PURBANCHAL UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

KHWOPA ENGINEERING COLLEGE

LIBALI-8, BHAKTAPUR

A FINAL REPORT
ON
PADDY PRODUCTION FORECASTING WITH XGBOOST

Project work submitted in partial fulfillment of requirements for the award of the degree of Bachelor of Engineering in Computer Engineering (Seventh Semester).

SUBMITTED BY :
Rajesh Hamal (770329)
Rabin Thimi (770328)
Saurav Neupane (770340)
Salim Lawot (770336)
Jenish Prajapati (770314)

SUBMITTED TO :
Department of Computer Engineering

UNDER THE SUPERVISION OF
Er. Suresh Ghatuwa

09 March 2025

DEPARTMENT OF COMPUTER ENGINEERING

KHWOPA ENGINEERING COLLEGE

LIBALI-8, BHAKTAPUR

CERTIFICATE OF APPROVAL

This is to certify that the project entitled **”PADDY PRODUCTION FORECASTING WITH XGBOOST”** submitted by **Rajesh Hamal, Rabin Thimi, Saurav Neupane, Salim Lawot, Jenish Prajapati** as partial fulfillment of the requirements for the award of the Degree of **Bachelor in Computer Engineering** of **Purbanchal University** We have examined it and may place it before the examination board for their consideration.

Panel of Examiners:

Name	**Signature**	**Date**
External Examiner		
Er.		
Head of Department		
Er. Bikash Chawal		
Project Supervisor		
Er. Suresh Ghatuwa		

ACKNOWLEDGEMENT

We extend our sincere gratitude to all relevant individuals and oraganizations whose support has enabled us to successfully complete this project. We are grateful for all their valuable suggestions, information sharing and guidance. Their guidance has been instrumental in helping us conceptualize the state of paddy production in Nepal and follow a structured, step-by-step approach to successfully complete this project

A special thanks to **Khwopa Engineering College** and the **Department of Computer Engineering** for providing us this opportunity to conduct the project. It helped us to expand our knowledge of artificial intelligence, an emerging cutting-edge technology. During this project we are able to learn the standard process followed by experts of all over the world.

We express special gratitude to our supervisor teacher **Er. Suresh Ghatuwa** for his continuous support throughout this project. He has helped us to understand new methodologies, reviewed our work, broadly expanded our thinking, and shared industrial working experiences to conduct the project more professionally.

We would especially like to acknowledge the invaluable mentorship and guidance provided by **Mr. Arjun Kumar Gaire**, Professor at **Khwopa Engineering College**. He shared his years of teaching and research experience in statistics with us to perform data analysis and model validation.

Also, we would like to express our deep sense of gratitude to **Er. Shree Ram Khaitu**, for his informative insights and suggestion in our project.

Moreover, we would like to extend our gratitude to **Siddhi Ganesh SACCOS - (Chorcha, Bhaktapur)** for sharing their collected agriculture data. We also express our sincere thanks to **Jagati Agro Concern - (Jagati, Bhaktapur), Bhaktapur Agro - (Adarsha, Bhaktapur)**, and **Jagat Agriculture Center - (Adarsha, Bhaktapur)** for giving their valuable time to explain paddy production info in Bhaktapur.

Finally, a special thanks to our friends and colleagues for sharing their suggestions to our project.

ABSTRACT

This study explores the use of XGBoost (Extreme Gradient Boosting), a powerful machine learning technique, for predicting agricultural yields in Nepal. Despite a declining tendency in the agricultural sector's share in Nepal's GDP, paddy cultivation remains a significant contributor to the national economy. The XGBoost algorithm, known for its accuracy and performance, has been used to model paddy yields based on well-structured time series data. The model combines the gradient boosting technique with Classification and Regression Trees (CART) for better predictive performance. Necessary data preprocessing steps like missing values treatment, decomposition, and outlier detection had a significant impact on data preparation for modeling. The results reveal that XGBoost is capable of predicting paddy yields with good accuracy and hence provides valuable insights for agricultural sustainability. However, forecast accuracy of the model could be enhanced by integrating more data like weather patterns, soil types, and diseases. Future research will focus on expanding the dataset to make the model more accurate and reliable.

Keyword: XGBoost, gradient boosting, CART

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Background

Nepal's agricultural sector, once the cornerstone of the economy, is going through big changes. Now the service sector is increasing while the agriculture sector is decreasing gradually. In 2021/22 the contribution of agriculture to total GDP was 25.8 percent which now went down to 24.7 percent in 2022/2023, with the industrial sector remaining relatively constant (Ministry of Finance [MoF], 2023). This shows the gradual descent of the whole agricultural economy of Nepal. However, agriculture still holds the major part of the economy and shows positive signs. Recently published data from MoF (2023) reveals a 3.9 percent increase in paddy production which is important since rice has been a major food source in Nepal for generations. However, there are some problems like droughts, diseases, lack of fertilizer, and inflation that remain.[1]

Over the past few years, Nepal has seen a positive change in the import of rice. According to the Nepal Rastra Bank(2023) report, in the fiscal year 2022/23, the import of paddy reached Rs. 36,404.3 Million, which is a 23.1 percent decrease compared to the previous year Rs. 47,356 Million. Rice imports constituted a notable 2.3 percent share in imports of 2022/23. So declining in rice imports hints a potential progress toward self-sufficiency in rice production and food security.[1]

Due to these changes in the agricultural sector, many nepali researchers are developing AI models. AI models helps to analyze data more efficiently and to uncover hidden patterns inside data. By using AI, researchers aim to enhance agricultural productivity, address challenges, and support Nepal's progress toward self-sufficiency in food production.

This research aims to develop and test another option for the time series data of agriculture model XGBoost and compare its accuracy with existing models. It is gaining popularity gradually among researchers due to its unique gradient-boosting method.

1.2 Motivation

For any country in the world, paddy cultivation is a cornerstone of the agricultural sector and a critical component of the nation's food security and rural economy. Despite its importance, paddy farmers in Nepal face significant challenges due to the inherent volatility in agricultural commodity prices. Factors such as changing weather patterns, limited access to advanced forecasting tools, and fluctuating market demands increases this volatility, making it difficult for farmers to plan effectively and secure stable incomes. Additionally, the lack of accurate, localized forecasting models often leads to inefficient resource allocation and market ineffi ciencies.

Motivated by these challenges, this project aims to develop a deep learning-based sales forecasting system specifically tailored to paddy in Nepal. By leveraging advanced time series forecasting techniques, the project seeks to provide more accurate predictions of paddy supply and demand, thereby helping farmers make informed decisions, optimize production strategies, and reduce the adverse effects of price volatility. This approach not only aims to stabilize farmer incomes but also contributes to overall agricultural sustainability and food security in the region.

1.3 Statement of Problem

In the context of Nepal, despite the importance of agriculture, the condition of the farmers and the market of the crops produced by these farmers face many challenges. Identifying and predicting the condition of padding production in the future is important.

1.4 Objective

The main objective of this project is to create a forecasting model for the paddy crops using gradient boosting technique in XGBoost.

