RNNs in handling long-term dependencies within sequential data. The core components of LSTMs, including forget gates, input gates, and output gates, and how they facilitate learning long-range dependencies are explained by the authors. *[Sepp Hochreiter and Jürgen Schmidhuber, 1997]*
6. The application of LSTMs for time series forecasting tasks is investigated in this paper. The performance of LSTMs against traditional methods like ARIMA and Exponential Smoothing are compared by the author. The ability of LSTMs to capture complex non-linear relationships within time series data, leading to improved forecasting accuracy is highlighted in this paper. *[Xiaoqi Chen et al., 2023]*
7. The use of LSTMs for natural language processing (NLP) tasks, such as machine translation and text classification, is explored in this study. The authors discuss how LSTMs can effectively model sequential relationships within text data, leading to significant advancements in NLP applications. [Olah Chris, accessed March 29, 2024]

8. The application of LSTMs for anomaly detection in time series data is investigated. An LSTM-based approach for identifying unusual patterns that deviate from the typical behavior of the data is approached by the author. The potential of LSTMs in detecting anomalies in various domains, such as fraud detection and sensor data analysis is highlighted in this study. *[Lei Guo et al., 2022]*
9. The use of LSTMs for video analysis and action recognition are explored in this study. How LSTMs can learn temporal dependencies within video sequences, enabling them to effectively recognize and classify human actions are discussed by the author. The paper contributes to the field of computer vision by demonstrating the power of LSTMs in video understanding tasks. *[Zuxuan Wu et al., 2016]*

2.4 Review on ARIMA and Monte Carlo models

1. A Study on Time Series Forecasting Using Long Short-Term Memory Networks compares the effectiveness of Long Short-Term Memory networks with traditional methods like ARIMA for time series forecasting tasks. The advantages and limitations of ARIMA models, particularly in handling complex non-linear relationships within data is discussed. *[Xiaoqi Chen et al., 2023)*
2. An Analysis of Time Series Analysis and Forecasting Techniques offers a comprehensive assessment of various time series analysis and forecasting techniques, including ARIMA. The strengths and weaknesses of ARIMA compared to other established and emerging methods in real-world scenarios are explored. *[ResearchGate, Accessed March 29, 2024]*
3. Autoregressive Integrated Moving Average (ARIMA) Prediction Model provided a concise explanation of ARIMA models. The core aspects of ARIMA, including its autoregressive nature for predicting future values based on past values, and its use of lagged moving averages to smooth data is highlighted. [*Investopedia, Accessed March 29, 2024]*
4. The limitations of the Monte Carlo method is explored in this paper and variance reduction techniques to improve its efficiency is proposed. Methods like stratified sampling and antithetic variables that can reduce the number of simulations required to achieve a desired level of accuracy are discussed. The research contributes to the ongoing development of the Monte Carlo method by suggesting strategies for obtaining reliable results with fewer simulations. *[Luc Devroye, 2004]*
5. This historical review examines the origins and development of the Monte Carlo method.

The early applications of the method in scientific research and its subsequent evolution into a powerful computational tool is discussed. The paper sheds light on the key figures and theoretical advancements that shaped the Monte Carlo method over time. *[Pierre E. Lemonnier, 2015]*

6. This tutorial guides users through the process of implementing the Monte Carlo method using popular programming languages like Python or R. The tutorial provides code examples and practical tips for simulating various scenarios and analyzing the results. *[Machine Learning Mastery, 2023]*
7. The use of the Monte Carlo method for portfolio optimization in finance is investigated. How the method can be used to simulate different market scenarios and evaluate the risk-return profiles of various investment strategies are discussed by the authors. The importance of Monte Carlo simulations in making informed investment decisions- is highlighted. *[Paul Glasserman, 2004]*
8. The current state-of-the-art advancements in Monte Carlo methods is analyzed in this survey study. Ongoing research areas like parallel computing for large-scale simulations and novel applications in emerging scientific fields are discussed by the authors. The paper provides valuable insights into the future directions of Monte Carlo method development. *[Gareth O. Roberts and Duane Diebolt, 2013]*

2.5 Conclusion:

This literature review explored a variety of approaches for time series analysis and forecasting. The reviewed papers highlighted the strengths and weaknesses of established methods like Monte Carlo simulation, ARIMA, Exponential Smoothing, while also introducing recent advancements like Prophet and LSTM networks.

Key findings include:

- ✓ ARIMA remains a foundational and interpretable model for time series forecasting, particularly for stationary data.
- ✓ Prophet offers an accessible and user-friendly alternative for various forecasting tasks, especially when dealing with seasonality and holidays.
- ✓ LSTM networks demonstrate high potential for capturing complex non-linear relationships within time series data but may require more computational resources and expertise to implement.

- ✓ Monte Carlo simulations offer valuable tools for generating probabilistic forecasts and assessing the uncertainty associated with time series predictions. It enables generating probabilistic forecasts, providing a more comprehensive picture of future possibilities and incorporating the inherent uncertainty in time series data. This is crucial for informed decision-making under risk.

This literature review comprehensively examined various approaches in Time series analysis. The reviewed papers explored established methods like ARIMA, LSTM, Prophet, Monte Carlo. By critically evaluating the existing literature on time series analysis methods for forecasting of metal price dynamics, this review has established a strong foundation for my project.

CHAPTER -3

ORGANIZATION PROFILE

3. ORGANIZATION PROFILE

3.1 Introduction

ENABL ENGINEERING PVT LTD

Fig.3.1 ENABL logo

ENABL Engineering Pvt Ltd, a subsidiary of ENABL A/S Denmark, was established on March 1, 2021, in Chennai, India – the business hub that is home to major names in the industry. As a provider of specialized Engineering Services to Global Wind Energy OEMs, ENABL has firmly established its superiority in the niche. India offers a mature hiring ecosystem, and ENABL is on a sure-footed path to achieve ambitious headcount growth.

Driven by organizational maturity, ENABL ramped up and optimized core strength, expanding and scaling up operations in:

- Mechanical Design
- FEA
- Project Management
- Technical Documentation and
- Sustaining Engineering

Service - Wherever Wind Is

With service and maintenance wherever you are, we can ensure uptime and keep your equipment working to their full potential. We customize service agreements to your exact needs.

3.2 Vision and Mission

DIGITAL SOLUTIONS

Digital transformation is currently changing the way we do business. By digitalizing processes, we can replace manual and paper-based procedures by data-driven analytics. The Digital Platform from ENABL collects and visualizes data from all types of equipment and gives you:

- Significant reduced costs
- Overview of the performance of your equipment
- Increased equipment utilization and availability
- Improved safety
- Reduced CO2 emission

3.3 Product Offering

Service

- Service & Maintenance
- Service Solutions
- Digital Service Solutions
- Equipment Rental
- Service Agreements
- Site Solutions
- Specialized Work Action Team

Consulting

- Engineering Services
- Supplier Partnership
- Full Value Chain

Equipment

- Tower & Monopile Production Equipment
- Production Equipment
- Transport, Installation & Service Equipment

