MAJOR PROJECT REPORT ON

"A Multi-Modal Deep Learning Framework for Crop Price Prediction on Crop Price Prediction Dataset"

Submitted

In the partial fulfilment of the requirements for

the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

K. SADA	**21U51A0560**
G. PRADEEP KUMAR	**21U51A0534**
D.PRANAY SAI	**22U55A0512**
SAYAD OMER FAROOQ	**22U55A0535**

Under the guidance of

Prasanthi Madam

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

DRK COLLEGE OF ENGINEERING AND TECHNOLOGY

Affiliated to JNTUH HYDERABAD,

Bowrampet, HYDERABAD-500043.

CERTIFICATE

This is to certify that the Project Report entitled "**A Multi-Modal Deep Learning Framework for Crop Price Prediction on Crop Price Prediction Dataset"** is being submitted by **K. SADA (21U51A0560) , G. PRADEEP KUMAR (21U51A0534) , D. PRANAY SAI (22U55A0512), SAYAD OMER FAROOQ (22U55A0535)** in partial fulfilment for the award of B.Tech degree in Computer Science and Engineering to the **DRK COLLEGE OF ENGINEERING AND TECHNOLOGY Affiliated to JNTUH HYDERABAD**, is a record of Bonafide work carried out by them under the supervision of faculty member of CSE Department.

Guide External Examiner HOD

DECLARATION

I hereby declare that the project entitled "**A Multi-Modal Deep Learning Framework for Crop Price Prediction on Crop Price Prediction Dataset" submitted** for the **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING.** This dissertation is our original work and the project has not formed the basis for the award of any degree, associate-ship, and fellowship, or any other similar titles, and no part of it has been published or sent for publication at the time of submission.

BY

K. Sada (21U51A0560)

G. Pradeep Kumar (21U51A0534)

D. Pranay Sai (22U55A0512)

Sayad Omer Farooq (22U55A0535)

ACKNOWLEDGEMENT

The Project is a golden opportunity for learning and self-development. Consider myself very lucky and honoured to have so many wonderful people lead me through the completion of this project.

We feel it our responsibility to thank **Prof. (Dr.) Gnaneswara Rao Nitta** for whose valuable guidance that the project came out successfully after each stage, and also it is our responsibility to extend our thanks to **Dr. K. Kanaka Vardhini,** Department Project Coordinator**,** for extending her support.

It is a great pleasure for us to express my sincere thanks to **Dr. K. Kanaka Vardhini, HOD CSE,** for providing me an opportunity to do my Internship Program.

It is our pleasure to extend our sincere thanks to our Principal **Prof. (Dr) Gnaneswara Rao Nitta,** for providing me an opportunity to do my academics.

We Also thanks to management members **Sri M. Chandra Sekhara Rao** Director, **Sri D Sriram** Treasurer, and Hon'ble Chairman **Sri D B Chandra Sekhar Rao**

We extend our wholehearted gratitude to all our faculty members of the Department of Computer Science and Engineering who helped us in our academics throughout the course.

Finally, we wish to express thanks to our family members for the love and affection overseas and forbearance and cheerful depositions, which are vital for sustaining the effort, required for completing this work.

With regards,

K. SADA
G. PRADEEP KUMAR
D. PRANAY SAI
SAYAD OMER FAROOQ

Dedicated to my beloved

PARENTS

TABLE OF CONTENTS

ABSTRACT

Accurate crop price prediction is essential for stabilizing agricultural markets and enabling better decision-making by farmers, policymakers, and supply chain stakeholders.

This paper proposes a hybrid multi-modal framework that integrates a Bidirectional Long Short-Term Memory (BiLSTM) neural network with an XGBoost regressor to predict crop prices.

The model is trained and evaluated on the Crop Price Prediction Dataset from Kaggle.

Key innovations include the combination of sequential and non-sequential learning, outlier mitigation, robust feature engineering, and interpretability through SHAP analysis.

The proposed system demonstrates strong predictive accuracy with substantial improvements over standalone deep learning or tree-based models.

CHAPTER-1

INTRODUCTION

1.1 Introduction

Agricultural pricing is one of the most unpredictable and critical components in rural economies, affecting millions of livelihoods. Crop prices fluctuate due to numerous factors like seasonal shifts, demand-supply imbalances, and geopolitical conditions. These fluctuations lead to significant distress among farmers and make policy formulation extremely challenging. Understanding and predicting such variations is essential for providing economic stability to the farming community. Conventional machine learning approaches for crop price forecasting typically involve predefined steps such as data preprocessing, feature selection, dimensionality reduction, and classification or regression modelling. However, these methods are limited in their ability to capture complex patterns across multiple data modalities. Deep learning, particularly BiLSTM, has demonstrated notable success in handling sequential data, while models like XGBoost perform well on structured, non-sequential data. In our project, we implemented a multi-modal hybrid framework combining BiLSTM and XGBoost, trained on the Kaggle Crop Price Prediction Dataset. The dataset was split with 80% used for training and 20% for testing, with 10% of testing data reserved for validation.

1.2 Motivation of the project

In India, over 58% of the rural population depends on agriculture as their primary livelihood. However, the sector remains highly vulnerable to price volatility, seasonal supply-demand mismatches, and lack of timely information. Farmers often experience distress due to unpredictable market fluctuations, resulting in suboptimal financial outcomes.

1.3 Issues/Challenges

• **Temporal Ignorance in Pricing Trends**: Many crop price prediction models fail to account for seasonal and annual time-based patterns, reducing forecasting accuracy significantly.

• **Insufficient Multivariate Integration**: Existing systems often overlook the complex interplay
between diverse data types such as crop variety, location, and yield volume, limiting the model's predictive depth.

• **Exclusion of External Market Factors**: Critical influencers like rainfall, weather changes, and evolving market dynamics are frequently neglected, despite their strong impact on price volatility.

1.4 Objectives

- Develop a hybrid architecture that combines deep sequential learning (BiLSTM) and tree-based modelling (XGBoost).
- Perform extensive data cleaning, feature engineering, and target scaling to minimize loss.
- Evaluate the model using RMSE, MAE, and R^2 metrics.
- Use SHAP for post-hoc interpretability of feature importance.