1.5 Scope and limitation

The major scope of the project is, this project can provide a foundation for further research in agricultural forecasting, potentially leading to the development of similar models for other crops or regions.

And the main limitation of this project is the sufficient data. With the help of the gradient boosting method, we can forecast a target label that is influenced by many factors like (soil nutrition, weather and climate, seed quality and variety, fertilizer used, etc). However, there is no sufficient record of data available related to those factors performing it.

CHAPTER 2
LITERATURE REVIEW

Classification or prediction is the most widely used data mining task. Classification algorithms are supervised methods that uncover the hidden relationship between the target class and the independent variables (Salzberg, 1994). Supervised learning algorithms allow labels to be assigned to the observations so that new data can be classified based on training data(Han et al., 2012; Kumar et al., 2015). Examples of classification tasks are image and pattern recognition, medical diagnosis, loan approval, detecting faults or financial trends(Salzberg, 1994; Wu et al., 2008).[2]

Aditya Pokhrel and Renisha Adhikari from Nepal Rastra Bank (2023) compared a comparative analysis of the ARIMA and ARIMAX model in terms of paddy production forecasting in Nepal. The ARIMAX model with agricultural land availability (AGLAND) as an external variable provided a more accurate forecast for 2022 at 5681.17 metric tonnes/hectare compared to the forecast by the ARIMA model at 5787.64 metric tonnes/hectare. Additionally, the ARIMAX model had better error metrics with a mean absolute error (MAE) of 0.0247, a mean absolute percentage error (MAPE) of 0.667, and a root mean square error (RMSE) of 0.0301 compared to an MAE of 0.0295, a MAPE of 0.797, and an RMSE of 0.0373 by the ARIMA model, thereby indicating improved forecast accuracy. These results emphasize the importance of incorporating external factors like AGLAND that represent real-world constraints like land availability that have a direct impact on paddy production.

Chaturbhujj Bhatt, Subarna Shakya, and Tej Bahadur Shahi (2020) conducted a research that compared several machine learning algorithms for predicting paddy yield in Nepal. Their assessment included Support Vector Machine (SVM), Neural Network, Decision Tree, and Naïve Bayes algorithms. The most accurate model came out to be the Decision Tree classifier with an accuracy rate of 80.19 percent, thereby making it the most suitable one for predicting paddy productivity in situations where there is limited availability of data. The performance of the Naïve Bayes algorithm was competitive; on the other hand, the SVM and Neural Network models had difficulties in properly classifying between diverse levels of productivity. The study emphasized on the potential benefits of utilizing decision tree methods with higher classification accuracy in predicting paddy productivity in Nepal.

CHAPTER 3
PROJECT MANAGEMENT

3.1 Team Management

Our 7th-semester project had focused on predicting paddy production using AI models, specifically XGBoost. The project, titled "Paddy Production Prediction with XGBoost," had been developed by our group members:

Rajesh Hamal (770329)
Rabin Thimi (770328)
Saurav Neupane (770340)
Salim Lawot (770336)
Jenish Prajapati (770314)

3.2 Work Breakdown Planning

Our work breakdown planning is as follow,

Job Description	Week							
	1st	2nd	3rd	4th	5th	6th	7th	8th
problem Identification	✓							
Requirement Analysis	✓	✓						
System design	✓	✓						
Coding And development			✓	✓	✓	✓	✓	
Testing And Implementation							✓	
Documentation	✓	✓	✓	✓	✓	✓	✓	✓

3.3 Feasibility Study

A feasibility test is done to check if a project is possible, economical, and time-consuming. We assess the availability of datasets required, required tools, and libraries to see if the project can succeed. We found the following outcomes:

3.3.1 Resources Feasibility:

For this project, we needed a reliable dataset containing labels of area, production, and yield. We went through similar project reports that performed a paddy production prediction with a different model. For the model training and prediction with the AI. We found out some people collect data sets directly with related people by questionnaire method and some people get datasets from the NRB database, but these people were employees of NRB who have access to the NRB database

Dataset: We made the dataset required for us by going through all available reports from MoAD and other related government offices. We enter data manually in an Excel file. Data were found in the different segments (1975-88), (1999-2011), (2014-15), (2018-22). We used different methods to fill the gaps, which are briefly explained in the methodology.

3.3.2 Technical Feasibility:
In technical requirements, we need a computer to train the model, model (XGBoost), and different mathematical and statistical tools. We found the project to be technically feasible. The XGBoost model is an open-source model. Statistical tools are found in free libraries, and for hardware requirements, we used Google Collab for coding and training.

3.3.3 Economic Feasibility:
We found all the required tools, models, and hardware in free service and free-to-use. So the project is found feasible in an economic way.

3.3.4 Time Feasibility:
We use ready-made statistical and mathematical tools for calculation and evaluation, making the project easier and not overwhelming. It fits well within a one-semester timeframe, so the project is manageable and feasible in terms of time.

3.4 Software Requirements

For our project, the required software includes,
1. Google Colab
2. Python
3. Required libraries (eg. xgboost)
4. Excel
5. GitHub
6. Render

3.5 Hardware Requirements

In hardware, we use Google Colab to code and train the model. Google Colab provides all the necessary libraries and system resources like GPU, CPU, storage, and RAM. It offers 12.7 GB of RAM and 107.7 GB of storage, which is more than enough for our project.

3.6 Functional Requirements

The functional requirement for the systems are,

a. The system should generate forecasts for paddy sales volumes and prices based on the trained model.

b. The system must be able to ingest historical data on paddy production and other relevant factors.

c. An interactive dashboard for users to view forecasts, visualize trends, and access historical data.

3.7 Non-Functional Requirements

These are essential for the better performance of the system. The points below focus on the non-functional requirement of the system,

a. Ensure high accuracy in forecasts, with continuous monitoring and refinement of model performance.

b. The system should deliver timely forecasts with minimal latency. Training times should be optimized to handle large datasets efficiently.

CHAPTER 4
SYSTEM DESIGN and ARCHITECTURE

4.1 System Block Diagram

This figure illustrates the working of the XGBoost model, which forms the foundation of our project. Initially, we use the current dataset for training and evaluation. We set labels within the data, then predict future values. As new data becomes available, we can easily feed it into the model, improving accuracy without starting from scratch, enabling parallel updates.

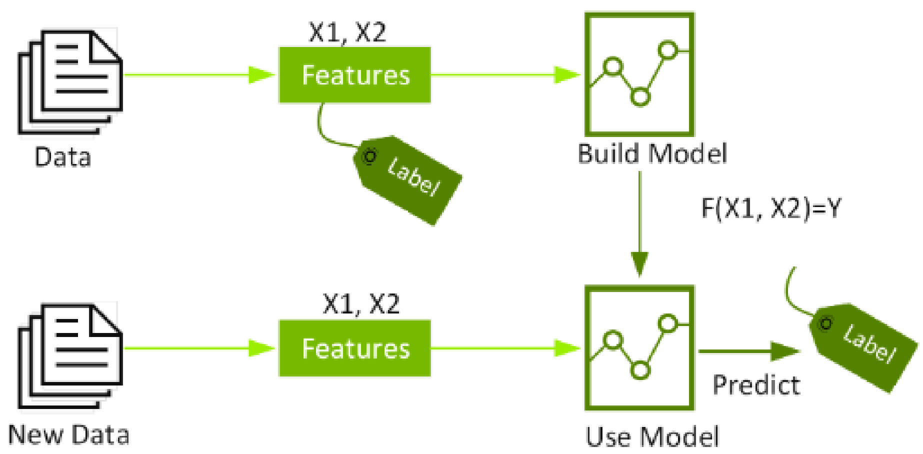

figure 4.1 basic representation of system

4.2 Use Case Diagram

The figure shows the use case of paddy production prediction system where user can select model.