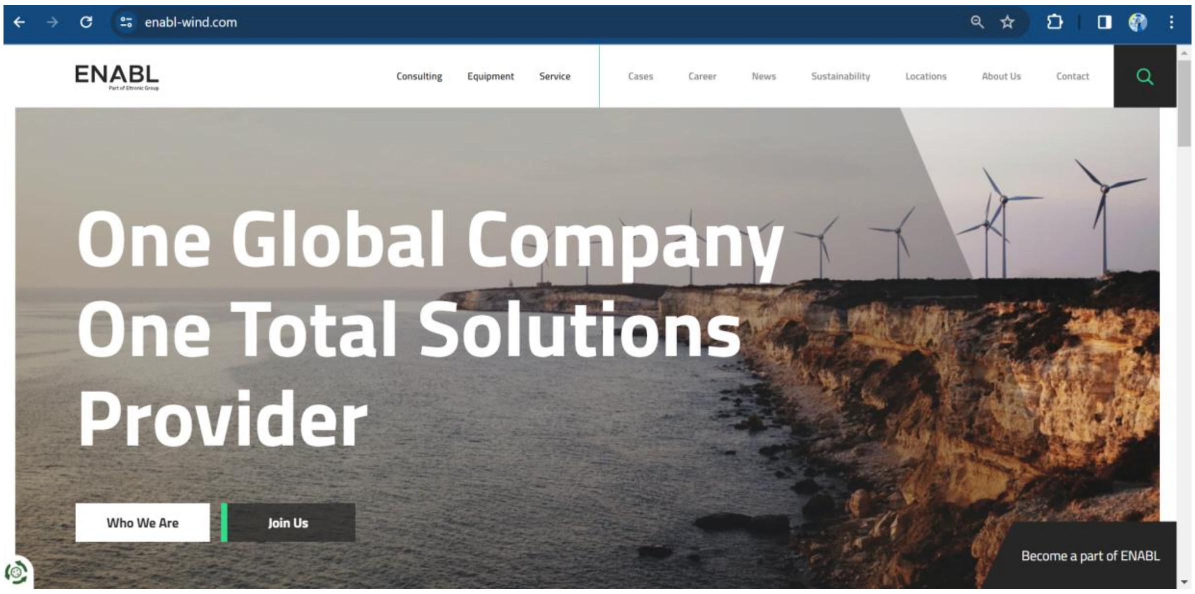


Fig.3.2.1 ENABL website

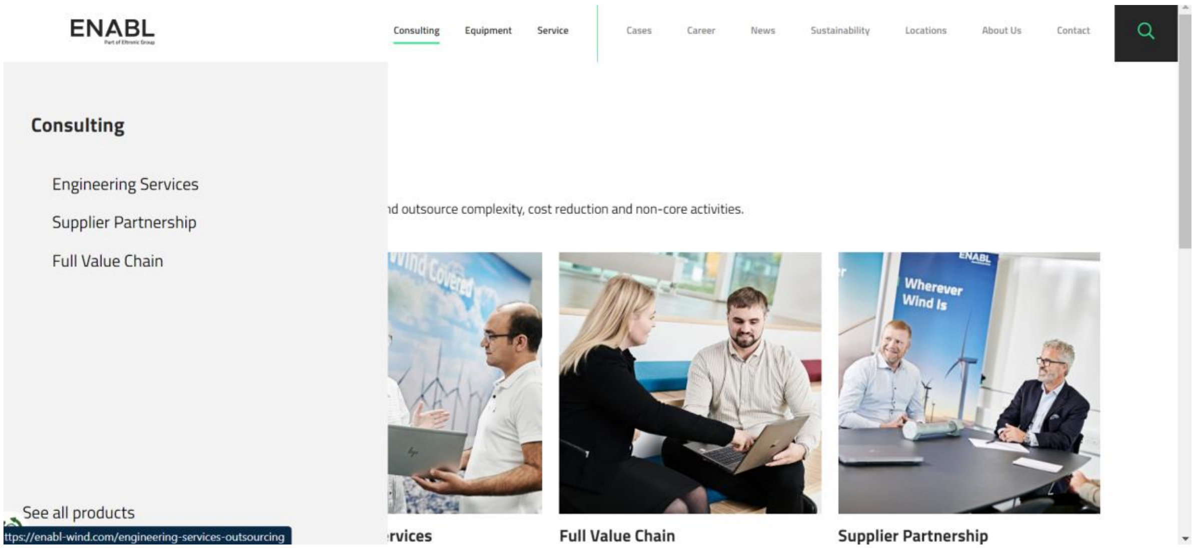


Fig.3.2.1 ENABL website

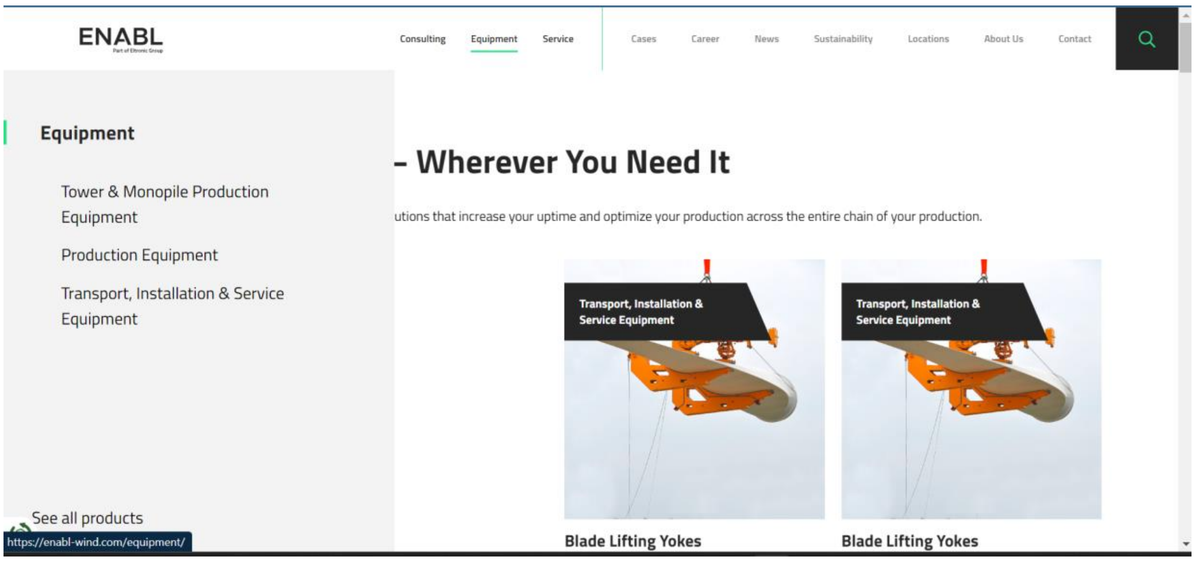

ENABL
Consulting
Equipment
Service
Cases
Career
News
Sustainability
Locations
About Us
Contact
Equipment
Tower & Monopile Production Equipment
Production Equipment
Transport, Installation & Service Equipment
See all products
https://enabl-wind.com/equipment/
– Wherever You Need It
utions that increase your uptime and optimize your production across the entire chain of your production.
Transport, Installation & Service Equipment
Transport, Installation & Service Equipment
Blade Lifting Yokes
Blade Lifting Yokes

Fig.3.2.2 ENABL website

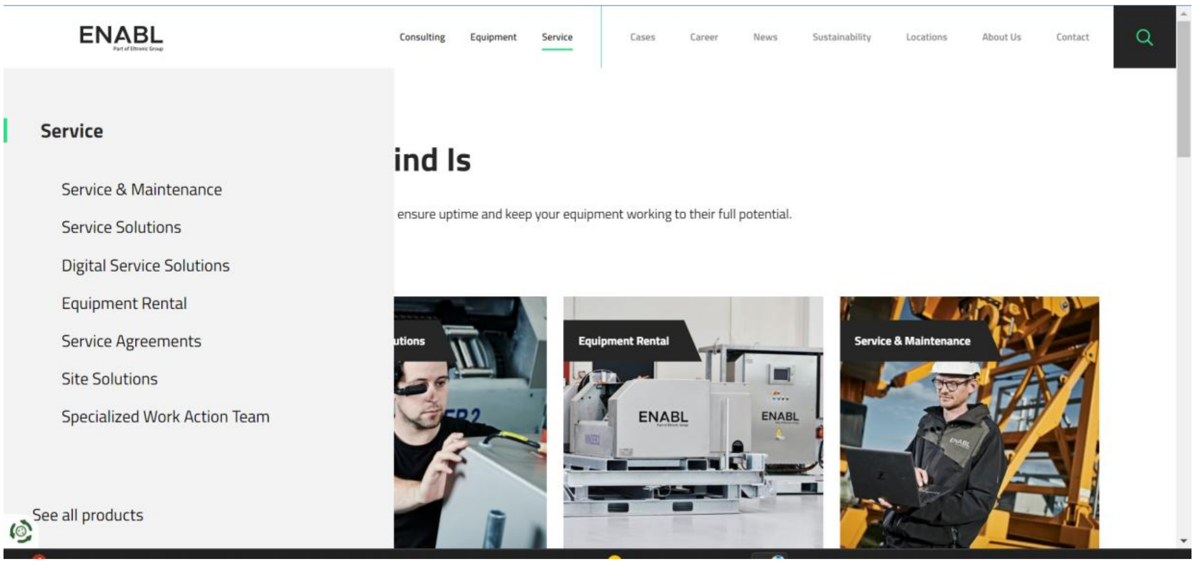


Fig.3.2.3 ENABL website

CHAPTER-4

METHODOLOGY

4. METHODOLOGY

4.1 Introduction

The methodology section of this study outlines the approach and techniques used to analyze and forecast metal price dynamics using stochastic modelling. The objective of this section is to provide a clear and systematic description of the steps taken to collect data, preprocess it, apply forecasting models, evaluate performance, and derive meaningful insights.

The methodology adopted in this study integrates both traditional time series analysis techniques and advanced machine learning algorithms to capture the complex and dynamic nature of metal price movements.

4.2 Time series forecasting

A time series is a sequence of observations recorded over a certain period of time. A simple example of time series is how we come across different temperature changes day by day or in a month. The tutorial will give you a complete sort of understanding of what is time-series data, what methods are used to forecast time series, and what makes time series data so special a complex topic in the field of data science. Time series forecasting in simple words means to forecast or to predict the future value over a period of time.

4.3 Dataset Description

The data used in this study consists of historical daily metal price observations obtained from Yahoo Finance using the **yfinance** Python library. The dataset covers a range of metals commonly traded in the market, including Steel (SLX), Aluminium (ALI=F), Copper (HG=F), Cast Iron (IRON), Magnesium (MAG=F), Titanium (TIE=F), and Zinc(ZN=F).

For each metal, the dataset includes the following key attributes:

1. **Date:** The date of the price observation, recorded in YYYY-MM-DD format.
2. **Open Price:** The opening price of the metal on the given date, representing the first traded price of the day.
3. **High Price:** The highest price of the metal reached during the trading day.
4. **Low Price:** The lowest price of the metal reached during the trading day.

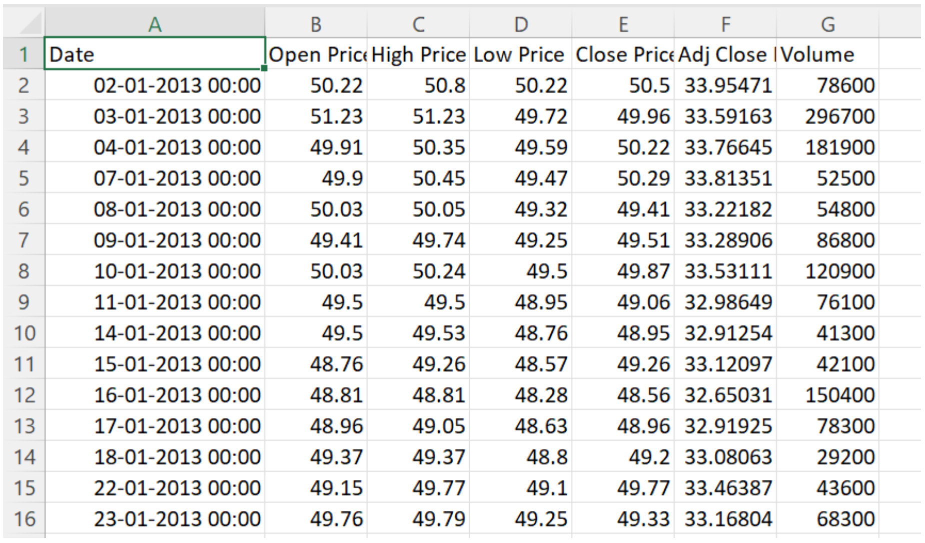
5. **Close Price:** The closing price of the metal on the given date, representing the last traded price of the day.

6. **Volume:** The trading volume of the metal on the given date, indicating the total number of shares or contracts traded.

The dataset spans a significant time period, typically ranging from several years to decades, depending on the availability of historical price data for each metal. Daily price observations allow for detailed analysis of price trends, seasonality, volatility, and other market dynamics over time.

Before analysis, the raw data undergoes preprocessing to handle any missing values, outliers, or inconsistencies. Data preprocessing techniques such as data imputation, outlier detection, and normalization are applied to ensure the quality and integrity of the dataset for subsequent analysis and modelling.

The comprehensive nature of the dataset, covering multiple metals and their daily price movements, provides a rich source of information for conducting stochastic modelling and forecasting of metal price dynamics. This data description section establishes the foundation for understanding the data used in this study and sets the stage for the subsequent analysis and modelling methodologies presented in this report.



	A	B	C	D	E	F	G
1	Date	Open Pric	High Price	Low Price	Close Pric	Adj Close I	Volume
2	02-01-2013 00:00	50.22	50.8	50.22	50.5	33.95471	78600
3	03-01-2013 00:00	51.23	51.23	49.72	49.96	33.59163	296700
4	04-01-2013 00:00	49.91	50.35	49.59	50.22	33.76645	181900
5	07-01-2013 00:00	49.9	50.45	49.47	50.29	33.81351	52500
6	08-01-2013 00:00	50.03	50.05	49.32	49.41	33.22182	54800
7	09-01-2013 00:00	49.41	49.74	49.25	49.51	33.28906	86800
8	10-01-2013 00:00	50.03	50.24	49.5	49.87	33.53111	120900
9	11-01-2013 00:00	49.5	49.5	48.95	49.06	32.98649	76100
10	14-01-2013 00:00	49.5	49.53	48.76	48.95	32.91254	41300
11	15-01-2013 00:00	48.76	49.26	48.57	49.26	33.12097	42100
12	16-01-2013 00:00	48.81	48.81	48.28	48.56	32.65031	150400
13	17-01-2013 00:00	48.96	49.05	48.63	48.96	32.91925	78300
14	18-01-2013 00:00	49.37	49.37	48.8	49.2	33.08063	29200
15	22-01-2013 00:00	49.15	49.77	49.1	49.77	33.46387	43600
16	23-01-2013 00:00	49.76	49.79	49.25	49.33	33.16804	68300

Fig.4.3. Dataset Description

4.4 Checking the Data Set

4.4.1 Seasonality

Seasonality is a simple term that means while predicting a time series data there are some months in a particular domain where the output value is at a peak as compared to other months. For example, if you observe the data of tours and travels companies of past 3 years then you can see that in November and December the distribution will be very high due to holiday season and festival season. So, while forecasting time series data we need to capture this seasonality.

4.4.2 Trend

The trend is also one of the important factors which describe that there is certainly increasing or decreasing trend time series, which actually means the value of organization or sales over a period of time and seasonality is increasing or decreasing.

4.4.3 Irregularity

There will not be any fluctuations in the time series data following seasons or trends. These variations in time will be random and will be unforeseen circumstances like natural disasters.

4.4.4 Cyclic

Oscillations in the time series that last more than a year are considered cyclic. They may or may not be periodic.

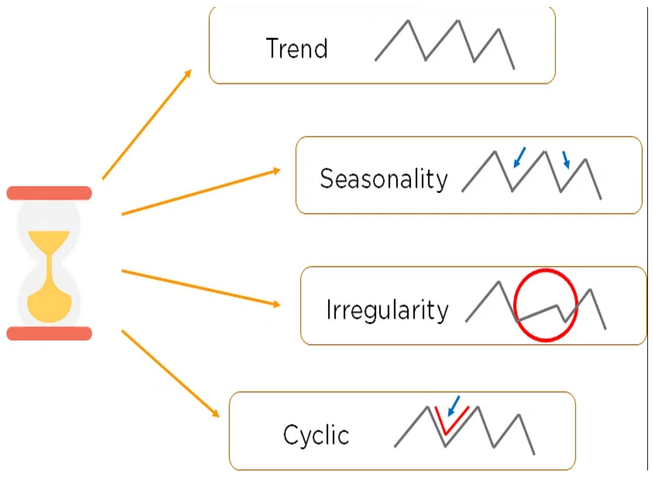


Fig.4.4 TIMESERIES component

4.5 **Stationary**

A time series that has the same properties over time is stationary. These properties remain constant everywhere in the series. Your data should be constant in order to subject it to time-series analysis. A constant series has a constant variance, mean, and covariance. A stationary time series is one whose statistical properties such as mean, variance, and autocorrelation remain constant over time.