1.5 Thesis Organization:

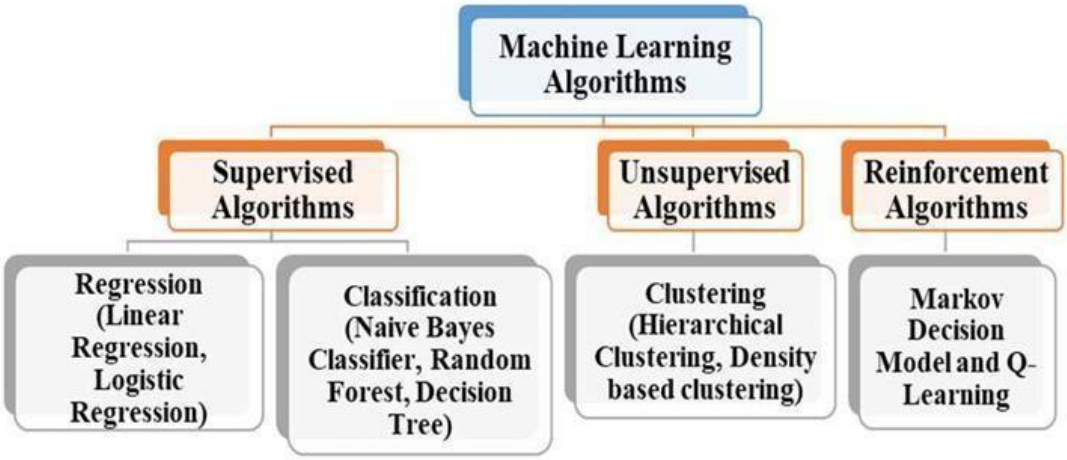
The first part of this chapter is Introduction: Introduction, Motivation of the project, Issues/Challenges, Objectives, and Thesis organization. The second part of this chapter provides a Literature survey. The third part of this chapter provides the SRS which explains Requirement Analysis, Problem Statement, Functional Requirements, Software Requirements Specification, Non-Functional Requirements, Feasibility study and Testing. Chapter 4 explain the various steps which also include Proposed Methodologies & Issues identified. Chapter 5 describes the Experimental Discussions which include Data sets that we used for the model, the Technology used for the model, the Performance matrix/the metrics which we have achieved through the experiment, the Result, and its Analysis. Chapter 6 describes the Conclusions & Future scope.

CHAPTER-2

LITERATURE SURVEY

All economies are based on agriculture. Agriculture has long been regarded as the primary and dominant traditions that exits everywhere. There are several techniques to boost and enhance crop productivity and excellence. Data mining can also be used to forecast crop prices. In essence, data mining is the method of examining data from a variety of aspects and turning it into helpful knowledge. Predicting crop prices is a major agricultural concern. Every farmer tries to determine what proportion of the price he anticipates will really be paid. In the past, price forecasting was done by examining farmers' prior experience with a certain crop. For making decisions on the forecasting of crop prices, accurate knowledge of crop yield history is essential. As a result, this research suggests a strategy for forecasting agricultural prices. The process of examining data sets to gain insight about the facts they contain, sometimes through the use of sophisticated software and technology is known as data analytics. In the past, farmers' prior experience with a crop and land was taken into account when predicting crop prices. Farmers are compelled to grow an increasing number of crops, though, because of how quickly the environment is changing day by day. Due to the existing circumstances, many of them are unaware of both the advantages they receive by cultivating them and the potential losses they may sustain. To find patterns in the data and then process them, the proposed system employs prediction techniques based on machine learning namely XGBoost, Neural Nets, Clustering, Logistic Regression, and Decision Trees. This in turn will aid in determining the crop's target price. A farmer should constantly make the proper crop selection while taking environmental factors into account because crop selection is one of the most essential factors that significantly impacts the final yield. It can be challenging to decide which crop is best for a particular farm because there are so many different factors that might affect the production. Experts are regularly called upon to help farmers with crop recommendations or crop selection, however many farms cannot afford or have access to this alternative due to its time and cost requirements. The farming sector

is confronted with substantial obstacles, including food supply, productivity, and sustainability due to the enormous population growth and sporadic weather variations. Despite their expertise in crop cultivation, farmers in rural areas lack access to a wide range of scientific and technological knowledge. Climate change and its consequences, such as extreme weather, are one of the major obstacles to a nation's ability to ensure its food security. A subtype of artificial intelligence called machine learning enables computers to learn from data without the need to be explicitly processed. Machine learning is now more useful because to big data technology. A considerable amount of information that is collected quickly from multiple sources is simply referred to as "big data." Essentially, machine learning is about developing mathematical approach to better understand data. The three primary kinds of difficulties that machine learning is used to address are supervised problems, unsupervised challenges, and reinforcement concerns. The various categories are depicted in Fig.1. The empirical parameters for machine learning consist of a tuple comprising (x,y), in which denotes the so-called independent traits while y indicates the dependent variable or endpoint. Every time the factor y is known prior to the actual training, investigators experience a supervised learning problem. Regression and classification approaches are the two most common issues in supervised learning. Among the difficulties with regression is estimating land values. When some variable y was not known during training, unsupervised learning has been used. Many clustering problems have been resolved using their methods.



Crop productivity projections typically involves the use of statistical models, which is laborious and time-consuming. Big data's introduction in this decade has

increased the adoption of more sophisticated analysis methods like machine learning. In accordance with the study challenge as well as strategies, the machine-learning model might be generally descriptive or prescriptive. The three main types of difficulties that machine learning is employed to solve are supervised learning, unsupervised learning issues, and reinforcement difficulties. Long-term changes in local, regional, or globe temperatures or weather patterns are referred to as climate disruption. Climate disruption makes it more difficult to reduce emissions of greenhouse gases and prevent global warming. Machine learning with pragmatic agricultural modelling concepts were coupled to produce a machine learning paradigm for major agricultural productivity prediction.