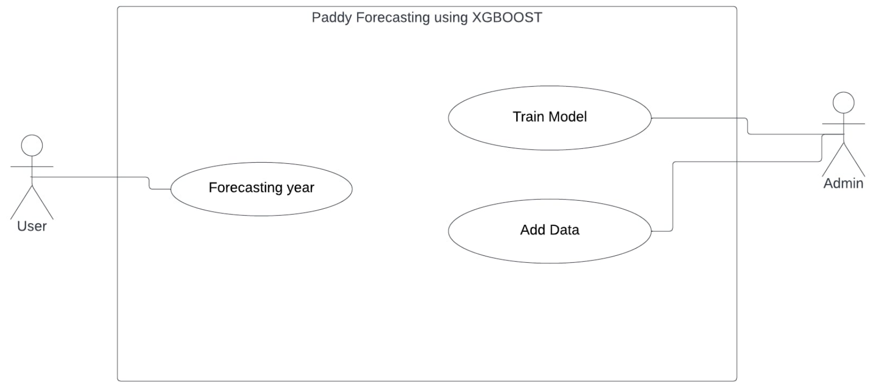

figure 4.2: usecase diagram of system

4.3 Work Flow Block Diagram

The figure shows the work flow of paddy production prediction system. It starts with data preparation, including cleaning, filling, and splitting. The XGBoost model is then trained and evaluated. After ensuring accuracy, it forecast future values. Finally, the trained model is saved for future use, enabling efficient forecasting.

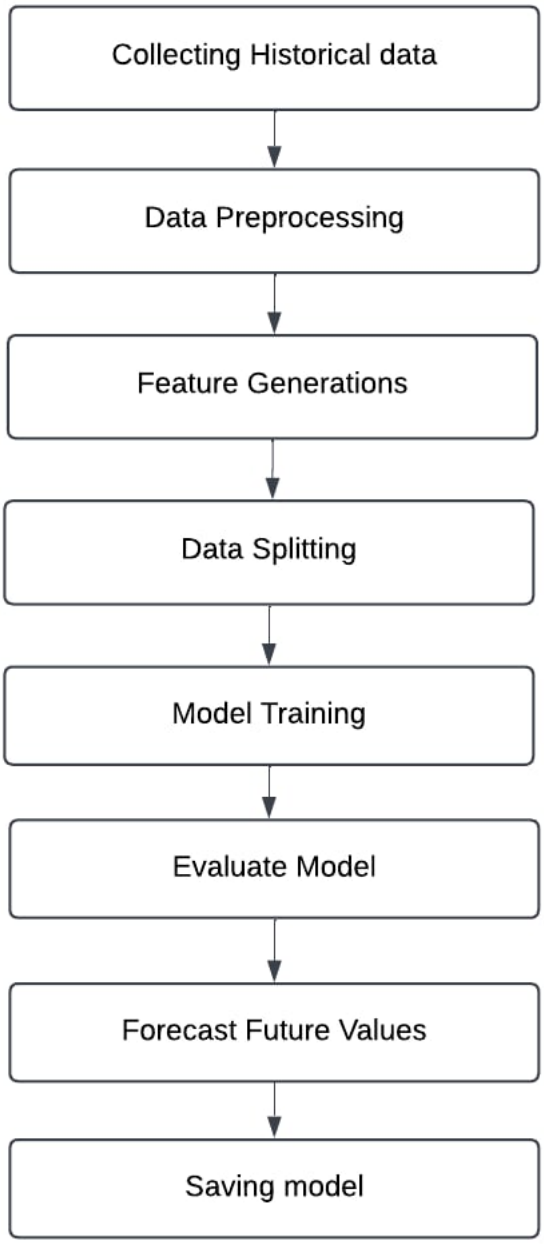

figure 4.3: basic block diagram of system workflow

CHAPTER 5
METHODOLOGY

5.1 Model Specification(XGBoost)

5.1.1 Introduction:

XGBoost stands for "Extreme Gradient Boosting", where the term "Gradient Boosting" originates from the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves problems in a supervised learning environment. [3]

When XGBoost is applied to supervised learning issues, a target variable y_i is predicted using the training data (which contains several features) x_i.. In mathematical form, XGBoost can be expressed as below,[3]

$$\hat{y}_i = \sum_j \theta_j x_{ij}$$

where,
θ = parameter which we need to learn from data
$\hat{y}_i$ = predicted data: weighted sum of multiple tree (CART)

5.1.2 Gradient Boosting:

Boosting is an ensemble technique that combines weak models, usually decision trees, to form a strong predictive model by correcting previous errors. Gradient Boosting builds on this by iteratively adding trees that address the residual errors of prior trees. Each tree is trained to minimize the gradient of the loss function, which measures the difference between predicted and true values. This process continues until the model converges or a set number of trees is reached.

5.1.3 Classification and Regression Trees (CART):

XGBoost uses CART (Classification and Regression Trees) instead of simple decision trees for learning through boosting. Unlike traditional decision trees, where leafs only contain decision values, CART assigns a real score to each leaf. This offers more detailed interpretations beyond just classification and enables a more structured and unified approach to optimization.[3]

We can see here a real score is assigned to the each leafs.

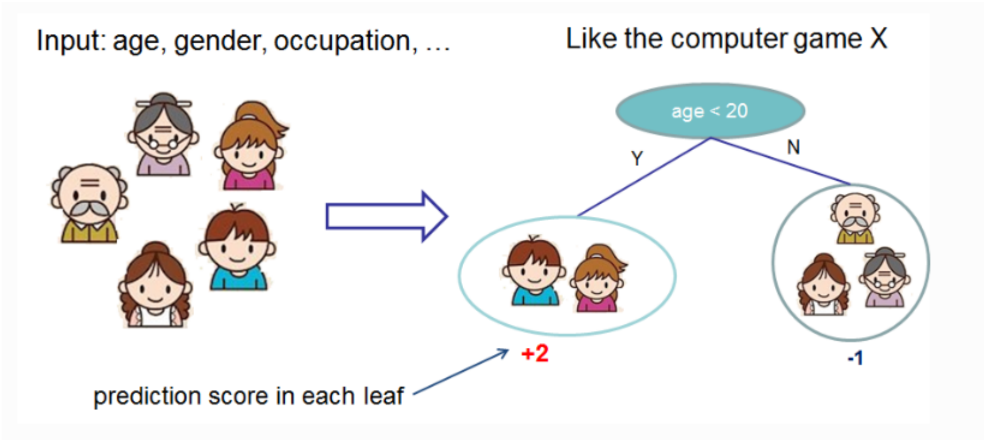

figure 5.1: example of CART (Classification and Regression Tree)

The scores of all tree of same leaf are summed, and this process is repeated across all trees in the ensemble. But in gradient boosting error is minimized in each new tree generation. Which is explain in **Tree Boosting and Scoring**.