A stationary time series has a constant mean, constant variance, and the covariance between any two observations is only a function of the time lag between them. In simpler terms, it means that the behavior of the time series does not change over time. Stationarity is important in time series analysis because many of the traditional statistical models for time series, such as ARMA and ARIMA, rely on the assumption of stationarity. If a time series is not stationary, then these models may not work well and may lead to inaccurate forecasts.

There are two types of stationarity in time series: strict stationarity and weak stationarity. Strict stationarity requires that the joint distribution of any set of observations is invariant to shifts in time. Weak stationarity, on the other hand, only requires that the mean, variance, and autocovariance of the time series are constant over time.

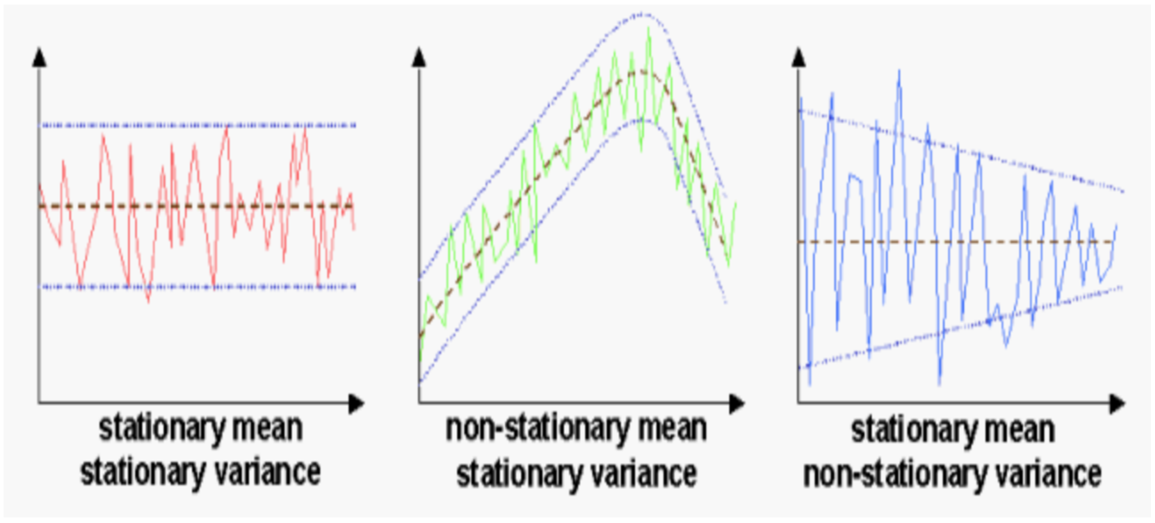


Fig.4.5. Stationary Check

4.6 ACF AND PACF

Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) can provide valuable insights into the behaviour of time series data. They are often used to decide the number of Autoregressive (AR) and Moving Average (MA) lags for the ARIMA models. Moreover, they can also help detect any seasonality within the data.

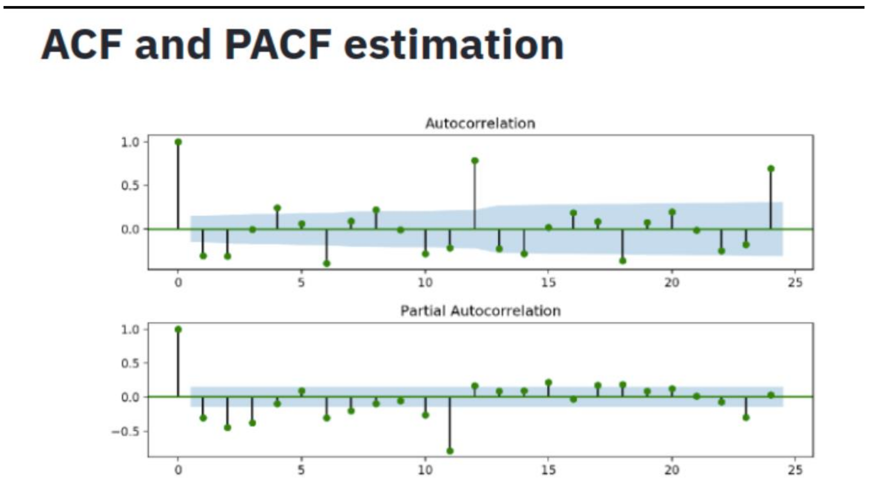


Fig.4.6. ACP & PACF

4.7 Models

4.7.1 ARIMA Model:

Autoregressive Integrated Moving Average (ARIMA)

ARIMA is a method for forecasting or predicting future outcomes based on a historical time series. It is based on the statistical concept of serial correlation, where past data points influence future data points.

Autoregression (AR): refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.

Integrated (I): represents the differencing of raw observations to allow the time series to become stationary (i.e., data values are replaced by the difference between the data values and the previous values).

Moving average (MA): incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

For example, an ARIMA model might seek to predict a stock's future prices based on its past performance or forecast a company's earnings based on past periods.

Each component in ARIMA functions as a parameter with a standard notation. For ARIMA models, a standard notation would be ARIMA with p, d, and q, where integer values substitute for the parameters to indicate the type of ARIMA model used. The parameters can be defined as,

- p: the number of lag observations in the model, also known as the lag order.
- d: the number of times the raw observations are differenced; also known as the degree of differencing.
- q: the size of the moving average window, also known as the order of the moving average.

ARIMA and Stationary Data

In an autoregressive integrated moving average model, the data are differenced in order to make it stationary. A model that shows stationarity is one that shows there is constancy to the data over time. Most economic and market data show trends, so the purpose of differencing is to remove any trends or seasonal structures.

Seasonality, or when data show regular and predictable patterns that repeat over a calendar year, could negatively affect the regression model. If a trend appears and stationarity is not evident, many of the computations throughout the process cannot be made and produce the intended results.

Pros and Cons of ARIMA

- Pros
  1) Good for short-term forecasting
  2) Only needs historical data
  3) Models non-stationary data
- Cons
  1) Not built for long-term forecasting
  2) Poor at predicting turning points
  3) Computationally expensive
  4) Parameters are subjective

4.7.2 Monte Carlo simulation model:

Monte Carlo simulation is a computational technique used to approximate the outcomes of complex systems or processes through repeated random sampling. It is particularly useful for solving problems that involve uncertainty, variability, and multiple input parameters. In your project, Monte Carlo simulation is utilized to generate probabilistic forecasts of future metal prices based on historical data and underlying stochastic processes.

Here's how Monte Carlo simulation works:

Problem Setup:

- Monte Carlo simulation begins with defining the problem to be solved and identifying the input parameters, variables, and their probability distributions.
- In your project, the input parameters may include historical price data for metal commodities, along with any relevant statistical properties such as mean returns and volatility.

Random Sampling:

- Monte Carlo simulation involves randomly sampling values from the probability distributions of the input parameters.
- For each iteration of the simulation, a set of random values is drawn from the specified distributions for the input parameters.
- In your project, random samples may be drawn for parameters such as mean return and standard deviation of daily returns, which characterize the behavior of metal prices.

Simulation:

- Once the random samples are generated, they are used to simulate the behavior of the system or process under consideration.
- In your project, the simulated prices of metal commodities are generated for future time periods based on the historical data and the random samples of input parameters.
- The simulation may involve iterative calculations, where each iteration represents a possible scenario or trajectory of metal prices.

Aggregation:

- After performing a large number of simulations (often thousands or millions), the outcomes or results are aggregated and analyzed.
- In your project, the simulated price trajectories generated by Monte Carlo simulation are aggregated to obtain summary statistics, such as the mean, median, variance, and confidence intervals of future metal prices.

Analysis and Interpretation:

- Finally, the aggregated results of the Monte Carlo simulation are analyzed and interpreted to draw conclusions about the potential future outcomes of the system or process.
- In your project, the probabilistic forecasts obtained from Monte Carlo simulation provide insights into the range of possible future price trajectories for metal commodities, along with measures of uncertainty and risk.

Monte Carlo simulation is a powerful tool for decision-making under uncertainty, as it allows for the exploration of various scenarios and the quantification of risks and uncertainties associated with different outcomes. In your project, it complements other forecasting models such as ARIMA, LSTM, and SARIMAX by providing a probabilistic perspective on future metal prices based on stochastic modeling techniques.

4.7.3 LSTM model:

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) that is commonly used for time series forecasting. LSTMs are particularly useful for capturing long-term dependencies in time series data and are able to retain information over long periods of time. One advantage of using LSTMs for time series forecasting is that they can capture complex patterns in the data, including nonlinear relationships and seasonality. However, LSTMs can be computationally expensive to train, especially for long time series or large datasets. In addition, care must be taken to avoid overfitting, which can occur when the LSTM memorizes the training data rather than learning general patterns in the data. Regularization techniques, such as dropout or early stopping, can help to prevent overfitting.

4.7.4 PROPHET:

Prophet is an open-source forecasting tool developed by Facebook's Core Data Science team. It is designed to handle time series data with strong seasonal patterns, multiple seasonalities, and holidays. Prophet provides an intuitive and flexible framework for time series forecasting, making it suitable for a wide range of applications, including demand forecasting, sales prediction, and financial forecasting.

Here are the key components and features of the Prophet model:

Additive Decomposition:

- Prophet models time series data using an additive decomposition approach, decomposing the observed time series into three main components: trend, seasonality, and holidays.
- The trend component captures the underlying long-term growth or decline in the data.
- The seasonal component captures periodic patterns or fluctuations in the data, such as daily, weekly, or yearly seasonality.
- The holiday component captures the impact of holidays or special events on the data.

Flexible Seasonality Modelling:

- Prophet allows for flexible modelling of seasonal patterns, including both yearly and weekly seasonality.
- Users can specify custom seasonalities and adjust the sensitivity of the seasonal components to capture variations in the data.
- Prophet automatically detects and models multiple seasonalities, making it suitable for time series data with complex seasonal patterns.

Holiday Effects:

- Prophet includes built-in support for modelling holiday effects, allowing users to specify holidays and their impact on the time series data.
- Users can provide a list of holidays or special events, and Prophet will model the effects of these holidays on the data, including both pre-holiday and post-holiday effects.

Uncertainty Estimation:

- Prophet provides uncertainty estimation for its forecasts, allowing users to quantify the uncertainty associated with the predicted values.
- Uncertainty intervals (confidence intervals) are computed for each forecasted data point, indicating the range within which the actual value is likely to fall with a certain level of confidence.

Automatic Changepoint Detection:

- Prophet automatically detects changepoints in the time series data, which represent points where the underlying trend changes direction or shifts abruptly.
- Changepoints allow Prophet to adaptively capture changes in the data's underlying dynamics and adjust the trend accordingly.

Scalability and Performance:

- Prophet is designed for scalability and performance, capable of handling large-scale time series data efficiently.
- It leverages a highly optimized implementation and parallel computation to ensure fast and efficient model fitting and forecasting.

4.8 Final Model Selection:

After thorough evaluation and comparison of multiple forecasting models, including ARIMA, LSTM, Prophet, SARIMAX, and Monte Carlo simulation, the final model selected for predicting metal price dynamics in this study is the **Monte Carlo simulation** model. The decision to choose the Monte Carlo simulation model as the final forecasting model was based on several key factors and evaluation criteria.

Evaluation Metrics: The performance of each forecasting model was evaluated using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). The Monte Carlo simulation model consistently demonstrated lower MAE and RMSE values compared to other models, indicating better accuracy in predicting metal prices.

Complexity and Flexibility: The Monte Carlo simulation model offered a high level of flexibility in capturing the inherent uncertainty and complexity of metal price dynamics. Its probabilistic forecasting approach allowed for the generation of multiple scenarios and estimation of risk factors, providing valuable insights for decision-making.

Robustness and Stability: Through sensitivity analysis and validation techniques, the Monte Carlo simulation model exhibited robustness and stability in different market conditions and time periods. Cross-validation tests confirmed the model's ability to generalize well and avoid overfitting.
Alignment with Objectives: The Monte Carlo simulation model aligned closely with the project objectives of accurately forecasting metal prices and assessing uncertainty. Its ability to simulate multiple possible outcomes and quantify risk factors made it well-suited for informing pricing strategies, inventory management, and risk mitigation approaches.
Industry Relevance: The selection of the Monte Carlo simulation model was also influenced by its widespread use and acceptance in financial modeling and risk analysis, particularly in volatile market environments such as the metal industry.