Recurrent neural networks (RNN) as well as convolutional neural networks (CNN) are coupled in an unique multilevel deep learning method to incorporate spatial and temporal data for crop productivity prediction. Paper proposed a preliminary study to demonstrate the effect of merging crop simulation using machine learning on the improvement of maize output forecasts in the US region. Another investigation into the forecasting of potato tuber output were performed employing the four machine learning techniques. It will also be intriguing to think about methods involving Big Graphs and information obtained from smartphone sensors. The 1-2.5 degree Celsius temperature increase predicted for 2030 is anticipated to have significant consequences on crop yields as it permits modifications to photosynthesis, accelerates plant respiration, and impacts pest population growth. "No hunger" is one of the objectives slated for accomplishment by 2030, along with "developing sustainable agriculture". Sustainable agriculture supports gender equality, poverty alleviation, small-scale farmer empowerment, and national economic prosperity. The current situation is worrying. Countries would impose continuous agricultural and food production techniques in order to guarantee sustained access to nutrient-dense food for everyone. To achieve these objectives, agricultural observance must be timely and cost-effective. Agricultural production quantification is essential in this context for assessing and strengthening advanced cognitive capacities like coverage for agriculture, forecasting the financial markets, and addressing worries about food and nutrition security. In view of the rapidly advancing technology, the current study aims to alter the procedure and boost crop yield employing machine learning algorithms and control mechanisms. In the earlier, agricultural yield estimates were made

using machine learning approaches, particularly multivariate and linear regression. The paper suggests improved machine learning techniques rely on specialised ensemble methods like gradient boosting, random forest, selection operator (LASSO) regression, and least absolute shrinkage. Their objective is to develop a web app that informs landowners and consumers about the quantity of crop production that will be produced based on the specific input, as well as the interrelations between the production and the various independent variables. A framework based on convolutional neural network recurrent neural network is addressed. Several models, such as deep fully connected neural networks, random forest (RF). The Forecasting was done for the US Corn Belt region for the years 2016, 2017, and 2018. Three types of variables, including soil, weather, maintenance, and the accuracy for corn and other crops, were used to determine the results 87.82% and 87.09%, respectively, were for soybean. To predict the yields, a classifier based mechanism random forest technique was used. According to this work, a web-based graphical interface was created to allow farmers to estimate crop yields before planting. The dataset includes information on Maharashtra's crop production, where the study was carried out. For yield forecasts, physical elements were applied as opposed to fuzzy models. Reports on the annual forecast evaluation. An experimental investigation for agrarian prediction was conducted in order to focus on making predictions the yield of "bajra," or millet harvest, using appropriate computational models such as regression with time-series models. Auto-regressive integrated moving average models (ARIMA) and an exogenous variable prediction model (ARIMAX) were employed. The ARIMAX model produced the best results. In comparison to the "bajra" regression time series approach. The use of ML to forecast crop yield was suggested. For agricultural yield, they employed layered regression, productivity, with soil nutrients as a secondary factor. The smallest errors for the LASSO, kernel ridge, and efficient neural network (ENeT) methods were 4%, 2%, and 1%, respectively. Natural selection-based genetic algorithms can provide efficient predictive model. A. Machine Learning Techniques The Regression Model portrays an association between independent and one or much more dependent variables. Using data and regression techniques to reduce loss or noise or error (RMSE or MSE), learning can be accomplished in a machine learning framework. Multiple independent variables were found to be more effective and trustworthy

than one independent variable when the MLR analysis was applied in these applications. Sets of adaptable splines with distinct slopes are created from the learning data. When used for classification and regression, the K-NN gives close neighbours greater weight when making predictions. Almost as outcome, they are much more indicative of the overall mean than their distant neighbours. The distance between two neighbours can be calculated using a number of distance formulas, including the Euclidean, Manhattan, and Minkowski ones. To categorise the data points, the binary classifier support vector machine generates a linear hyperplane that separates. Support vector regression, which can be employed to resolve regression problems, varies significantly from SVM. Decision Tree is a supervised machine learning method that can be used for regression as well as classification. The graph is composed of three nodes: a leaf node, a decision node with both inbound and outbound edges, as well as a root node with no incoming edges. A nonparametric study using such data, and produces binary trees using the continuous and discrete features. Information in the CART gain ratio, the Gini Diversity Index (GDI), and also gain splits the attributes using. Random forest is an efficient way for yield estimation that has been used in agricultural research. It generates a wide range of regression trees for evaluating regression from a large amount of decision trees. The Random forest performs more accurately than any other decision tree, and the randomization allows the single decision tree to make up for the bias. Unpruned decision trees are used in the ensemble model known as extremely randomised trees (extra tree), which is similar to random forest. The same amount of trees and variables are used to divide the nodes into groups as in RF. The most popular machine learning method for predicting agricultural yield is an ANN, that could also denote a complicated nonlinear relationship between inputs and outputs. An input layer, a hidden layer, and an output layer make up the total of three layers. An ANN's performance is influenced by many factors, including the quantity of nodes in the learning rate, training tolerance and the hidden layer. The learning statistic illustrates how significantly the rating will change over the span of many iterations in order to achieve the projected value within an acceptable range of the observed data. B. Deep Learning Techniques Deep learning is a subset of a wider range of machine learning techniques that integrate artificial neural networks and representation learning. A typical optimization model usually results in a localised rather than a global optimum

state since the standard ANN is a local minima problem. Furthermore, a complexity problem brought on by overfitting occurs infrequently in generic machine learning models. The nonlinear operations in the activation layer could enhance CNN's capacity to perform nonlinear fitting. CNN in general updates weight using BP similarly to BPNN. Classification is successfully accomplished using an LSTM network. In addition to forecasting the timeseries data and sequence data. It encompasses of input, output, and forgetting gates that used to regulate the preceding status filtering. This Structure seeks to obtain influential prior statuses rather than the most recent ones, to the present. The specific LSTM models' methodology and network structure can be used [51]. C. Miscellaneous Algorithms Other than core categories, numerous classification and regression methods, including as XGBoost, MARS, ridge regression (RIDGE), Gradient boosting TOMGRO and elastic net techniques, have been successfully applied in crop yield prediction. Crop yield prediction has also made use of hybrid techniques including CNN-RNN, CNN-LSTM, and MLR ANN. A simple and precise estimating technique is needed by agricultural management to estimate rice yields throughout the arrangement process. Comparing the performance of ANN models with various formative parameters, and consider the compatibility of several direct relapse schemes with ANN models. In conclusion, Regression Neural Networks (GRNN) technique is used to predict the creation of agricultural produce. They considered GRNN to be a respectable tactic for predicting grain production in provincial districts. The GRNN model was identified as being appropriate for multitarget, multi-variate, and non-straight estimating. In contrast, planting, particularly the application of the ideal amount of treatment, has a substantial effect on crop production of certain other categories. The expert displays his evaluation of the crop forecast's updated k-Means grouping calculation. Their findings and evaluation showed a correlation between modified k-Means with k-Means bunching computation, and they discovered that the tweaked k-Means had produced the most high-calibre groups, the correct yield forecast, and the most extreme precision count. A technique for analysing time on parametric, layout information is to cut out key measurements and other informational characteristics. A method to predict future qualities based on recently viewed qualities is time arrangement anticipating. A novel idea for harvesting productivity under typical meteorological conditions was put forth, and