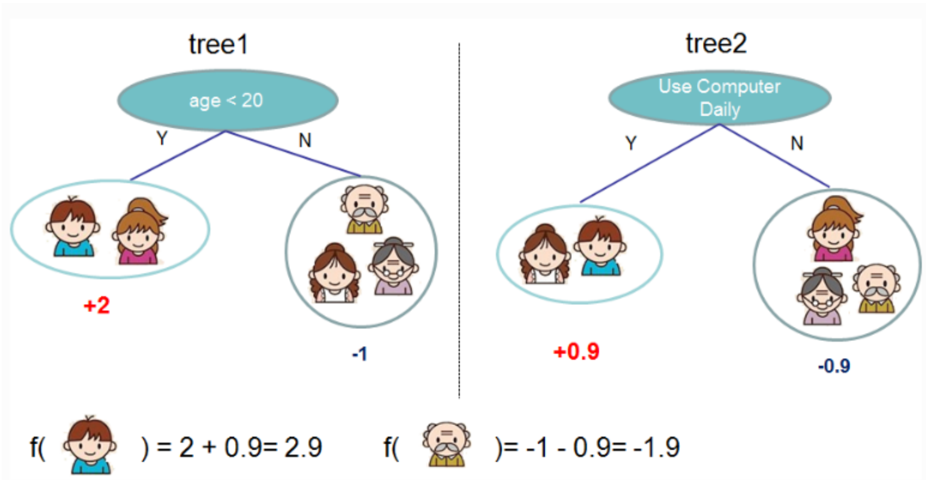

figure 5.1: example of boosting tree

Tree Boosting and Scoring

The parameters of trees include the tree structure and leaf scores. Learning the tree structure is more complex than traditional optimization, where gradients can be easily used. Instead of learning all trees at once, we use an additive strategy, fixing the learned trees and adding one new tree at a time to refine the prediction. While adding a new tree a target value of error is set which must be less than the previous which is gradient boosting. For this following function is used. It has two parts loss function and regularization function and the sum of this function is our objective function.

$$obj(\theta) = L(\theta) + \Omega(\theta)$$

where,
$L(\theta)$ = Loss function(we will use mean squared error) =¿ $\sum_i (y_i - \hat{y}_i)^2$
$\therefore L(\theta) = \sum_i (y_i - \hat{y}_i)^2$ $\Omega(\theta)$ = Regularization function =¿ $\sum_i \omega(f_i)$

We get,

$$obj = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \sum_{i=1}^{t} \omega(f_i)$$

which can be further derived into,

$$obj^t = \sum_{j=1}^{T} [G_j \omega_j + \frac{1}{2}(H_j + \lambda)\omega_j^2] + \gamma T$$

here in this expression there is two main parameter λ and γ. These two parameter controls the regularization of tree and eventually of model itself. While training model we need to provide this parameter values which is described below parameter value.[3]

5.2 Work Flow

We followed a step-by-step process as below from data preparation and model training to evaluation and prediction.

5.2.1 Data Collection:

In this project, we require a dataset with three key labels: production, area, and yield. Even though searching through various government websites, we couldn't find the necessary data. We then explored similar projects sourced their data and discovered that many obtained their data from either internal databases, while others created their datasets through direct surveys and questionnaires
We found one dataset in Kaggle by Ashutosh Chapagain, which he also found from another source we checked in the source but not found. We couldn't verify the dataset. In conclusion, we downloaded all available yearly reports from the MoAD website and manually entered data. But there was another problem when we finished the manual data entry we found data large gap in data from 1988-1999(10 years), 2012-2013, 2016,2017, and 2021.
Due to the large gap in the data, we couldn't use KNN or regression methods. After reading blogs in the community and consulting with teachers, we discovered two effective methods to address the issue.

1. Exponential Smoothing
2. Polynomial curve fitting (Degree 3)

1. Exponential Smoothing:
We used the Holt-Winters Exponential Smoothing method to fill the missing gap in the data, which does not have seasonality. The results are shown below.

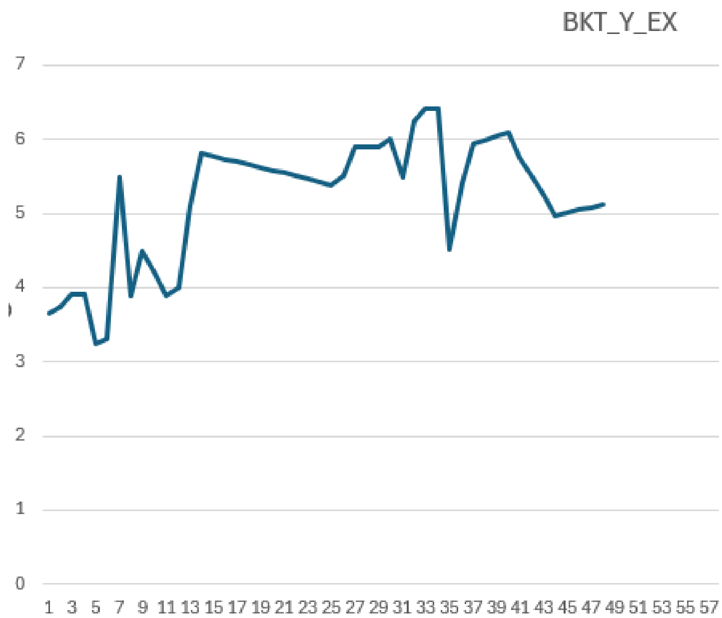

figure 5.2.1: exponential method result

2. Polynomial curve fitting (Degree 3)

We used a 3rd-degree polynomial function which fit nd followed the trend line of data to generate missing data. It show more accuracy than the exponential method with an R^2 value of 71.