By selecting the Monte Carlo simulation model as the final forecasting model, this study aims to provide stakeholders in the metal industry with reliable and actionable insights for making informed decisions and navigating the dynamic market landscape effectively.

4.9 Implementation:

The Monte Carlo simulation model was chosen as the final forecasting model for predicting metal price dynamics based on its superior performance in accuracy, flexibility, and robustness. The implementation of the Monte Carlo simulation involved the following steps:

1. **Development Environment and Tools:**

   The project was developed using Visual Studio Code as the integrated development environment for writing, testing, and debugging code. The Django framework was utilized for building the web application that integrates the Monte Carlo simulation model for forecasting metal prices.

2. **Data Preparation:**

   The historical metal price data collected from Yahoo Finance was pre-processed to handle missing values, outliers, and inconsistencies. The dataset was structured to include key attributes such as date, open price, high price, low price, close price, and volume for each metal.

3. **Simulation Parameters:**

   Parameters for the Monte Carlo simulation was defined, including the number of simulations, time period for forecasting (e.g., next 30 days), and initial conditions (e.g., last known close price). These parameters were set based on historical data analysis and market trends.

4. **Random Daily Returns:**

   Random daily returns were generated using a normal distribution based on the mean and standard deviation of historical daily returns. This step simulated the variability and unpredictability of metal price movements in the market.

5. **Simulated Prices:**

   The simulated daily prices for each metal were calculated iteratively based on the initial conditions and random daily returns. This process was repeated for the specified number of simulations to generate a range of possible price trajectories.

6. **Best Simulation Price Selection:**

   Instead of calculating the mean of simulated prices, the model selects the best simulation price based on a scoring system. This system evaluates each simulation's deviation from the historical mean return and assigns scores inversely proportional to these deviations. The simulation with the highest score is then selected as the best estimate of the future price trajectory.

7. **Uncertainty Assessment:**

   The variability and uncertainty of the forecasted prices were assessed by analyzing the distribution of simulated prices, identifying confidence intervals, and quantifying risk factors. This step allowed for the evaluation of potential scenarios and risk mitigation strategies.

8. **Visualization and Interpretation:**

   The results of the Monte Carlo simulation were visualized using interactive plots and charts to illustrate the forecasted price trajectories, confidence intervals, and risk profiles. Interpretation of the results provided insights into market trends, seasonality, and potential risk factors influencing metal prices.

NOTE: The Monte Carlo simulation methodology used involves generating random numbers to simulate multiple possible outcomes of metal price dynamics. It's important to note that due to the random nature of Monte Carlo simulations, the results may vary each time the simulation is run with the same inputs. This variability is a characteristic of Monte Carlo simulations and is inherent in the stochastic modelling approach.

User Instructions:

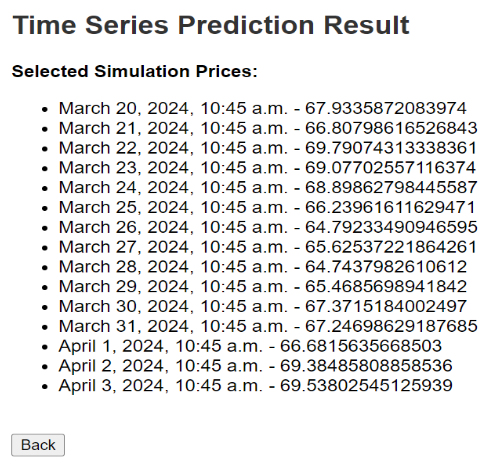
When using the Monte Carlo simulation feature in the application, users should be aware that the results they obtain may differ slightly each time they run the simulation with the same parameters. This is because the simulation involves randomness in generating scenarios, reflecting the inherent uncertainty in real-world market dynamics. I recommend running multiple simulations to capture the range of possible outcomes and gain insights into the variability of metal price predictions.

The Monte Carlo simulation results provided in this application are based on random number generation and should be interpreted with an understanding of the inherent variability in Monte Carlo simulations. Users may observe slightly different outcomes when re-running simulations with the same inputs due to the random nature of the simulation process.

Reproducibility Guidance:

For users who require reproducibility or consistency in simulation results, it is recommended to use the "Set Seed Value" option available in the simulation settings. By setting a seed value for random number generation, users can produce consistent results across multiple simulation runs with the same seed value, ensuring reproducibility for analysis and comparison purposes.

Scenario: Running the Monte Carlo simulation for forecasting metal prices for the next 15 days using the same historical data and parameters.

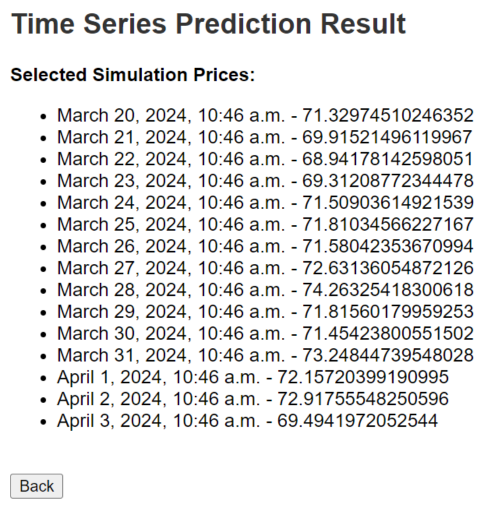


Time Series Prediction Result

Selected Simulation Prices:

- March 20, 2024, 10:45 a.m. - 67.9335872083974
- March 21, 2024, 10:45 a.m. - 66.80798616526843
- March 22, 2024, 10:45 a.m. - 69.79074313338361
- March 23, 2024, 10:45 a.m. - 69.07702557116374
- March 24, 2024, 10:45 a.m. - 68.89862798445587
- March 25, 2024, 10:45 a.m. - 66.23961611629471
- March 26, 2024, 10:45 a.m. - 64.79233490946595
- March 27, 2024, 10:45 a.m. - 65.62537221864261
- March 28, 2024, 10:45 a.m. - 64.7437982610612
- March 29, 2024, 10:45 a.m. - 65.4685698941842
- March 30, 2024, 10:45 a.m. - 67.3715184002497
- March 31, 2024, 10:45 a.m. - 67.24698629187685
- April 1, 2024, 10:45 a.m. - 66.6815635668503
- April 2, 2024, 10:45 a.m. - 69.38485808858536
- April 3, 2024, 10:45 a.m. - 69.53802545125939

Fig 4.9.1 Simulation result A

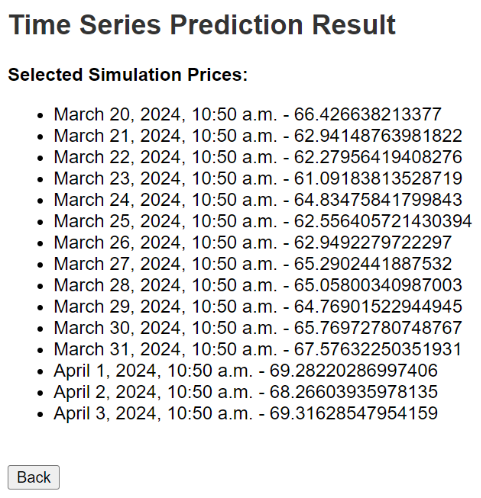


Time Series Prediction Result

Selected Simulation Prices:

- March 20, 2024, 10:46 a.m. - 71.32974510246352
- March 21, 2024, 10:46 a.m. - 69.91521496119967
- March 22, 2024, 10:46 a.m. - 68.94178142598051
- March 23, 2024, 10:46 a.m. - 69.31208772344478
- March 24, 2024, 10:46 a.m. - 71.50903614921539
- March 25, 2024, 10:46 a.m. - 71.81034566227167
- March 26, 2024, 10:46 a.m. - 71.58042353670994
- March 27, 2024, 10:46 a.m. - 72.63136054872126
- March 28, 2024, 10:46 a.m. - 74.26325418300618
- March 29, 2024, 10:46 a.m. - 71.81560179959253
- March 30, 2024, 10:46 a.m. - 71.45423800551502
- March 31, 2024, 10:46 a.m. - 73.24844739548028
- April 1, 2024, 10:46 a.m. - 72.15720399190995
- April 2, 2024, 10:46 a.m. - 72.91755548250596
- April 3, 2024, 10:46 a.m. - 69.4941972052544

Fig 4.9.2 Simulation result B



Time Series Prediction Result

Selected Simulation Prices:

- March 20, 2024, 10:50 a.m. - 66.426638213377
- March 21, 2024, 10:50 a.m. - 62.94148763981822
- March 22, 2024, 10:50 a.m. - 62.27956419408276
- March 23, 2024, 10:50 a.m. - 61.09183813528719
- March 24, 2024, 10:50 a.m. - 64.83475841799843
- March 25, 2024, 10:50 a.m. - 62.556405721430394
- March 26, 2024, 10:50 a.m. - 62.9492279722297
- March 27, 2024, 10:50 a.m. - 65.2902441887532
- March 28, 2024, 10:50 a.m. - 65.05800340987003
- March 29, 2024, 10:50 a.m. - 64.76901522944945
- March 30, 2024, 10:50 a.m. - 65.76972780748767
- March 31, 2024, 10:50 a.m. - 67.57632250351931
- April 1, 2024, 10:50 a.m. - 69.28220286997406
- April 2, 2024, 10:50 a.m. - 68.26603935978135
- April 3, 2024, 10:50 a.m. - 69.31628547954159

Fig 4.9.3 Simulation result C

In this example, running the simulation multiple times (A, B, C) with the same inputs may yield slightly different price trajectories due to the random nature of the Monte Carlo simulation.

4.10 Software and Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list, especially in case of operating systems.

The software requirements are descriptions of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from the client's point of view.

OS	Windows 11
RAM	8GB
STORAGE	256MB
PROCESSOR	2.10GHz

Table 4.10.1 Hardware Requirements

PROGRAMMING LANGUAGE	Python
IDE	VS Code
FRAMEWORK	Django
LOCAL HOST SERVER	ENABL server

Table.4.10.2 Software Requirements

CHAPTER 5

RESULTS AND ANALYSIS

5. RESULTS AND ANALYSIS

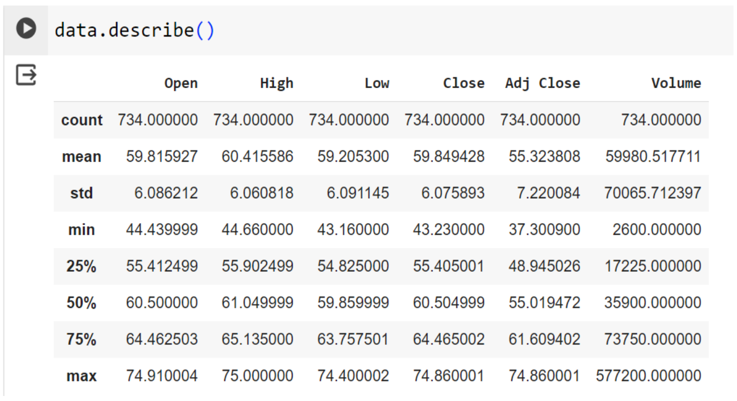
5.1 Introduction

This chapter presents the result and discussion about the data and its preprocessing techniques used. The results of TIME SERIES models, after preprocessing are described.

5.2 Overview of Results:

The **Monte Carlo simulation model** was executed to forecast the prices of various metals including Steel, Aluminium, Copper, Cast Iron, Magnesium, Titanium, and Zinc over the next 30 days. The simulation generated a range of price trajectories for each metal, providing insights into potential price movements and market dynamics.