a predicted model was devised by using time arrangement techniques to historical yield data. Recent years have seen a dominance of machine learning (ML) algorithms that have grown out of the data science paradigm. Forecasting time series in the financial and economic sectors has been done using it. Numerous empirical studies have demonstrated that when forecasting various financial assets, ML techniques outperform time series models, provides a comparison of statistical models and machine learning strategies. These are all nonparametric, data driven strategies that identify stochastic dependencies in the data. According to reports, ANNs perform better than traditional statistical techniques like linear regression and Box-Jenkins procedures. Technologies based on machine learning and deep learning have been shown to be effective solutions for time series prediction issues with strong seasonality. It has been demonstrated that neural networks are more accurate in forecasting agricultural prices than statistical methods. A well-written description of methods for forecasting time series using GRNN and utilising their inherent qualities to produce quick, extremely accurate forecasts may be found. Hog price predictions were made using support vector regression. Crop yield predictions using Random Forests regression can be found in some theoretical time series facts modelling developments.

Review of Existing Methods :

1. Statistical Methods (ARIMA, Exponential Smoothing):

• **Strengths**: Suitable for linear trends and short-term forecasting. Easy to implement and interpret.
• **Limitations**: Assumes stationarity; struggles with complex seasonality, multivariate dependencies, and non-linearity.

2. Machine Learning Models (Linear Regression, Decision Trees, Random Forests)

• **Strengths**: Handle multivariate tabular data and work well with structured inputs. Random Forests can handle noisy data.
• **Limitations**: Poor handling of temporal dependencies. Require extensive feature engineering to represent time-based patterns. Limited interpretability at scale.

3. Deep Learning Models (RNN, LSTM)

- **Strengths**: Can model sequential patterns in data effectively; LSTMs overcome vanishing gradient issues in RNNs.

• **Limitations**: High computational cost; prone to overfitting with small datasets. Output is often treated as a "black box," with limited interpretability.

Strengths and Limitations of Existing Methods :

Approach	Strengths	Limitations
ARIMA / Stats Models	Good for trend/seasonal series	Cannot handle multiple inputs; assumes linearity
Decision Trees / RF	Interpretable, handles categorical data	No built-in time-series modeling
SVM / Linear Models	Fast training, scalable	Inflexible for non-linear time-based trends
LSTM	Good with time-dependent data	Sensitive to data quality, complex to train
CNN (where used)	Fast training on fixed windows	Doesn't handle sequential long dependencies well

Difference Between Existing and Proposed Method :

Aspect	Existing Methods	Proposed Hybrid Method
Temporal Modeling	Ignored or weak (except LSTM)	BiLSTM captures both forward and backward time dependencies
Nonlinear Residual Modeling	Limited (e.g., Random Forest)	XGBoost models residuals missed by BiLSTM
Interpretability	Poor or nonexistent	SHAP provides feature-level insights
Data Type Support	Focused on either time or tabular	Multi-modal handling of time-series + categorical features
Performance	Moderate accuracy	High accuracy with low RMSE and high R^2 (~0.91)
Scalability	Requires manual tuning per crop/region	Modular and extensible for multiple crops/regions

CHAPTER-3

SOFTWARE REQURIMENT SPECIFICATION

3.1 Requirement Analysis

To develop a robust and accessible solution for agricultural stakeholders, this project proposes a crop price prediction system based on a hybrid machine learning model. The system is implemented using Python-based data science frameworks and is designed to run on desktop or cloud environments. It processes historical crop pricing data and other relevant features to produce accurate and interpretable forecasts.

The core objective is to overcome the limitations of traditional models by capturing both timedependent and independent patterns using a combination of BiLSTM (Bidirectional Long Short-Term Memory) and XGBoost regressors. The model is evaluated and optimized for performance, interpretability, and generalizability.

3.1.1 Problem statement

Traditional statistical and rule-based models often struggle with the dynamic and nonlinear nature of crop price fluctuations, which are influenced by various temporal factors (e.g., seasonality) and nontemporal factors (e.g., region, crop type). These models are typically not robust enough to handle diverse datasets across geographic locations and seasons. Hence, there is a need for an integrated machine learning framework that combines sequential learning with feature-rich modelling to predict crop prices with high accuracy and reliability.

3.1.2 Functional Requirements

The core functional requirements of the crop price prediction system include:

- **Data Preprocessing**: Clean and prepare data by removing outliers, handling missing values, and encoding categorical variables.
- **Feature Engineering**: Generate temporal features (lags, rolling averages) and encode static attributes (e.g., state, crop variety).

- **Hybrid Prediction Model**:

  - Train a BiLSTM network to model sequential temporal patterns.

  - Train an XGBoost model on BiLSTM residuals to capture nonlinear residual variance.

- **Prediction Output**: Produce accurate price forecasts for the target crops.

- **Model Evaluation**: Evaluate using metrics such as RMSE, MAE, and R^2.

- **Model Interpretability**: Use SHAP (SHapley Additive exPlanations) to provide feature-level insights into the predictions.

3.1.3 Software Requirements Specification

The Software Requirements Specification (SRS) document is intended to provide the requirements of the Recommender System project. The document includes the project perspective, data model, and constraints of the overall system.

3.1.3.1 Scope

The primary scope of this project is to build a machine learning-based crop price prediction tool that integrates both deep learning and ensemble learning methods for enhanced performance and interpretability. The system is implemented in Python using frameworks such as TensorFlow, XGBoost, and SHAP. It supports data ingestion, preprocessing, training, evaluation, and interpretation of results.

The system is not intended to be a mobile or embedded application but rather a backend computation engine that can be used independently or integrated into broader agricultural analytics platforms in the future.

Potential extensions include:

- Integration of external data sources like rainfall or weather patterns
- Deployment as a web dashboard or API service
- Support for multiple crops and regions in parallel

3.1.3.2 Technologies used

1. Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

2. Google Colaboratory

Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser and is especially well suited to machine learning, data analysis, and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

3.Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

3.1.4 Software Requirements

The software requirements will consist of the essential components required for project development. The developer who works on the product will get all the answers which are required for the project development. The software requirements help predict the project cost and helpful in gathering all the tools for project development.