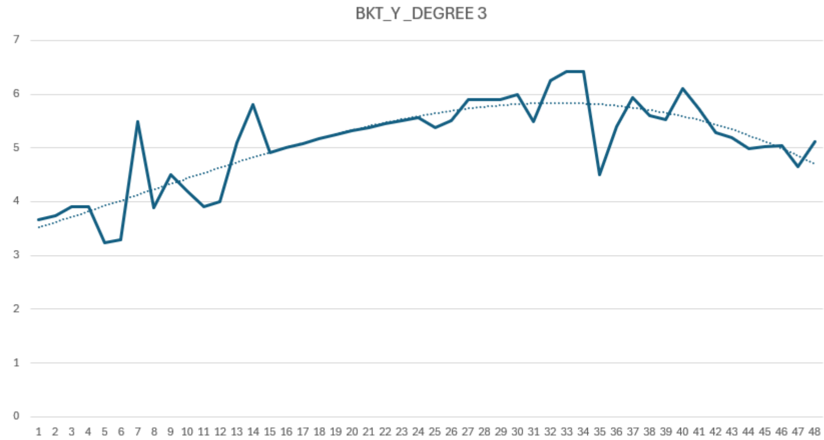

figure 5.2.1: polynomial method result

5.2.2 Data Preprocessing:

In this project, preprocessing raw data was a crucial step in ensuring the effectiveness of time series forecasting for using XGBoost. Since XGBoost is not inherently designed to handle sequential dependencies, it was necessary to transform the raw time series data into a structured format suitable for modeling. We handled missing values, used addictive and multiplicative decomposition to break data into components like seasonality, trend and residual (noise), used Augmented Dickey-Fuller (ADF) test to check if data is stationary or not and outliers detection to structure the time series data for XGBoost. We found the preprocessing steps essential for improving model accuracy. XGBoost requires structured input, and these techniques helped optimize learning. The preprocessing was feasible using Python libraries like Pandas and Scikit-learn, and we performed data preparation in Google Colab for efficiency. These preprocessing techniques helped improve the model's ability to learn patterns effectively, ultimately enhancing forecasting accuracy and robustness.

YEAR	BKT_A	BKT_P	BKT_Y	KTM_A	KTM_P	KTM_Y	LLT_A	LLT_P	LLT_Y
1975	6000	21930	3.655	13000	48800	3.6	6500	22600	3.477
1976	5293	19742	3.73	12101	43563	3.6	5500	20900	3.8
1977	5060	19750	3.903	14170	54830	3.869	4920	16830	3.421
1978	5560	21700	3.903	12090	46780	3.869	5190	17750	3.42
1979	5370	17360	3.233	11720	37420	3.193	4400	15090	3.43
1980	6500	21450	3.3	11720	35280	3.01	4400	14960	3.4
1981	5220	28700	5.498	11060	52530	4.75	4860	18950	3.899
1982	5230	20340	3.889	11550	42390	3.67	3950	15370	3.891
1983	4830	21730	4.499	11000	49500	4.5	5290	21160	4
1984	5220	21920	4.199	11240	44960	4	5560	18350	3.3
1985	4890	19070	3.9	11200	39200	3.5	4700	15280	3.251
1986	4800	19200	4	11100	40000	3.604	4680	16000	3.419
1987	5190	26460	5.098	10460	43410	4.15	4590	18220	3.969
1988	5070	29410	5.801	10260	49250	4.8	4450	19580	4.4
1989									
1990	4770	31480	6.6	9830	46990	4.78	4200	19740	4.7
1991	4750	24390	5.135	9730	40230	4.135	4070	18510	4.548
1992	4710	23310	4.949	9690	39240	4.05	4000	16000	4
1993	5000	25002	5	11160	44280	3.968	4000	16800	4.2
1994	5000	25200	5.04	11160	51336	4.6	4100	17405	4.245
1995	4700	23030	4.9	11146	45730	4.103	4628	18890	4.082
1996	4700	23050	4.904	11140	52410	4.705	4280	16950	3.96
1997	4700	23050	4.904	11146	53500	4.8	5249	24434	4.655
1998	4700	23500	5	11000	54000	4.909	5250	25200	4.8
1999	4700	25310	5.385	9000	47529	5.281	4200	19740	4.7
2000	4700	25850	5.5	9000	45015	5.002	4200	19664	4.682
2001	4577	27010	5.901	8100	46170	5.7	5044	27540	5.46
2002	4577	27010	5.901	8100	46170	5.7	4949	25423	5.137
2003	4577	27010	5.901	8100	46170	5.7	4949	25423	5.137
2004	4503	27018	6	8100	46170	5.7	4655	26068	5.6
2005	4480	24610	5.493	8000	41250	5.156	4800	23000	5
2006	4480	28000	6.25	8000	40800	5.1	4840	22504	4.85
2007	4400	28248	6.42	8000	42800	5.35	4650	22645	4.87
2008	4400	28248	6.42	8050	43250	5.373	4650	22645	4.87
2009	4326	19493	4.506	8050	35700	4.435	4680	21060	4.5
2010	4300	23220	5.4	8025	41730	5.2	4650	21390	4.6
2011	4252	25241	5.936	8000	46080	5.76	4600	26876	5.799
2012									
2013									
2014	4348	26523	6.1	7930	45245	5.706	4680	24804	5.3
2015	4,250	24,400	5.741	7,905	40,200	5.085	4,650	21,166	4.552
2016									
2017									
2018	3,979	19,815	4.98	6,015	26,346	4.38	4,211	18,613	4.42
2019	3,925	19,705	5.02	5,959	26,518	4.45	4,175	18,579	4.45
2020	3,979	20,094	5.05	6,015	26,887	4.47	4,211	18,107	4.3
2021									
2022	3,891	19,922	5.12	5,556	25,058	4.51	4,012	17,894	4.46

figure 5.1: Data with missing values

YEAR	BKT_A	BKT_P	BKT_Y	KTM_A	KTM_P	KTM_Y	LLT_A	LLT_P	LLT_Y
1975	6000	21930	3.655	13000	48800	3.6	6500	22600	3.477
1976	5293	19742	3.73	12101	43563	3.6	5500	20900	3.8
1977	5060	19750	3.903	14170	54830	3.869	4920	16830	3.421
1978	5560	21700	3.903	12090	46780	3.869	5190	17750	3.42
1979	5370	17360	3.233	11720	37420	3.193	4400	15090	3.43
1980	6500	21450	3.3	11720	35280	3.01	4400	14960	3.4
1981	5220	28700	5.498	11060	52530	4.75	4860	18950	3.899
1982	5230	20340	3.889	11550	42390	3.67	3950	15370	3.891
1983	4830	21730	4.499	11000	49500	4.5	5290	21160	4
1984	5220	21920	4.199	11240	44960	4	5560	18350	3.3
1985	4890	19070	3.9	11200	39200	3.5	4700	15280	3.251
1986	4800	19200	4	11100	40000	3.604	4680	16000	3.419
1987	5190	26460	5.098	10460	43410	4.15	4590	18220	3.969
1988	5070	29410	5.801	10260	49250	4.8	4450	19580	4.4
1989	4920	30445	6.20049999	10045	48120	4.79000000	4325	19660	4.550000000000001
1990	4770	31480	6.6	9830	46990	4.78	4200	19740	4.7
1991	4750	24390	5.135	9730	40230	4.135	4070	18510	4.548
1992	4710	23310	4.949	9690	39240	4.05	4000	16000	4
1993	5000	25002	5	11160	44280	3.968	4000	16800	4.2
1994	5000	25200	5.04	11160	51336	4.6	4100	17405	4.245
1995	4700	23030	4.9	11146	45730	4.103	4628	18890	4.082
1996	4700	23050	4.904	11140	52410	4.705	4280	16950	3.96
1997	4700	23050	4.904	11146	53500	4.8	5249	24434	4.655
1998	4700	23500	5	11000	54000	4.909	5250	25200	4.8
1999	4700	25310	5.385	9000	47529	5.281	4200	19740	4.7
2000	4700	25850	5.5	9000	45015	5.002	4200	19664	4.682

figure 5.1: Data with filled missing values

We used rolling window mean with size 3 to fill the missing values above.

```
ADF Statistic: -3.285880233126018
P-Value: 0.01552154742905565
Critical Value (1%): -3.5778480370438146
Critical Value (5%): -2.925338105429433
Critical Value (10%): -2.6007735310095064
```

figure 5.1: Addictive Decomposition

As showen in above image our data only have long term patterns i.e. trend and do not have any seasonality and residual.