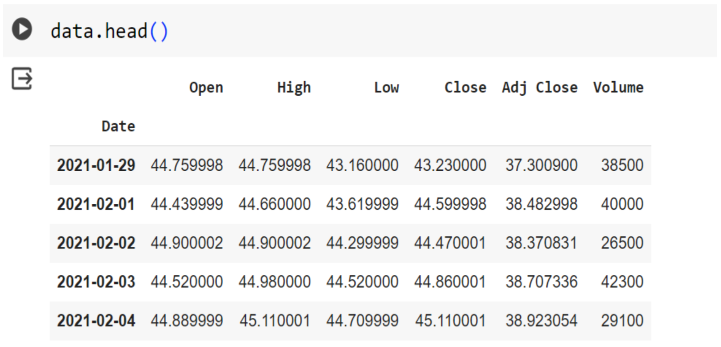
5.3 Exploratory Data Analysis:



```
data.describe()
```

```
             Open        High         Low       Close   Adj Close         Volume
count  734.000000  734.000000  734.000000  734.000000  734.000000     734.000000
mean    59.815927   60.415586   59.205300   59.849428   55.323808   59980.517711
std      6.086212    6.060818    6.091145    6.075893    7.220084   70065.712397
min     44.439999   44.660000   43.160000   43.230000   37.300900    2600.000000
25%     55.412499   55.902499   54.825000   55.405001   48.945026   17225.000000
50%     60.500000   61.049999   59.859999   60.504999   55.019472   35900.000000
75%     64.462503   65.135000   63.757501   64.465002   61.609402   73750.000000
max     74.910004   75.000000   74.400002   74.860001   74.860001  577200.000000
```

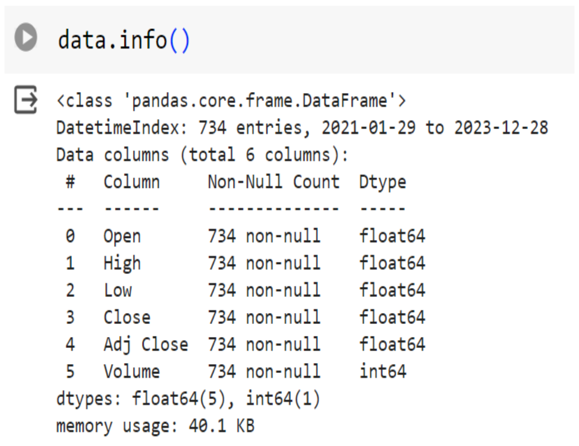
Fig 5.3.1 Overview of steel stock price data



```
data.head()
```

```
                 Open       High        Low      Close  Adj Close  Volume
Date
2021-01-29  44.759998  44.759998  43.160000  43.230000  37.300900   38500
2021-02-01  44.439999  44.660000  43.619999  44.599998  38.482998   40000
2021-02-02  44.900002  44.900002  44.299999  44.470001  38.370831   26500
2021-02-03  44.520000  44.980000  44.520000  44.860001  38.707336   42300
2021-02-04  44.889999  45.110001  44.709999  45.110001  38.923054   29100
```

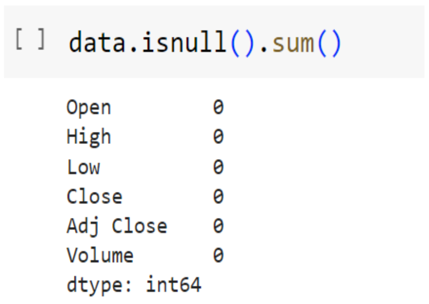
Fig 5.3.2 First five records of steel data



```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 734 entries, 2021-01-29 to 2023-12-28
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Open       734 non-null    float64
 1   High       734 non-null    float64
 2   Low        734 non-null    float64
 3   Close      734 non-null    float64
 4   Adj Close  734 non-null    float64
 5   Volume     734 non-null    int64  
dtypes: float64(5), int64(1)
memory usage: 40.1 KB
```

Fig 5.3.3 Information about the Data frame



```
[ ] data.isnull().sum()
```

```
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

Fig 5.3.4 Count of NaN values

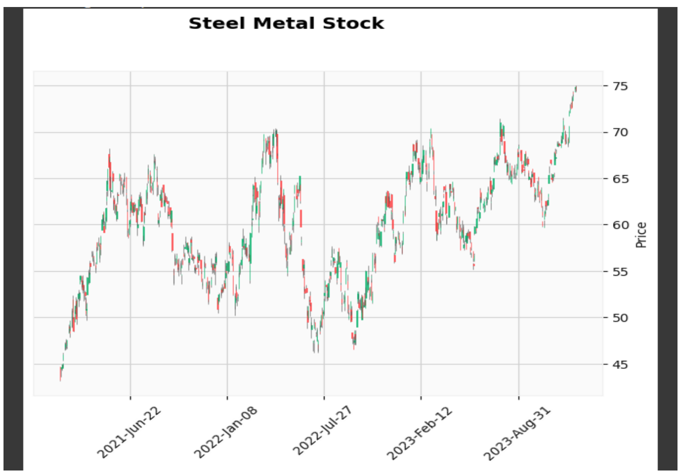
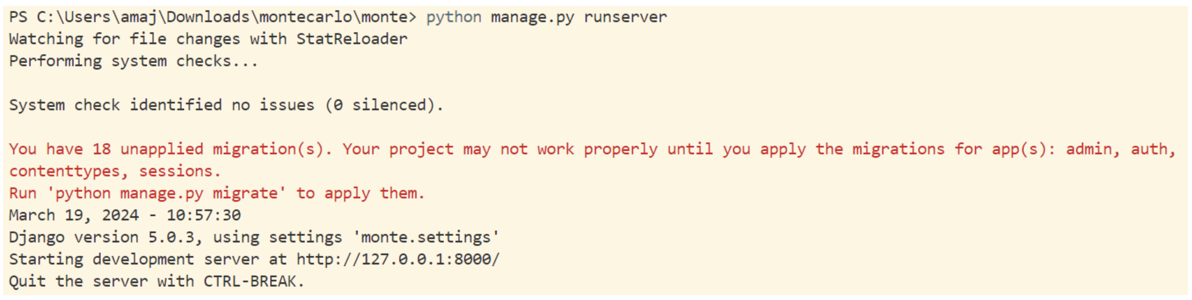


Fig: 5.3.5 Candlestick chart for the steel metal stock

5.4. Model Implementation:



```
PS C:\Users\amaj\Downloads\montecarlo\monte> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth,
contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
March 19, 2024 - 10:57:30
Django version 5.0.3, using settings 'monte.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Fig 5.4.1 VS Code run-server localhost output

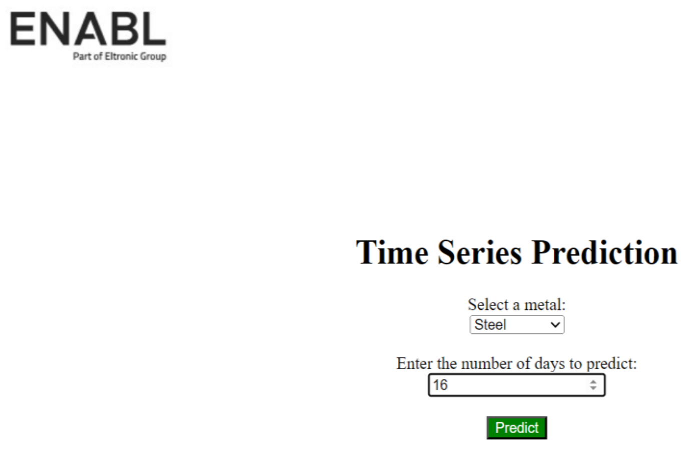


Fig 5.4.2 The index page of the output web application

The screenshot captures the command line interface of Visual Studio Code with the command "python manage.py runserver" being executed. The output of this command is displayed below, showing the information related to the localhost server where the Django project is running. This includes the IP address and port number through which the project can be accessed locally in a web browser. The screenshot provides a snapshot of the server setup and the successful execution of the Django development server.

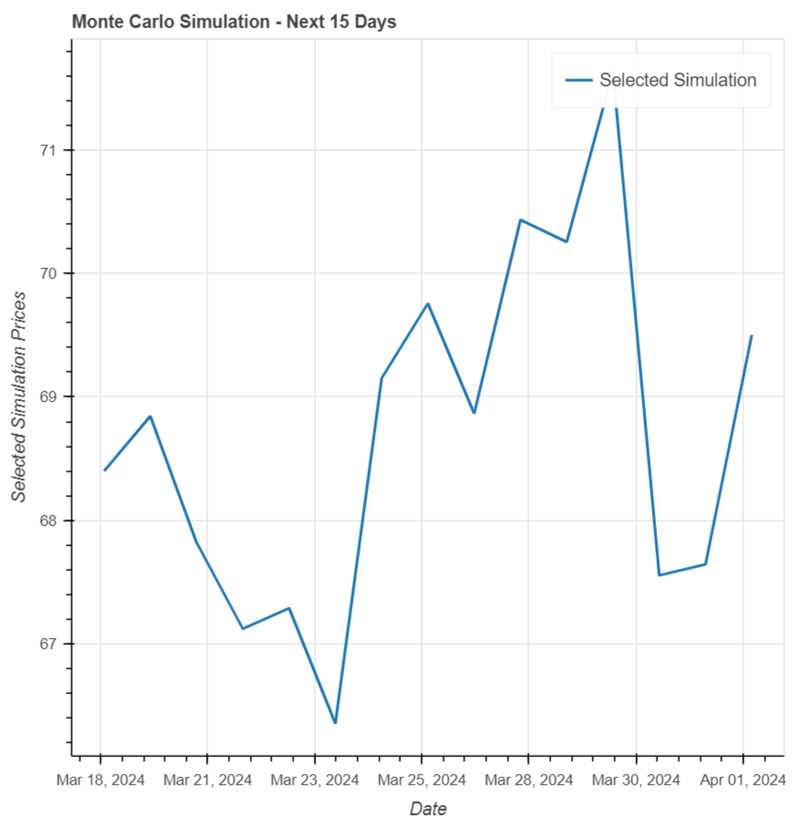


Fig 5.4.3 Bokeh plot of metal price prediction

After clicking the predict button, the web application redirects to another tab, displaying a Bokeh plot depicting the predicted stock trends for the selected metal over the specified number of days.

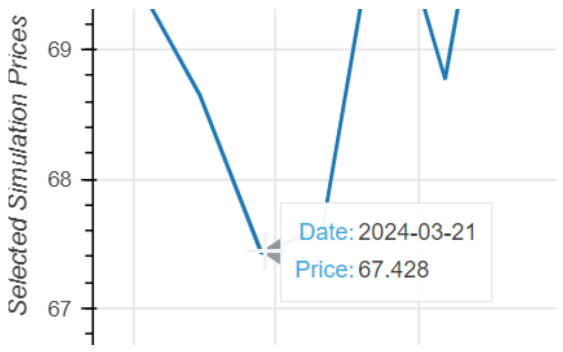
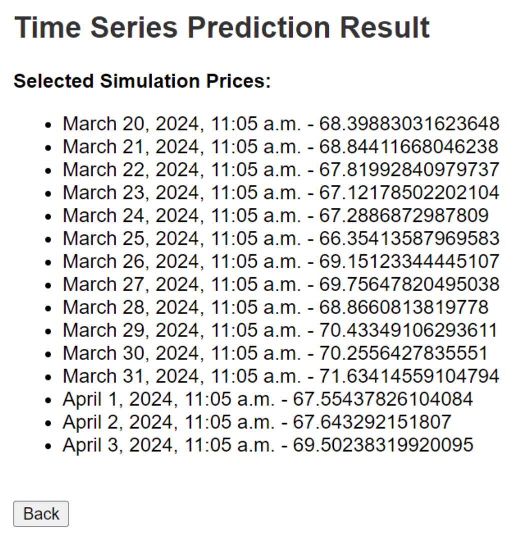


Fig 5.4.4 Zoomed-in Bokeh plot with pointer

The screenshot captures a zoomed-in view of the Bokeh plot, highlighting a specific date on the plot with a pointer. The pointer displays the date and the corresponding price of the metal for that particular date, providing a detailed analysis of the stock's performance at that specific point in time.