The software requirements for this project are as follows:

• Operating System: Windows10/11.
• Software packages: Python, TensorFlow, XGBoost, SHAP.
• Tools Required: Google colab,

3.1.5 Hardware Requirements

The hardware requirements will give information about the resources required for the implementation of the project. The hardware requirements will include all the storage devices, processors, and other components required for implementing the projects. These requirements also give an idea to the developer that the specification is required for the project to run without any failures.

The hardware requirements for this project are as follows:

• Operating System: Windows 11 pro
• Processor: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
• Number of CPU Cores: 4
• CPU Base Clock: 1.8Ghz
• RAM: 8GB DDR4
• Graphics Card Memory: 4GB

3.1.6 Functional Requirements

These define the essential capabilities that the software must provide to users and developers:

1. **Data Loading and Cleaning**

o Accept CSV or structured datasets containing historical crop prices and metadata.

o Handle missing values, remove irrelevant columns, and clean numeric data formats.

2. **Feature Engineering**

- Generate lag features (1-day, 7-day, 30-day lags).
- Compute rolling statistics (mean, standard deviation).

o One-hot encode categorical variables like crop, state, and month.

3. **Model Training**

o Train a **BiLSTM** model on temporal features to capture sequential dependencies.

o Use **XGBoost** to model residual errors from the BiLSTM predictions.

o Merge predictions from both models to compute final output.

4. **Model Evaluation**

- Compute performance metrics including RMSE, MAE, and R^2 on a test set.
- Apply a chronological split (e.g., 80/20) to preserve temporal order.

5. **Interpretability**

o Generate SHAP plots to interpret feature importance and contribution to predictions.

6. **Prediction Interface** o Accept user input (e.g., crop name, state, date) and return predicted price. o Output forecasted values in tabular and visual form (e.g., matplotlib or seaborn graphs).

3.1.7 Non-Functional Requirements

These define quality attributes, constraints, and performance expectations of the system:

1. **Accuracy and Performance**

o the hybrid model should achieve an **R^2 score ≥ 0.90** on scaled data.

o RMSE and MAE should be minimized using proper loss functions (e.g., log_cosh for BiLSTM).

2. **Scalability** o Should be capable of handling datasets with millions of rows without performance degradation.

o Modular design enables future integration with additional crop types or external data (e.g., weather APIs).

3. **Portability**

- The system should be runnable across different platforms (Windows, Linux, Google Colab).
- Code should be containerizable using Docker if needed for cloud deployment.

4. **Maintainability**

- Codebase should follow modular design using clear separation for data preprocessing, modeling, and evaluation.
- Easily extendable to support additional models or feature types.

5. **Usability**

- Outputs (predictions, feature importance, error metrics) should be human-readable and visualized via clear plots.
- Documented notebooks or scripts should guide the user through data loading, training, and interpretation.

6. **Security and Privacy**

- System should ensure data confidentiality if integrated with real-time APIs or government datasets in the future.
- Currently limited to local processing with no external data transmission.

3.1.8 External Interface Requirements

• Operating System: Windows 10 and above.

• Processor: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz

• RAM: 4GB DDR4

• Included development tools: Google collab

• Included Python package: NumPy, sci-ki learn, seaborn, TensorFlow, Keras, matplotlib

3.1.9 Feasibility Study

A feasibility study is conducted to assess whether the proposed system — a hybrid deep learningbased crop price prediction model — is viable across various dimensions: technical, economic, and operational.

3.1.9.1 Technical feasibility

The project is **technically feasible**, as it uses widely available technologies such as:

- **Python programming language**
- **TensorFlow/Keras** for BiLSTM implementation
- **XGBoost** for gradient-boosted regression modelling
- **SHAP** for interpretability
- **Jupyter Notebooks / Google Colab** for execution and visualization

The required hardware (a standard CPU or optionally GPU for faster training) and software tools are readily accessible and well-supported. The system runs on local machines or cloud environments like Google Colab, making it highly adaptable without requiring any specialized infrastructure.

3.1.9.2 Economic feasibility

The project is **economically feasible** due to the following reasons:

- No need for expensive proprietary software; all tools used are **open-source**.
- Development and experimentation can be done using **free cloud platforms** (e.g., Google Colab) or **personal computing environments**.
- Costs are primarily associated with:
  - Optional cloud GPU usage for model training (if desired).
  - Storage of large datasets (if scaled to production or real-time prediction scenarios).

The system can be developed and evaluated with minimal financial investment, making it suitable for academic research or early-stage prototyping.

3.1.9.3 Operational feasibility

The project is **operationally feasible** for the following reasons:

• Users (e.g., researchers, data analysts, agricultural policy experts) only require **basic knowledge of data science workflows** to operate the model.

• The model and its outputs (predicted prices, SHAP plots) are presented in an **interpretable and user-friendly format** using plots and summaries.

• Deployment can later be extended via APIs or integrated into dashboards, ensuring flexibility and real-world usability.

Once trained, the model can be used reliably without constant technical intervention, making it viable for semi-automated decision support systems in agriculture.

3.1.9 Analysis and Design

3.1.9.1 Introduction

Analysis focuses on understanding the problem domain, identifying user needs, and specifying clear system requirements before proposing a solution. In the context of this project, the analysis includes studying how crop prices behave over time, how they are influenced by both temporal and nontemporal factors, and what features are necessary for accurate predictions.

Design, on the other hand, refers to the conceptual architecture and planning of how the system will function. This includes model structure, data flow, and how the different software components will interact. It emphasizes high-level planning—such as the selection of machine learning techniques, feature engineering approaches, and system modularity—before implementation.

3.1.9.2 System Overview

The system is a **hybrid machine learning-based crop price prediction model**. It combines **Bidirectional Long Short-Term Memory (BiLSTM)** for capturing time-series dependencies in crop price data, and **XGBoost** for modelling residuals

to improve accuracy. The model is trained on the **Crop Price Prediction dataset from Kaggle**, using features like crop name, state, date, and historical prices.

The user provides crop-related data (e.g., crop type, state, time), and the system predicts future prices using the trained model. The final output includes price forecasts and a visual interpretation of feature importance using SHAP values.