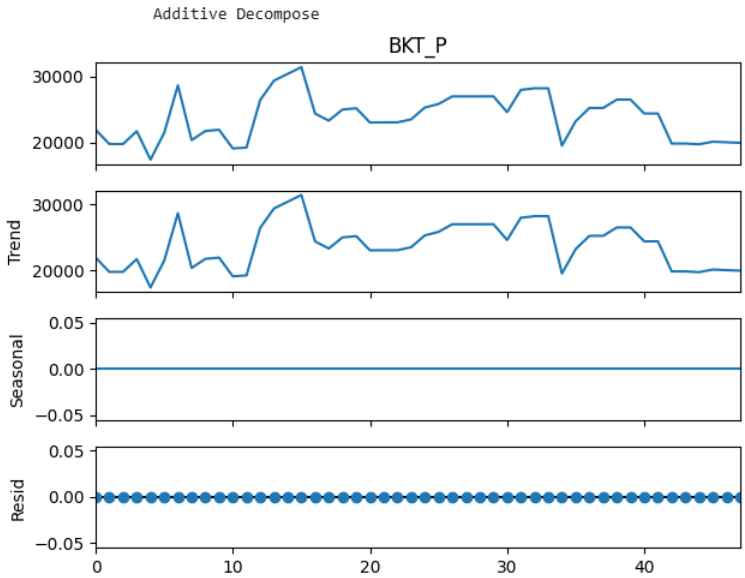

figure 5.1: ADFUller test for stationary

From the image we can see that the p-value is less than 0.05, and the ADF statistic is below the 5% critical value, we can conclude that the dataset is stationary with 95% confidence.

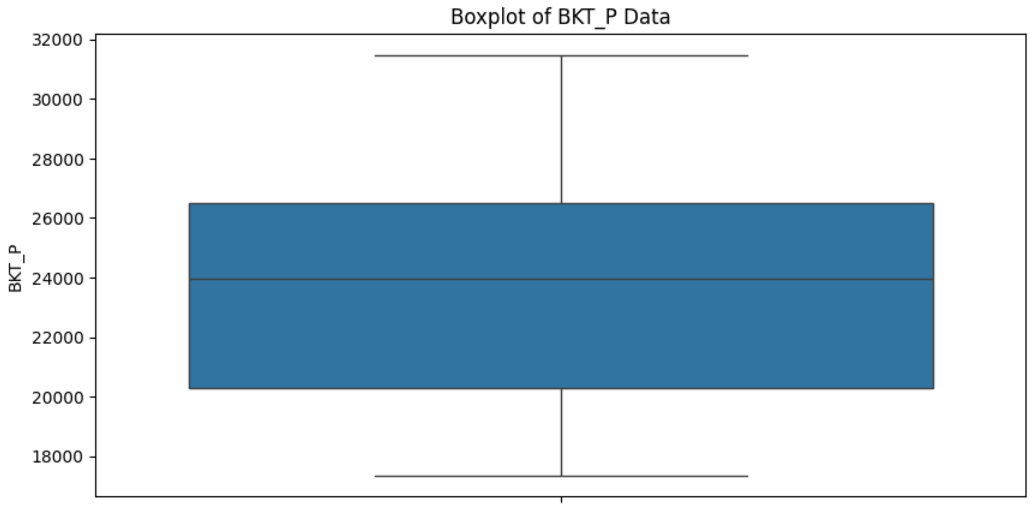

figure 5.1: Boxplot method for outliers

Above boxplot method shows that the there are no extreme ouliers in our data

5.2.3 Features Generations:

In feature generation, we created lag features and rolling window statistics to capture temporal patterns for XGBoost. Lag features helped the model learn dependencies, while rolling windows provided trend and seasonality insights. We found these techniques essential for improving forecasting accuracy. XGBoost requires structured inputs, and feature engineering optimized model learning. The process was feasible using Python libraries like Pandas and NumPy, and we also performed feature generation in Google Colab for efficiency.

	YEAR	BKT_A	BKT_P	BKT_Y	KTM_A	KTM_P	KTM_Y	LLT_A	LLT_P	LLT_Y	sales_log	lag_1	rolling_mean_1	expanding_mean	lag_2
0	1978.0	5560.0	21700.0	3.903	12090.0	46780.0	3.869	5190.0	17750.0	3.420	9.985114	19750.0	19750.0	20780.500000	19742.0
1	1979.0	5370.0	23945.0	3.233	11720.0	37420.0	3.193	4400.0	15090.0	3.430	9.761982	21700.0	21700.0	21413.400000	19750.0
2	1980.0	6500.0	21450.0	3.300	11720.0	35280.0	3.010	4400.0	14960.0	3.400	9.973527	23945.0	23945.0	21419.500000	21700.0
3	1981.0	5220.0	23945.0	5.498	11060.0	52530.0	4.750	4860.0	18950.0	3.899	10.264687	21450.0	21450.0	21780.285714	23945.0
4	1982.0	5230.0	20340.0	3.889	11550.0	42390.0	3.670	3950.0	15370.0	3.891	9.920394	23945.0	23945.0	21600.250000	21450.0
5	1983.0	4830.0	21730.0	4.499	11000.0	49500.0	4.500	5290.0	21160.0	4.000	9.986495	20340.0	20340.0	21614.666667	23945.0
6	1984.0	5220.0	21920.0	4.199	11240.0	44960.0	4.000	5560.0	18350.0	3.300	9.995200	21730.0	21730.0	21645.200000	20340.0
7	1985.0	4890.0	19070.0	3.900	11200.0	39200.0	3.500	4700.0	15280.0	3.251	9.855924	21920.0	21920.0	21411.090909	21730.0
8	1986.0	4800.0	19200.0	4.000	11100.0	40000.0	3.604	4680.0	16000.0	3.419	9.862718	19070.0	19070.0	21226.833333	21920.0
9	1987.0	5190.0	26460.0	5.098	10460.0	43410.0	4.150	4590.0	18220.0	3.969	10.183427	19200.0	19200.0	21629.384615	19070.0

figure 5.1: Data with generated features

5.2.4 Data Splitting:

In data splitting, we separated target and test features to train XGBoost effectively. The first 33 rows was divided into features X_train and X_test with test features and target features respectively, while the remaining rows were assigned to y_train and y_test with test and target features respectively. This ensured proper model training and evaluation. We found data splitting essential for avoiding data leakage and ensuring unbiased performance assessment. The process was feasible using Python libraries like Pandas and Scikit-learn, and we performed data partitioning in Google Colab for efficiency.