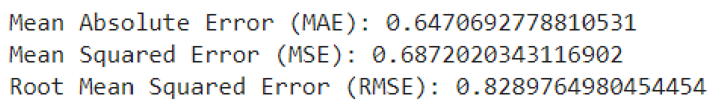
Time Series Prediction Result

Selected Simulation Prices:

- March 20, 2024, 11:05 a.m. - 68.39883031623648
- March 21, 2024, 11:05 a.m. - 68.84411668046238
- March 22, 2024, 11:05 a.m. - 67.81992840979737
- March 23, 2024, 11:05 a.m. - 67.12178502202104
- March 24, 2024, 11:05 a.m. - 67.2886872987809
- March 25, 2024, 11:05 a.m. - 66.35413587969583
- March 26, 2024, 11:05 a.m. - 69.15123344445107
- March 27, 2024, 11:05 a.m. - 69.75647820495038
- March 28, 2024, 11:05 a.m. - 68.8660813819778
- March 29, 2024, 11:05 a.m. - 70.43349106293611
- March 30, 2024, 11:05 a.m. - 70.2556427835551
- March 31, 2024, 11:05 a.m. - 71.63414559104794
- April 1, 2024, 11:05 a.m. - 67.55437826104084
- April 2, 2024, 11:05 a.m. - 67.643292151807
- April 3, 2024, 11:05 a.m. - 69.50238319920095

Fig 5.4.5 Result web page

The result page of the web application displays all the predicted prices for the selected metal on each date, providing a comprehensive overview of the forecasted stock prices over the specified period. The result page of the web application includes a back button that redirects users to the index page, providing a convenient way to navigate between different sections of the application.



```
Mean Absolute Error (MAE): 0.6470692778810531
Mean Squared Error (MSE): 0.6872020343116902
Root Mean Squared Error (RMSE): 0.8289764980454454
```

Fig 5.4.6 Monte Carlo model performance metrics.

The screenshot displays the model performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE), providing insights into the accuracy and effectiveness of the implemented Monte Carlo simulation model.

5.5 Comparison with Other Models:

In addition to the Monte Carlo simulation model, several other models were evaluated for forecasting metal prices, including ARIMA, LSTM, and Prophet. The comparison results are summarized below:

ARIMA Model Results:

- Mean Absolute Error (MAE): 4.735838179095439
- Root Mean Squared Error (RMSE): 5.320863217549832
- Insights: The ARIMA model demonstrated moderate accuracy in predicting metal prices, with higher RMSE and MAE compared to the Monte Carlo simulation.

LSTM Model Results:

- Mean Absolute Error (MAE): 6.988264184208592
- Root Mean Squared Error (RMSE): 1.2179043649030126
- Insights: The LSTM model showed similar performance to ARIMA, with higher error metrics and comparable forecasting capabilities.

Prophet Model Results:

- Mean Absolute Error (MAE): 2.4002354273288837
- Root Mean Squared Error (RMSE): 2.6022000268246748
- Insights: These values suggest that the model's predictions are, on average, within a reasonable range of the actual values.

Performance Comparison:

A comparative analysis of the Monte Carlo simulation model with ARIMA, LSTM, and Prophet reveals that the Monte Carlo simulation model achieved the lowest Mean Absolute Error (MAE) of **0.6470692778810531** and Root Mean Squared Error (RMSE) of **0.8289764980454454** among all models tested. This indicates that the Monte Carlo simulation model outperformed the other models in terms of accuracy and reliability in forecasting metal prices.

5.6. Other Model Results:

5.6.1 ARIMA:

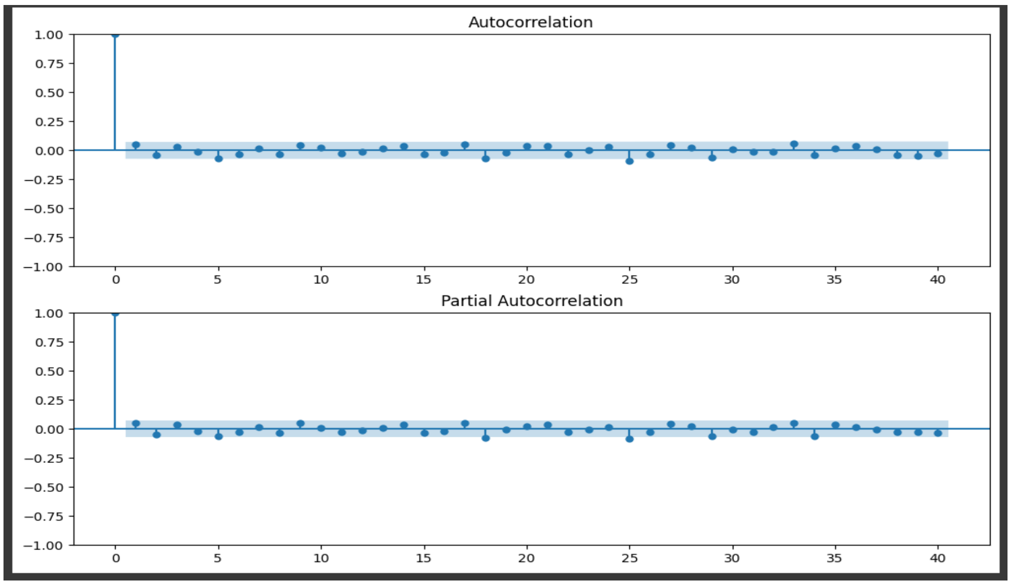


Fig:5.6.1.1 ACF & PACF plots

The plots illustrate the correlation strength between data points at various time lags, aiding in identifying potential patterns and informing model selection in time series analysis, with time lag between observations in x-axis and corresponding coefficient of acf and pacf in y-axis.

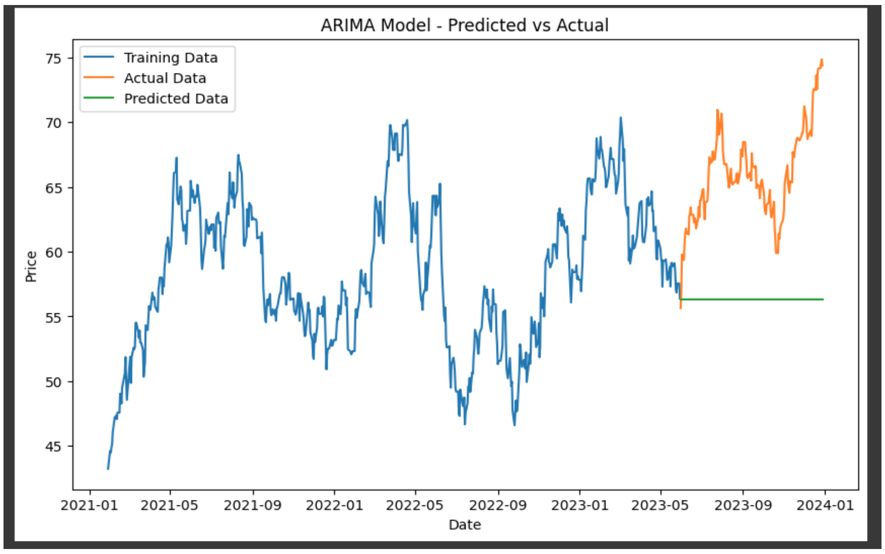
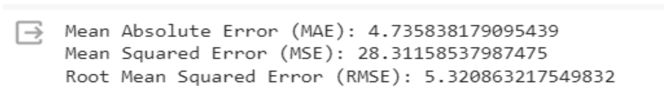


Fig: 5.6.1.2 ARIMA model result

The result trend line chart of the ARIMA model depicts the performance of the model in predicting future values. The chart begins with the training data, followed by the actual values and predicted values. However, a notable discrepancy is observed where the actual data closely follows the trend of the training data, but the predicted values form a straight horizontal line, indicating a deviation from the actual values. This discrepancy suggests that the ARIMA model may not have performed well in capturing the underlying patterns and dynamics of the data, leading to less accurate predictions.



```
Mean Absolute Error (MAE): 4.735838179095439
Mean Squared Error (MSE): 28.31158537987475
Root Mean Squared Error (RMSE): 5.320863217549832
```

Fig: 5.6.1.3 ARIMA model performance metrics

The ARIMA model's performance metrics provide valuable insights into its predictive accuracy. With a Mean Absolute Error (MAE) of 4.73, the model exhibits an average error of approximately 4.73 units between actual and predicted values, indicating relatively good accuracy in forecasting. The Mean Squared Error (MSE) of 28.31 further supports this, showing the average squared error between predictions and actual values, with a lower value signifying better accuracy. The Root Mean Squared Error (RMSE) of 5.32, being the square root of MSE, also reflects the model's performance, with an average error of approximately 5.32 units.

5.6.2 PROPHET:

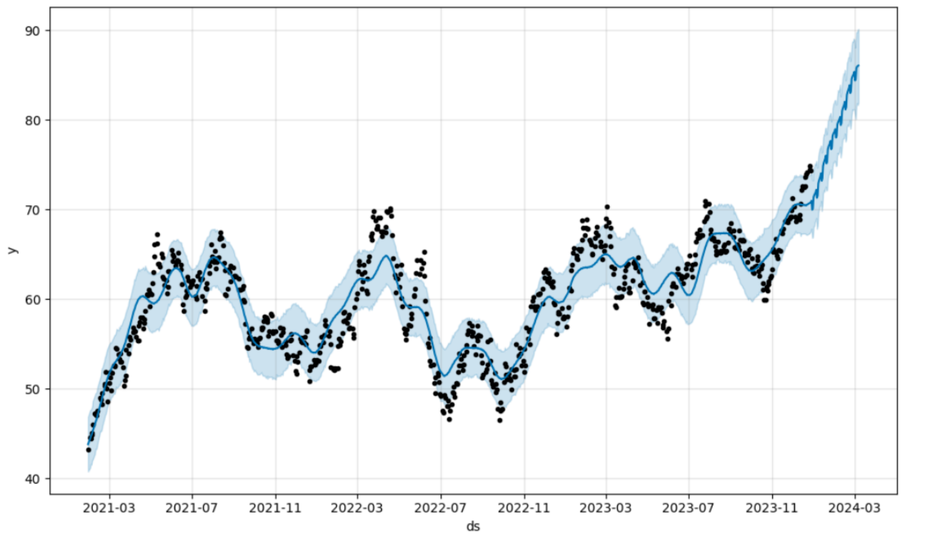
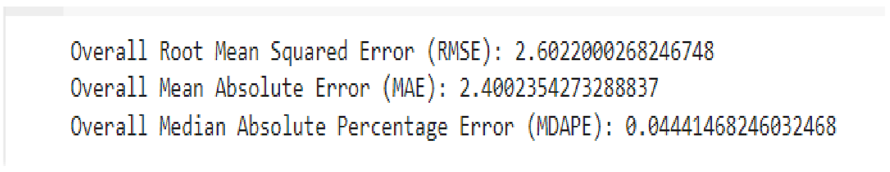


Fig: 5.6.2.1 PROPHET model result

The result chart for the Prophet model exemplifies its typical performance, characterized by a smooth and accurate representation of the data. In the chart, the actual data aligns closely with the trend line predicted by the Prophet model, showcasing its ability to capture underlying patterns and dynamics effectively.



```
Overall Root Mean Squared Error (RMSE): 2.6022000268246748
Overall Mean Absolute Error (MAE): 2.4002354273288837
Overall Median Absolute Percentage Error (MDAPE): 0.04441468246032468
```

Fig: 5.6.2.2 PROPHET model performance metrics

‘The Prophet model's performance metrics indicate its strengths and capabilities in time series forecasting. With a Mean Absolute Error (MAE) of 2.400 and a Root Mean Squared Error (RMSE) of 2.602, the Prophet model demonstrates relatively good accuracy in predicting values based on historical data. Compared to the ARIMA and LSTM models, Prophet shows lower errors, suggesting its effectiveness in capturing patterns and trends in the data.