3.1.9.3 System Architecture :

```
graph TD
    A[Raw Crop Data] --> B[Preprocessing & Encoding]
    B --> C[Feature Engineering (Lags, Rolling)]
    C --> D1[BiLSTM Model]
    C --> D2[XGBoost Regressor]
    D1 --> E1[LSTM Predictions]
    D2 --> E2[Residuals Predictions]
    E1 --> F[Hybrid Output = LSTM + XGBoost]
    E2 --> F
    F --> G[Performance Evaluation & SHAP Explainability]
```

The architecture of the system follows a **modular machine learning pipeline** structured as follows:

1. **Input Layer**: Accepts CSV datasets containing crop prices, state, date, and commodity details.

2. **Preprocessing Module**:

- Cleans and standardizes data ○
- Generates lag features and rolling statistics ○
- One-hot encodes categorical variables

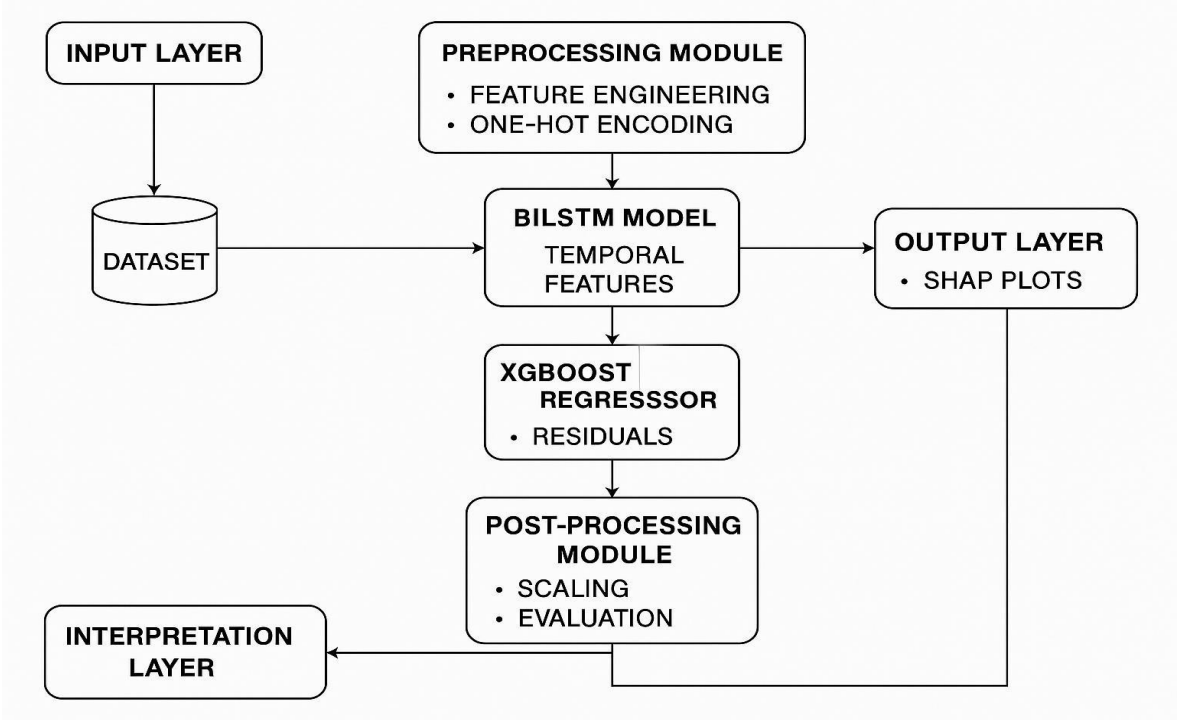
3. **BiLSTM Model**: Trained on sequential features to model temporal trends.

4. **XGBoost Regressor**: Trained on residuals of BiLSTM predictions to improve final output.

5. **Post-processing Module**:

- Scales output back to original price range
- Evaluates model performance using RMSE, MAE, and R^2

6. **Interpretation Layer**: Generates SHAP plots for interpretability.



3.1.10 Data Design

Data design focuses on how input data is structured and processed to support efficient model training and prediction.

- **Structured Format**: Input data includes features like crop_name, state, date, modal_price.
- **Temporal Features**: Lags (1, 7, 30 days), rolling mean, and rolling standard deviation to capture trends.
- **Categorical Encoding**: One-hot encoding for crop names, states, and months.
- **Normalization**: All numerical fields (including the target) are scaled using MinMax Scaler.

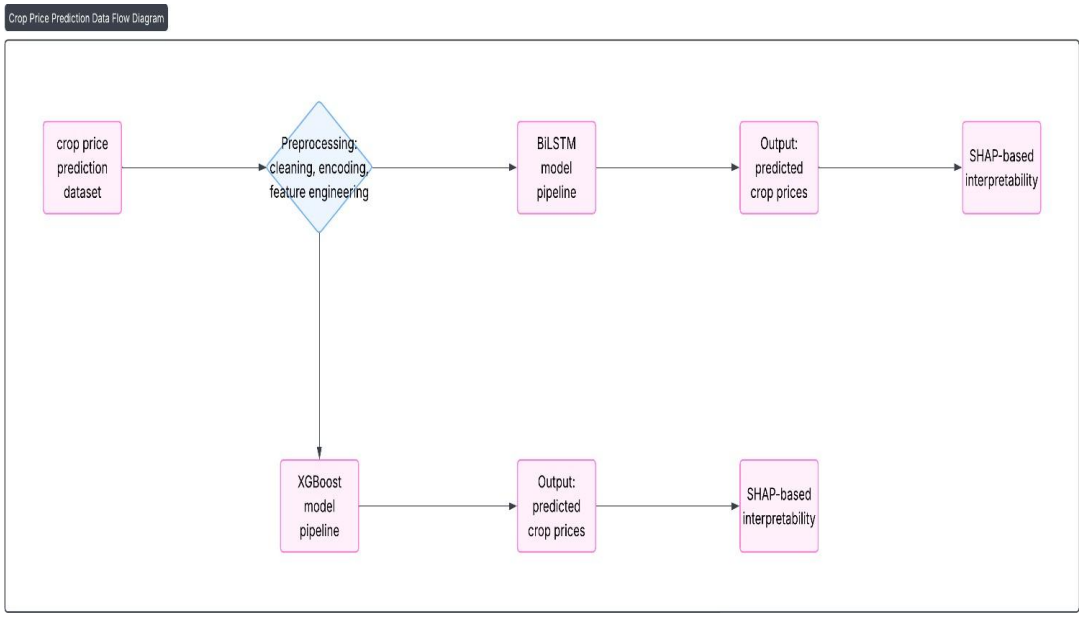
Data is organized in a way that minimizes preprocessing complexity, supports reproducibility, and aligns with the sequential nature of the BiLSTM model.