	sales_log	YEAR	BKT_A	BKT_Y	KTM_A	KTM_P	KTM_Y	LLT_A	LLT_P	LLT_Y
0	9.985114	1978.0	5560.0	3.9030	12090.0	46780.0	3.869	5190.0	17750.0	3.420
1	9.761982	1979.0	5370.0	3.2330	11720.0	37420.0	3.193	4400.0	15090.0	3.430
2	9.973527	1980.0	6500.0	3.3000	11720.0	35280.0	3.010	4400.0	14960.0	3.400
3	10.264687	1981.0	5220.0	5.4980	11060.0	52530.0	4.750	4860.0	18950.0	3.899
4	9.920394	1982.0	5230.0	3.8890	11550.0	42390.0	3.670	3950.0	15370.0	3.891
5	9.986495	1983.0	4830.0	4.4990	11000.0	49500.0	4.500	5290.0	21160.0	4.000
6	9.995200	1984.0	5220.0	4.1990	11240.0	44960.0	4.000	5560.0	18350.0	3.300
7	9.855924	1985.0	4890.0	3.9000	11200.0	39200.0	3.500	4700.0	15280.0	3.251
8	9.862718	1986.0	4800.0	4.0000	11100.0	40000.0	3.604	4680.0	16000.0	3.419
9	10.183427	1987.0	5190.0	5.0980	10460.0	43410.0	4.150	4590.0	18220.0	3.969
10	10.289124	1988.0	5070.0	5.8010	10260.0	49250.0	4.800	4450.0	19580.0	4.400
11	10.323710	1989.0	4920.0	6.2005	10045.0	48120.0	4.790	4325.0	19660.0	4.550
12	10.357139	1990.0	4770.0	6.6000	9830.0	46990.0	4.780	4200.0	19740.0	4.700
13	10.101969	1991.0	4750.0	5.1350	9730.0	40230.0	4.135	4070.0	18510.0	4.548
14	10.056681	1992.0	4710.0	4.9490	9690.0	39240.0	4.050	4000.0	16000.0	4.000
15	10.126751	1993.0	5000.0	5.0000	11160.0	44280.0	3.968	4000.0	16800.0	4.200

	lag_1	lag_2	lag_3	rolling_mean_1	rolling_mean_2	rolling_mean_3	expanding_mean	rolling_std_2	rolling_std_3
0	19750.0	19742.0	21930.0	19750.0	19746.0	20474.000000	20780.500000	5.656854	1260.939332
1	21700.0	19750.0	19742.0	21700.0	20725.0	20397.333333	21413.400000	1378.858223	1128.149517
2	23945.0	21700.0	19750.0	23945.0	22822.5	21798.333333	21419.500000	1587.454724	2099.228033
3	21450.0	23945.0	21700.0	21450.0	22697.5	22365.000000	21780.285714	1764.231419	1374.017831
4	23945.0	21450.0	23945.0	23945.0	22697.5	23113.333333	21600.250000	1764.231419	1440.488922
5	20340.0	23945.0	21450.0	20340.0	22142.5	21911.666667	21614.666667	2549.119946	1846.309382
6	21730.0	20340.0	23945.0	21730.0	21035.0	22005.000000	21645.200000	982.878426	1818.165284
7	21920.0	21730.0	20340.0	21920.0	21825.0	21330.000000	21411.090909	134.350288	862.612312
8	19070.0	21920.0	21730.0	19070.0	20495.0	20906.666667	21226.833333	2015.254326	1593.434446
9	19200.0	19070.0	21920.0	19200.0	19135.0	20063.333333	21629.384615	91.923882	1609.233772
10	26460.0	19200.0	19070.0	26460.0	22830.0	21576.666667	21794.785714	5133.595231	4229.590209
11	23945.0	26460.0	19200.0	23945.0	25202.5	23201.666667	21938.133333	1778.373555	3686.639165
12	23945.0	23945.0	26460.0	23945.0	23945.0	24783.333333	22063.562500	0.000000	1452.035927
13	23945.0	23945.0	23945.0	23945.0	23945.0	23945.000000	22200.411765	0.000000	0.000000
14	24390.0	23945.0	23945.0	24390.0	24167.5	24093.333333	22262.055556	314.662518	256.920870
15	23310.0	24390.0	23945.0	23310.0	23850.0	23881.666667	22406.263158	763.675324	542.778346

figure 5.1: Splitted test features

	BKT_P
0	21700.0
1	23945.0
2	21450.0
3	23945.0
4	20340.0
5	21730.0
6	21920.0
7	19070.0
8	19200.0
9	26460.0
10	23945.0
11	23945.0

figure 5.1: Splitted target feature

5.2.5 Model Training:

In model training, we used time-based cross-validation and RandomizedSearchCV to optimize the XGBoost DART model. We split the data using TimeSeriesSplit$n_splits = 5$ to maintain the temporal order. The XGBRegressor was initialized with the DART booster, which helps handle overfitting by dropping trees during training. We defined a parameter grid, including learning_rate, max_depth, min_child_weight, gamma, lambda, rate_drop, skip_drop, n_estimators, subsample, and colsample_bytree, to explore various model configurations.

RandomizedSearchCV performed hyperparameter tuning with 50 iterations, using negative mean squared error for evaluation. After randomized tuning of hyper-parameters the best combination of the hyperparameter was then selected using random_search.best_params_ to further train the model with single combination of hyperparameters. We applied feature scaling with scaler_transformX_train, X_test before fitting the model on X_train and y_train, with evaluation on X_test and y_test. The process was feasible using Scikitlearn and XGBoost.

5.2.6 Prediction:

After training, we used the best hyperparameters from RandomizedSearchCV to train an optimized XGBoost model. The XGBRegressor was initialized with random_search.best_params_, ensuring the bestperforming configuration. Before training, we applied scaler_transformX_train, X_test for feature scaling.

The model was trained on X_train and y_train, and predictions were made on X_test. The predicted values were stored in the test dataset under the column prediction. For evaluation, we calculated the mean squared error (MSE) and root mean squared error (RMSE) to assess model accuracy. The process was feasible using Scikitlearn and XGBoost, and execution was performed in Google Colab for efficiency.