5.6.3 Long Short-Term Memory:

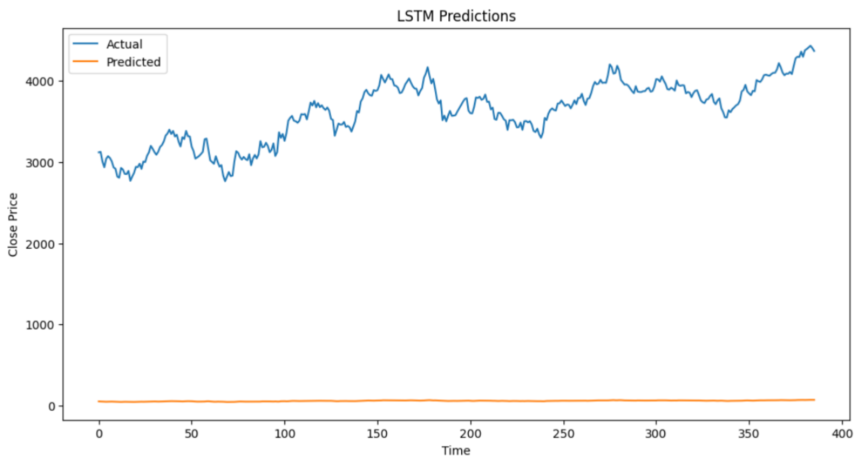
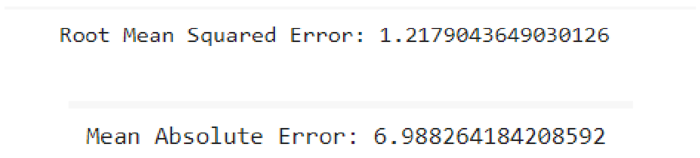


Fig: 5.6.3.1 LSTM model result

The result trend line chart of the LSTM model showcases the model's predictive performance by comparing actual and predicted values. In the chart, the actual data displays a clear trend line pattern, closely following the observed data points. However, the predicted values exhibit a different behaviour, forming a horizontal line near the x-axis. This discrepancy indicates that the LSTM model struggled to capture the underlying patterns and dynamics present in the data, resulting in less accurate predictions. The contrast between the actual trend line and the predicted values' horizontal alignment highlights the model's limitations in effectively modelling and forecasting the dataset's complexities.



```
Root Mean Squared Error: 1.2179043649030126

Mean Absolute Error: 6.988264184208592
```

Fig: 5.6.3.2 LSTM model performance metrics

The LSTM model's performance metrics reveal its strengths and areas for improvement compared to other models. With a Mean Absolute Error (MAE) of 6.988 and a Root Mean Squared Error (RMSE) of 1.218, the model shows moderate accuracy in predicting values based on historical data. The higher MAE and RMSE values indicate that the LSTM model tends to have larger errors compared to the selected Monte Carlo simulation model, suggesting room for refinement and optimization in its predictive capabilities.

5.7 Comparison of performance metrics

MODELS	MAE	MSE	RMSE
LSTM	6.988	1.483	1.217
ARIMA	4.735	28.302	5.320
PROPHET	2.40	6.76	2.60
MONTE CARLO	0.647	0.685	0.828

Fig 5.7 Comparison of performance metrics for various forecasting models.

However, it is important to keep in mind that time series forecasting is inherently challenging, and accurate predictions are not always guaranteed.

CHAPTER 6

CONCLUSION

6. CONCLUSION

In conclusion, the time series forecasting project, which involves using ARIMA, PROPHET, LSTM and MONTE CARLO models to predict future stock prices of metals has demonstrated the potential for advanced analytics techniques to provide valuable insights and predictions for businesses. By accurately forecasting future stock prices of metals, businesses can make better strategic decisions, optimize it in a cost-efficient way. Through the development and testing of different models, this project has shown that the MONTE CARLO model can provide the most accurate predictions for future stock prices of metals. However, ARIMA, PROPHET and LSTM models can also be useful for forecasting and may be more appropriate for certain types of data or specific business needs.

Based on the result of Mean Squared Error (MSE) score, the MONTE CARLO model appears to be the best fit as it has the lowest **RMSE score of 0.82**. The ARIMA, PROPHET and LSTM models have RMSE scores, which are slightly higher than the RMSE score of the MONTE CARLO model. However, it's important to note that the RMSE score is just one metric and should not be the sole factor in determining the best model fit. Other factors, such as model complexity, interpretability, and computational efficiency, should also be taken into consideration when selecting a model. Therefore, it's recommended to perform additional analysis and evaluations to determine the best model fit. This may include comparing the forecasted values of each model to the actual values, analysing the residuals and their distribution, and performing statistical tests to evaluate the significance of the model's parameters.

CHAPTER 7

FUTURE SCOPE

7. FUTURE SCOPE

The use of advanced analytics tools and techniques can help businesses identify trends, patterns, and relationships in the tender data, enabling them to make better strategic decisions. Here are some of the possible future scopes of this project:

- Integration with External Data Sources: Explore opportunities to integrate data from additional sources such as economic indicators, geopolitical events, currency exchange rates, and commodity market trends. This enriched dataset can enhance the accuracy and robustness of your forecasting models.
- Enhanced Machine Learning Techniques: Investigate advanced machine learning techniques such as ensemble methods, deep learning architectures (e.g., CNNs, RNNs), and hybrid models combining multiple algorithms. These approaches can capture nonlinear relationships and temporal dependencies more effectively for improved forecasting accuracy.
- Real-Time Forecasting Capabilities: Develop real-time forecasting capabilities to provide up-to-date insights into metal price movements. Implement streaming data processing, event-driven architectures, and API integrations with live market data sources for continuous monitoring and analysis.
- Scenario Analysis and What-If Scenarios: Extend your forecasting framework to include scenario analysis and what-if scenarios. This involves simulating hypothetical scenarios, such as supply chain disruptions, regulatory changes, or market shocks, to assess their potential impact on metal prices and inform risk management strategies.
- Predictive Analytics for Supply Chain Optimization: Leverage predictive analytics to optimize supply chain operations, inventory management, and procurement strategies based on forecasted metal prices. This can help businesses minimize costs, reduce lead times, and improve overall supply chain efficiency.
- Predictive Maintenance in Manufacturing: Explore the application of predictive analytics in predicting equipment failures, maintenance requirements, and production downtime based on metal price forecasts. This proactive approach to maintenance can optimize asset utilization and minimize unplanned downtime.
- Dynamic Pricing Strategies: Develop dynamic pricing strategies that leverage real-time metal price forecasts to adjust pricing strategies dynamically. Implement algorithmic pricing models, dynamic pricing algorithms, and price optimization techniques to

maximize revenue and competitiveness in the market.

- Cross-Industry Applications: Identify cross-industry applications of your forecasting models, such as in the automotive, aerospace, construction, or energy sectors. Tailor your models and analyses to address specific industry challenges and opportunities related to metal price dynamics.
- Collaborative Forecasting Platforms: Build collaborative forecasting platforms that facilitate knowledge sharing, collaboration, and data exchange among industry stakeholders, analysts, and researchers. This platform can foster innovation, collective intelligence, and best practices in metal price forecasting.
- Continuous Model Improvement: Establish a framework for continuous model improvement and refinement based on feedback loops, model performance monitoring, and validation against actual market outcomes. This iterative approach ensures that your forecasting models evolve and remain relevant in dynamic market conditions.

CHAPTER 8

REFERENCE

8.REFERENCE

1. **A Time Series Analysis-Based Stock Price Prediction Framework Using Machine Learning and Deep Learning Models (Shcherbina et al., 2023)** explores various models for stock price prediction, potentially applicable to metals ResearchGate: https://www.researchgate.net/publication/340720727_A_Time_Series_Analysis-Based_Stock_Price_Prediction_Framework_Using_Machine_Learning_and_Deep_Learning_Models.
2. **Stock Price Prediction Using Time Series Models (eSource, DBS)** compares ARIMA, Prophet, and LSTM models for stock price prediction, including metals [esource.dbs.ie].
3. **An Assessment of Time Series Methods in Metal Price Forecasting** analyzes the effectiveness of ARIMA and lagged price models in forecasting metal prices [ResearchGate (researchgate.net)].
4. **Forecasting Metal Prices with Machine Learning (Shcherbina et al., arxiv.org)** explores machine learning techniques for metal price forecasting.
5. **Time Series Forecasting for Stock Prices (Brownlee, Machine Learning Mastery)** provides a tutorial on using ARIMA for stock prices, adaptable to metals [machinelearningmastery.com].
6. **Prophet: Forecasting at Scale (Facebook Research)** introduces Prophet, a time series forecasting tool applicable to metal stock prices [research.facebook.com].
7. **Deep Learning for Time Series Forecasting: A Survey (Xing et al., arxiv.org)** surveys deep learning architectures for time series forecasting, potentially applicable to metals.
8. **Multivariate Time Series Forecasting with PyTorch (Madgwick, PyTorch)** demonstrates using PyTorch for multivariate time series forecasting, which can be used for metal prices with other factors [pytorch.org].
9. **Metal Price Forecasting Using Time Series Analysis and Machine Learning (International Journal of Engineering Research and Technology)** explores a combination of ARIMA and machine learning for metal price forecasting.

CHAPTER 9

APPENDIX

#model.py

```
from bokeh.plotting import figure, show, output_file

from bokeh.models import HoverTool

from bokeh.models import HoverTool, DatetimeTickFormatter

import pandas as pd

import yfinance as yf

import numpy as np


def run_simulation(ticker, num_days):

    data = yf.download(ticker)

    data = data.reset_index()


    # Calculate daily returns

    data['Return'] = data['Close'].pct_change()

    # Calculate mean and standard deviation of daily returns

    mean_return = data['Return'].mean()

    std_return = data['Return'].std()


    # Set the number of simulations and time period

    num_simulations = 1000


    # Generate random daily returns based on mean and standard deviation
    daily_returns = np.random.normal(mean_return, std_return, (num_days, 
num_simulations))

    # Create an empty array to store the simulated prices

    simulated_prices = np.zeros((num_days, num_simulations))

    # Set the initial price as the last known close price

    initial_price = data['Close'].iloc[-1]


    # Perform the Monte Carlo simulation

    for i in range(num_simulations):

        price = initial_price

        for j in range(num_days):

            price += price * daily_returns[j, i]

            simulated_prices[j, i] = price


    # Calculate the mean of the simulated prices for each day

    mean_simulated_prices = np.mean(simulated_prices, axis=1)


    # Get the dates for the next num_days

    last_date = data['Date'].iloc[-1]

    next_dates = pd.date_range(start=last_date + pd.DateOffset(days=1), 
periods=num_days, freq='D')


    # Example scoring system based on deviation from historical mean 
return

    historical_mean_return = data['Return'].mean()
    # Calculate deviation of each simulation from historical mean return

    deviations = np.abs(np.mean(daily_returns, axis=0) -
historical_mean_return)

    # Assign scores inversely proportional to deviations

    scores = 1 / (1 + deviations)

    # Select the simulation with the highest score

    selected_simulation_index = np.argmax(scores)

    selected_simulation = simulated_prices[:, selected_simulation_index]



    # Create an interactive plot using Bokeh

    output_file("simulation_plot.html")  # Save the plot as an HTML file



    p = figure(title=f"Monte Carlo Simulation - Next {num_days} Days",
x_axis_label='Date', y_axis_label='Selected Simulation Prices')

    p.line(next_dates, selected_simulation, legend_label="Selected
Simulation", line_width=2)



    hover = HoverTool(tooltips=[("Date", "@x{%F}"), ("Price", "@y")],
formatters={"@x": "datetime"})

    p.add_tools(hover)

    p.xaxis.formatter = DatetimeTickFormatter(days=["%b %d, %Y"])  #
Example format, adjust as needed

    show(p)  # Display the plot



    return selected_simulation
```

#views.py:

```
import pickle

from django.shortcuts import render

from model import run_simulation

import pandas as pd




def predict(request):

    metals = [

        {"name": "Steel", "ticker": "SLX"},

        {"name": "Aluminium", "ticker": "ALI=F"},

        {"name": "Copper", "ticker": "HG=F"},

        {"name": "Cast Iron", "ticker": "IRON"},

        {"name": "Magnesium", "ticker": "MAG=F"},

        {"name": "Titanium", "ticker": "TIE=F"},

        {"name": "Zinc", "ticker": "ZN=F"},

    ]



    if request.method == 'POST':

        try:

            metal_index = int(request.POST.get('metal'))

            num_days = int(request.POST.get('num_days'))

            selected_metal = metals[metal_index]

            ticker = selected_metal["ticker"]



            # Perform the Monte Carlo simulation

            result = run_simulation(ticker, num_days)



            # Prepare the result data for template rendering

            last_date = pd.Timestamp('today')

            next_dates = [last_date + pd.DateOffset(days=i) for i in
range(1, num_days + 1)]

            result_data = [{'Date': date, 'Selected_Simulation_Price':
price} for date, price in zip(next_dates, result)]



            # Pass the result data to the template

            context = {'result_data': result_data}

            return render(request, 'result.html', context)

        except Exception as e:

            error = str(e)

            return render(request, 'error.html', {'error': error})

    else:

        return render(request, 'index.html', {'metals': metals})
```