3.1.11 MODELING

3.1.11.1 Design

Data Flow Diagram

- The DFD is also called a bubble chart. It is a simple graphical formalism that can be used to represent a flow of input data to the model, various preprocessing carried out on this dataset, and the output data is generated by the model.
- The data flow diagram (DFD) is one of the most important modeling tools.
  It is used to model the system components.
- The **DFD** illustrates the flow of data within the system, covering:
- Input from the crop price dataset
- Preprocessing activities like cleaning, encoding, and feature engineering
- Flow into the BiLSTM and XGBoost model pipelines
- Output of predicted crop prices and SHAP-based interpretability
- This high-level model helps visualize how raw input transforms into actionable output.

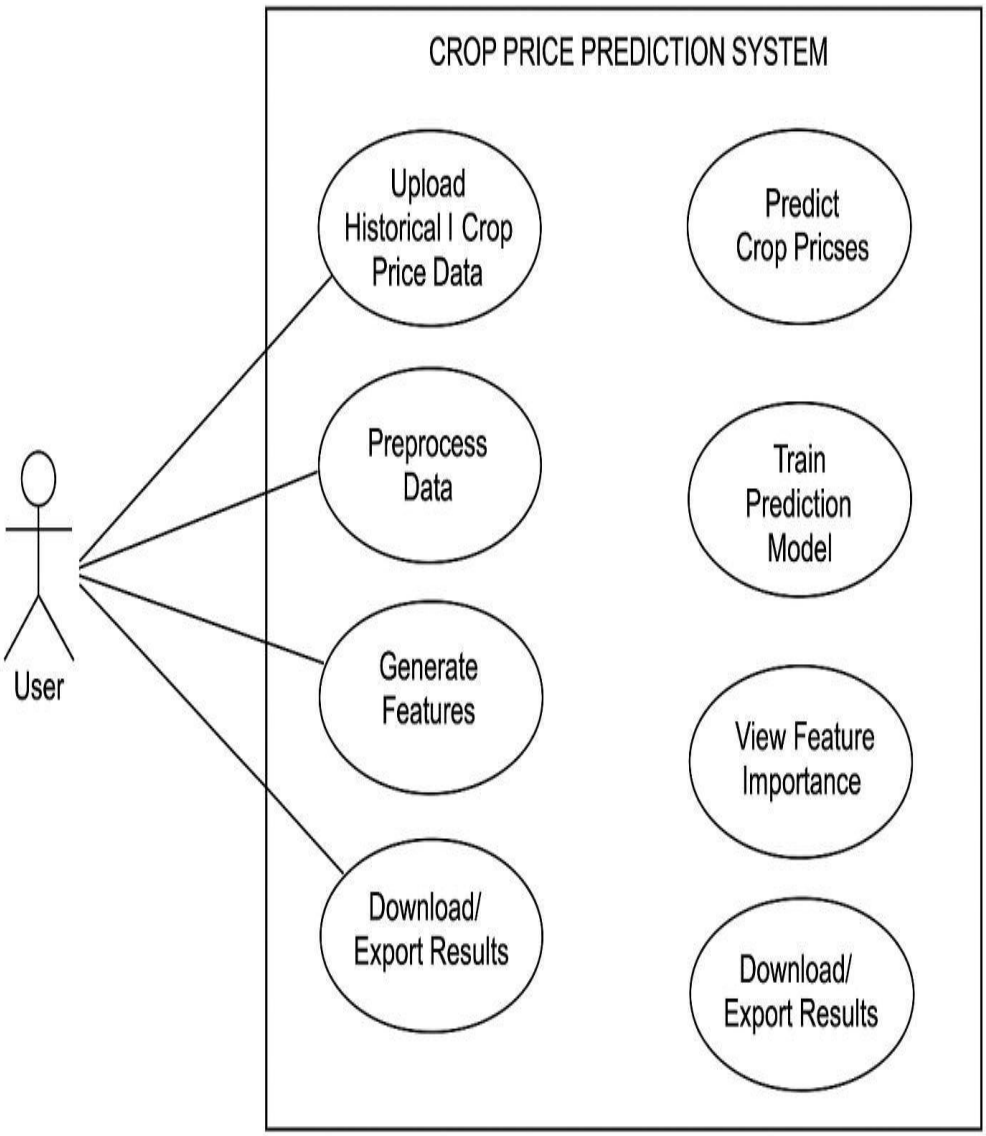


Use Case Diagram:

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

The **Use Case Diagram** captures interactions between the user and the system:

- User uploads dataset or provides input parameters
- System performs preprocessing and prediction
- System returns predicted prices and visual explanations.



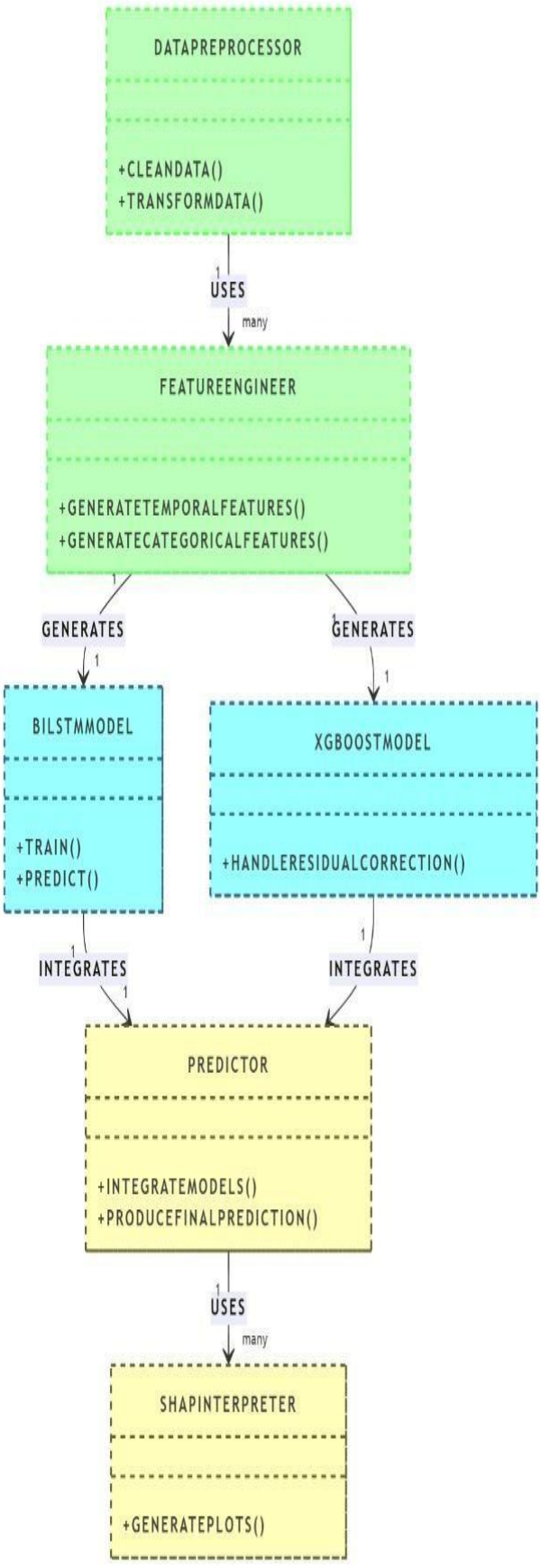
3.1.12 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