	prediction
33	25165.306641
34	25313.226562
35	26993.482422
36	26929.902344
37	24745.134766
38	24604.546875
39	20169.898438
40	20047.970703
41	20070.554688
42	20446.525391
43	20423.119141
44	20426.306641

dtype: float32

figure 5.1: Predicted Data

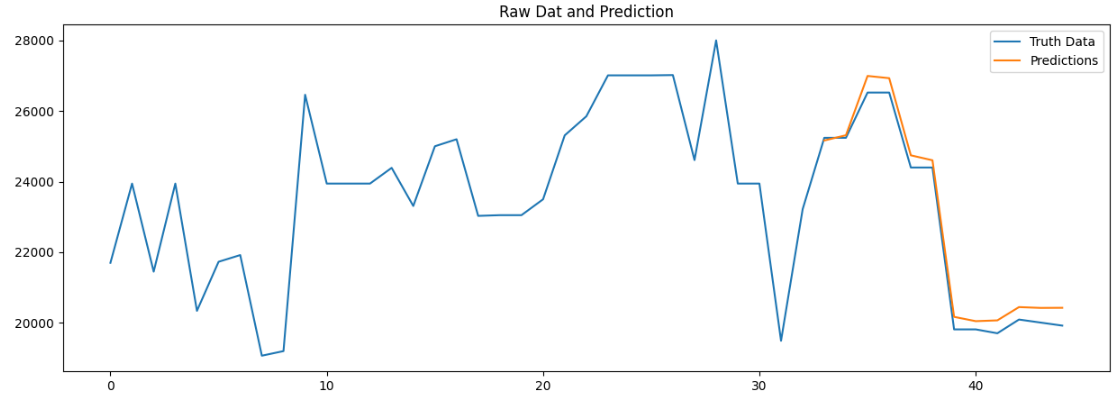

figure 5.1: Plotting original and predicted data

CHAPTER 6
RESULT AND OUTCOMES

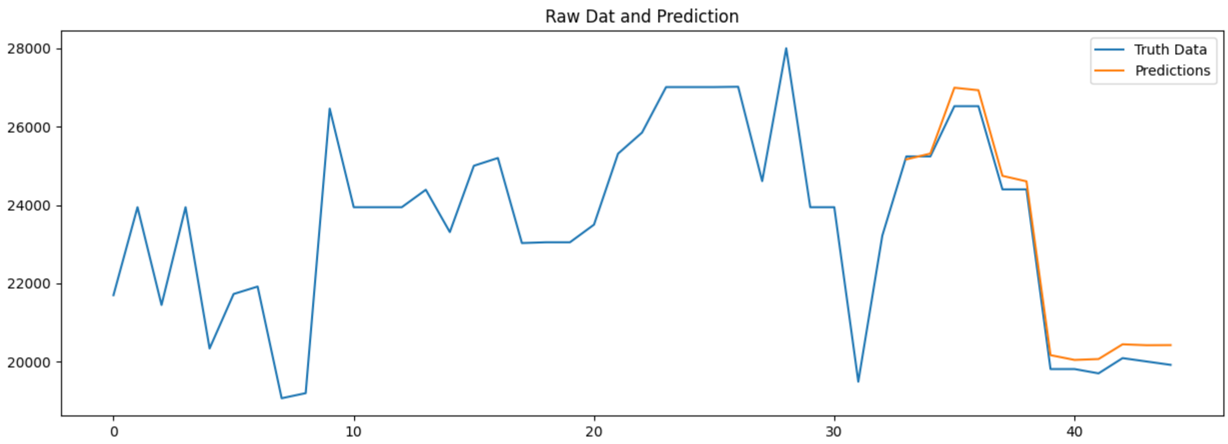

figure 5.1: Plotting original and predicted data

Here, we have plotted prediction of the test values along with the original data

sales_log	lag_1	rolling_mean_1	lag_2	rolling_mean_2	rolling_std_2	lag_3	rolling_mean_3	rolling_std_3	prediction
10.248813	23945.0	23945.0	28000.0	25972.5	2867.317998	24610.0	25518.333333	2174.754775	NaN
9.877862	23945.0	23945.0	23945.0	23945.0	0.000000	28000.0	25296.666667	2341.155342	NaN
10.052812	19493.0	19493.0	23945.0	21719.0	3148.039390	23945.0	22461.000000	2570.363398	NaN
10.136265	23220.0	23220.0	19493.0	21356.5	2635.386973	23945.0	22219.333333	2388.739486	25183.636719
10.136265	25241.0	25241.0	23220.0	24230.5	1429.062805	19493.0	22651.333333	2915.889630	25298.550781
10.185805	25241.0	25241.0	25241.0	25241.0	0.000000	23220.0	24567.333333	1166.824894	26903.134766
10.185805	26523.0	26523.0	25241.0	25882.0	906.510893	25241.0	25668.333333	740.163045	26891.585938
10.102379	26523.0	26523.0	26523.0	26523.0	0.000000	25241.0	26095.666667	740.163045	24607.158203
10.102379	24400.0	24400.0	26523.0	25461.5	1501.187696	26523.0	25815.333333	1225.714621	24439.031250
9.894245	24400.0	24400.0	24400.0	24400.0	0.000000	26523.0	25107.666667	1225.714621	19967.025391
9.894245	19815.0	19815.0	24400.0	22107.5	3242.084592	24400.0	22871.666667	2647.150984	19915.607422
9.888678	19815.0	19815.0	19815.0	19815.0	0.000000	24400.0	21343.333333	2647.150984	19920.425781
9.908226	19705.0	19705.0	19815.0	19760.0	77.781746	19815.0	19778.333333	63.508530	20243.265625
9.903937	20094.0	20094.0	19705.0	19899.5	275.064538	19815.0	19871.333333	200.525144	20206.359375
9.899630	20008.0	20008.0	20094.0	20051.0	60.811183	19705.0	19935.666667	204.338771	20205.541016

figure 5.1: Dataset with predicted values

Above dataset shows the last 15 rows of data along with original data and all the train features and predicted data. The null values in the prediction column denotes that the those rows were part of train set not the test set.

CHAPTER 7
CONCLUSION

rajesh write conclusion text based on outcome here

7.2 Future Enhancement

The XGBoost model showed a reasonable degree of accuracy in predicting paddy crop yields from a limited data sample that did not include many critical contextual factors such as climatic conditions, soil health, natural disasters, and plant diseases. These factors are major determinants of agricultural yields; their absence in the model might have resulted in inaccuracy in overall predictions. Future development would focus on adding more data related to meteorological conditions like temperature, rainfall, and humidity to make predictions more holistic with regard to environmental determinants of agricultural yields. Adding information related to soil health indicators like soil pH levels the availability of nutrients and organic matter would enable the model to consider soil health as a major driver of plant growth. Adding data related to pest and disease outbreaks and records of natural disasters would make predictions more accurate. Increasing the data used and making suitable adjustments within the model would bring substantial opportunities to make agricultural forecasts more precise and reliable and hence support more sustainable and more resilient farming practices.

REFERENCES

[1] Aditya Pokhrel and Renisha Adhikari ”Leveraging Exogenous Insights: A Comparative Forecast of Paddy Production in Nepal Using ARIMA and ARIMAX Models,” ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/publication/382448220$_L$
$08 - Mar - 2025$].

[2] C. Bhatt, S. Shakya, and T. B. Sha, ”Machine Learning Methods for the Prediction of Paddy Productivity in Nepal,” NU. International Journal of Science, vol. 17, no. 2, pp. 59–68, 2020.

[3] T. Chen and C. Guestrin, ”XGBoost: A Scalable Tree Boosting System,” XGBoost Documentation. [Online]. Available: https://xgboost.readthedocs.io/en/stable/. [Accessed: Mar. 8, 2025].