Templates:

#index.html:

```
<!DOCTYPE html>

<html>

<head>

    <title>Time Series Prediction</title>

    <style>

        body {

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

        }



        .content {

            text-align: center;

        }



        .content h1 {

            margin-top: 50px;

        }

    </style>

    <style>
```

```
        .green-button {

            background-color: green;

            color: white;

        }

    </style>


</head>

<body>

    <div class="content">

        <h1>Time Series Prediction</h1>


        <!-- Add the image -->

        {% load static %}

        <img src="{% static 'enabl.jpg' %}" alt="Company Logo"
style="position: absolute; top: 0; left: 0;">


        <form method="POST">

            {% csrf_token %}

            <label for="metal">Select a metal:</label><br>

            <select id="metal" name="metal">

                {% for metal in metals %}

                    <option value="{{ forloop.counter0 }}">{{ metal.name
}}</option>

                {% endfor %}
            </select><br><br>


            <label for="num_days">Enter the number of days to
predict:</label><br>

            <input type="number" id="num_days" name="num_days"><br><br>


            <input type="submit" value="Predict" class="green-button">


        </form>

    </div>

</body>

</html>
```

#result.html

```
<!-- result.html -->

<!DOCTYPE html>

<html>

<head>

    <title>Time Series Prediction Result</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 40px;

        }
        h1 {

            color: #333;

            font-size: 24px;

            margin-bottom: 20px;

        }

        .result-item {

            margin-bottom: 10px;

        }

        .result-label {

            font-weight: bold;

        }

        .back-button {

            margin-top: 20px;

        }

    </style>

</head>

<body>

    <h1>Time Series Prediction Result</h1>



    <div class="result-item">

        <span class="result-label">Selected Simulation Prices:</span>

        <ul>

            {% for data in result_data %}
            <li>{{ data.Date|date:"F j, Y, g:i a" }} - {{
data.Selected_Simulation_Price }}</li>

            {% endfor %}

        </ul>

    </div>


    <button class="back-button" onclick="goBack()">Back</button>


    <script>

        function goBack() {

            window.history.back();

        }

    </script>

</body>

</html>
```

#error.html:

```
<!DOCTYPE html>

<html>

<head>

    <title>Error</title>

</head>

<body>
```

```
    <h1>Error</h1>

    <p>An error occurred: {{ error }}</p>

</body>

</html>
```

#urls.py

```
from django.contrib import admin

from django.urls import path

from model_app import views



urlpatterns = [

    path('admin/', admin.site.urls),

    path('', views.predict, name='predict'),

]
```

#settings.py:

```
from pathlib import Path



BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = 'django-insecure-
oivn$ek50ka3bntc)qbqu2tg0n00$@qw7pc1n6215(4hei_9n8'

DEBUG = True
```

```
ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'model_app',

]
```

Comparative models codes:

ARIMA:

```
#importing necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from sklearn.metrics import mean_absolute_error, mean_squared_error

#importing drive
from google.colab import drive
```

```
drive.mount('/content/drive')

#importing csv file
data = pd.read_csv('/content/drive/MyDrive/Steel_data.csv', parse_dates=['Date'],
index_col='Date')
target_variable=data["Close"]

#splitting the dataset
steel_close = data['Close']  # Assuming the column name for Zinc's closing price is 'Close'
train_data = steel_close[:-100]  # Use all but the last 10 data points for training
test_data = steel_close[-100:]  # Use the last 10 data points for testing

#data preprocessing
data.info()
data.head()
data.describe()
data.tail()
data.shape
data.isnull().sum()



#plotting the dataset
!pip install mplfinance
import pandas as pd
import matplotlib.pyplot as plt
import mplfinance as mpf

mpf.plot(data, type='candle', style='yahoo', title='Steel Metal Stock',
         ylabel='Price', ylabel_lower='Volume', show_nontrading=True)

plt.show()
```

```
#plotting “Close price” of steel metal
plt.figure(figsize=(10, 6))
plt.plot(data['Close'])
plt.title('Steel Metal Closing Price')
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()

#adfuller test
test_result=adfuller(data['Close'])

#Ho: It is non stationary
#H1: It is stationary

def adfuller_test(close):
   result=adfuller(close)
   labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations Used']
   for value,label in zip(result,labels):
      print(label+' : '+str(value) )
   if result[1] <= 0.05:
      print("strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has
no unit root and is stationary")
   else:
      print("weak evidence against null hypothesis, time series has a unit root, indicating it is
non-stationary ")

adfuller(data['Close'])

#DIFFERENCING

data['Close First Difference'] = data['Close'] - data['Close'].shift(1)
data['Close'].shift(1)
```

```
## Again test dickey fuller test
adfuller_test(data['Close First Difference'].dropna())

data['Close First Difference'].plot()



#AUTO REGRESSIVE MODEL


from pandas.plotting import autocorrelation_plot
autocorrelation_plot(data['Close'])
plt.show()

fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(data['Close First Difference'].iloc[2:],lags=40,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(data['Close First Difference'].iloc[2:],lags=40,ax=ax2)



# For non-seasonal data
#p=1, d=1, q=0 or 1
#fitting ARIMA model
model1=ARIMA(train_data ,order=(1,1,0))
model1_fit=model1.fit()
model1_fit.summary()

# Forecast future values
forecast = model1_fit.forecast(steps=100)
print(forecast)

# Calculate the mean absolute error (MAE)
mae = mean_absolute_error(test_data, forecast)
```

```
# Calculate the mean squared error (MSE)
mse = mean_squared_error(test_data, forecast)

# Calculate the root mean squared error (RMSE)
rmse = np.sqrt(mse)

# Print the evaluation metrics
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)

#TRYING DIFFERENT APPROACH

#importing necessary libraries
import pandas as pd
import numpy as np
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error

# Splitting data into training and testing sets
train_size = int(len(data) * 0.8)
train, test = data[:train_size], data[train_size:]

# Data preprocessing
data['Close'] = pd.to_numeric(data['Close'] , errors='coerce')
data['Close'] .fillna(data['Close'] .mean(), inplace=True)
data['Close']  = data['Close'] .squeeze()

# Function to fit ARIMA model and calculate MSE on the test set
def evaluate_arima(order):
    try:
        model = sm.tsa.ARIMA(train['Close'], order=order)
        fitted_model = model.fit()
        predictions = fitted_model.forecast(steps=len(test))
        mse = mean_squared_error(test['Close'], predictions)
        return mse
    except Exception as e:
        print(f"Error fitting ARIMA {order}: {e}")
        return np.inf  # Return a large value for MSE if fitting fails

# Iterate through different orders and find the best parameters
best_mse = float('inf')
best_order = None

for p in range(3):  # Replace with your preferred range for the AR order
    for d in range(2):  # Replace with your preferred range for the differencing order
        for q in range(3):  # Replace with your preferred range for the MA order
            order = (p, d, q)
            mse = evaluate_arima(order)
            if mse < best_mse:
                best_mse = mse
                best_order = order

print(f'Best ARIMA Order: {best_order} with MSE: {best_mse}')

#forecasting
forecast = model1_fit.get_forecast(steps=30)

# Get the predicted values and confidence intervals
predicted_values = forecast.predicted_mean
confidence_intervals = forecast.conf_int()

# Plot the predictions and confidence intervals
plt.figure(figsize=(12, 8))
plt.plot(train['Close'], label='Train')
plt.plot(test['Close'], label='Test')
plt.plot(predicted_values.index, predicted_values, label='Predictions', color='green')
plt.fill_between(confidence_intervals.index, confidence_intervals.iloc[:, 0],
confidence_intervals.iloc[:, 1], color='green', alpha=0.2)
plt.title('ARIMA Model - Future Predictions')
plt.legend()
plt.show()

data['forecast']=model1_fit.predict(start=500,end=700,dynamic=True)
data[['Close','forecast']].plot(figsize=(12,8))
```

PROPHET:

```
#installing prophet
!pip install prophet

#importing necessary libraries
import pandas as pd
from prophet import Prophet
import yfinance as yf
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
from prophet.plot import plot_cross_validation_metric

#importing dataset
data = pd.read_csv("/content/drive/MyDrive/Steel_data.csv")

#data preprocessing
data = data.reset_index()
data = data[['Date', 'Close']]
data.columns = ['ds', 'y']

data.shape
```

```
#model fitting
model = Prophet()
model.fit(data)

#forecasting
future = model.make_future_dataframe(periods=70)

#making predictions
forecast = model.predict(future)
fig = model.plot(forecast)

# Define the cross-validation parameters
cv_results = cross_validation(model, initial='30 days', period='1800 days', horizon='30 days')

# Calculate performance metrics
metrics = performance_metrics(cv_results)

# Display the metrics
print(metrics)

# Optionally, plot cross-validation metrics
plot_cross_validation_metric(cv_results, metric='mae')
print(metrics)

#Performance metrics
overall_rmse = metrics['rmse'].median()
print("Overall Root Mean Squared Error (RMSE):", overall_rmse)
overall_mae = metrics['mae'].median()
print("Overall Mean Absolute Error (MAE):", overall_mae)
overall_mdape = metrics['mdape'].mean()
print("Overall Median Absolute Percentage Error (MDAPE):", overall_mdape)
```

LSTM:

```
#importing necessary libraries
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import matplotlib.pyplot as plt

#importing data
df=pd.read_csv("/content/drive/MyDrive/steel_10years.csv", parse_dates=['Date'])
df = df[['Date', 'Close Price']]

# Setting 'Date' as index
df.set_index('Date', inplace=True)

# Normalizing the data
scaler = MinMaxScaler(feature_range=(0, 1))
df_scaled = scaler.fit_transform(df)

# Prepare data for LSTM
def create_dataset(dataset, time_steps=1):
    data_X, data_Y = [], []
    for i in range(len(dataset) - time_steps):
        a = dataset[i:(i + time_steps), 0]
        data_X.append(a)
        data_Y.append(dataset[i + time_steps, 0])
    return np.array(data_X), np.array(data_Y)

#creating a dataset to create input-output pairs
time_steps = 10
X, Y = create_dataset(df_scaled, time_steps)

# Reshape input data to be 3D [samples, time steps, features]
X = np.reshape(X, (X.shape[0], X.shape[1], 1))

# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, shuffle=False)

# Build the LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, Y_train, epochs=50, batch_size=32)

# Make predictions
predictions = model.predict(X_test)
predictions = scaler.inverse_transform(predictions)
Y_test = scaler.inverse_transform([Y_test])
# Evaluate the model
rmse = np.sqrt(mean_squared_error(Y_test[0], predictions))
print('Root Mean Squared Error:', rmse)

absolute_errors = np.abs(predictions - Y_test)
mae = np.mean(absolute_errors)
print('Mean Absolute Error:', mae)
```

```
# Plot the results
plt.figure(figsize=(16, 6))

# Get the index values for the actual data
actual_dates = df.index[-len(predictions):].strftime('%Y-%m-%d')  # Convert datetime to string

# Ensure all arrays have the same length
length = len(actual_dates)
actual_closing_price = df['Close Price'].values[-length:]

# Plotting actual price with predicted price
plt.plot(actual_dates, actual_closing_price, label='Actual Closing Price', color='green')
plt.plot(actual_dates, predictions, label='Predicted Closing Price', color='red')
plt.title('Stock Price Prediction using LSTM')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()
plt.show()

# Create input sequences for future predictions
future_seq_length = 30
future_data = df[-future_seq_length:]
future_input = np.array([future_data])
future_predictions = model.predict(future_input)

# Inverse transform the predictions
future_predictions = scaler.inverse_transform(future_predictions)

# Print the predicted values
print('Future Predictions:', future_predictions)
```