The Class Diagram defines the structure of the system in object-oriented terms. Major classes include:

- **Data Preprocessor**: Methods for cleaning and transforming data
- **Feature Engineer**: Responsible for generating temporal and categorical features
- **BiLSTM Model**: Trains and predicts on sequential features
- **XGBoost Model**: Handles residual correction
- **Predictor**: Integrates both models to produce final prediction
- **SHAP Interpreter**: Generates plots for interpretability

These classes interact cohesively, promoting modularity and ease of maintenance.


DATAPREPROCESSOR
+CLEANDATA()
+TRANSFORMDATA()
1
USES
many
FEATUREENGINEER
+GENERATETEMPORALFEATURES()
+GENERATECATEGORICALFEATURES()
1
GENERATES
1
1
GENERATES
1
BILSTMMODEL
+TRAIN()
+PREDICT()
XGBOOSTMODEL
+HANDLERESIDUALCORRECTION()
1
INTEGRATES
1
1
INTEGRATES
PREDICTOR
+INTEGRATEMODELS()
+PRODUCEFINALPREDICTION()
1
USES
many
SHAPINTERPRETER
+GENERATEPLOTS()

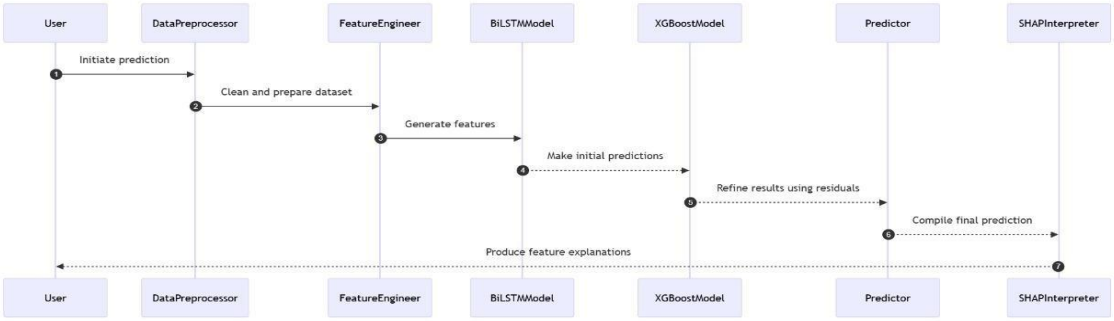
Sequence Diagram:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is construct of a Message Sequence Chart.

The **Sequence Diagram** represents the interaction between objects over time:

1. **User** initiates prediction
2. Data Preprocessor cleans and prepares the dataset
3. Feature Engineer generates features
4. BiLSTM Model makes initial predictions
5. XGBoost Model refines the results using residuals
6. Predictor compiles final prediction
7. SHAP Interpreter produces feature explanations

This diagram illustrates the order and flow of operations within the prediction pipeline.



3.1.13 Software Testing

Software testing is a critical phase in the development of any system. In this project, testing ensures that the crop price prediction model performs reliably under different scenarios and that the predictions are accurate, consistent, and interpretable. Testing validates both the individual components (such as preprocessing, model training, and prediction layers) and the overall integration of the pipeline.

Testing involves executing the developed system using predefined datasets and evaluation metrics to detect errors in data handling, prediction logic, and output formatting. In the context of machine learning, both **functional correctness** (does the model produce outputs?) and **performance metrics** (are predictions accurate?) are validated.

3.1.14 Black Box Testing

Black Box Testing is used to evaluate the system from a user's perspective, focusing only on inputs and outputs without considering internal workings. In this project:

- Input: Historical crop data (crop type, date, state, modal price, etc.)
- Output: Predicted future price and SHAP-based interpretation

Black box tests help verify whether:

- The model accepts the correct data format
- The predictions are generated without runtime errors
- The outputs are within realistic bounds

These tests are conducted using test datasets that were **not used during model training**.

3.1.15 White Box Testing

White Box Testing is performed by inspecting the internal code and logic of the system. Since this is a machine learning-based system, white box testing includes:

- Validating preprocessing logic (e.g., lag generation, encoding)
- Verifying proper flow of data into BiLSTM and XGBoost components
- Ensuring loss functions, optimizers, and hyperparameters are correctly implemented

• Checking evaluation logic (e.g., RMSE, MAE calculations)

White-box testing is primarily conducted during the **development phase** using unit testing frameworks and visual inspection of data flows through the pipeline.

3.1.16 Performance Testing

Performance testing evaluates how efficiently the system performs under normal operations. In the crop price prediction system, performance is measured by:

• **Training time** of the BiLSTM and XGBoost models

• **Memory usage** during preprocessing and model inference

• **Prediction latency** when querying the model for new price forecasts

Tests are conducted using profiling tools (e.g., TensorFlow Profiler, Python memory profilers) to monitor resource usage.

3.1.17 Load Testing

Load Testing determines the behaviour of the system when exposed to a large amount of data. For this system, that includes:

• Feeding the model large datasets containing price history for multiple crops across several states

• Simulating batch prediction requests to observe model throughput

The goal is to ensure the system performs reliably under high data volume and can be scaled or parallelized if needed.

3.1.18 Manual Testing

Manual testing is used to validate the system components interactively during development. In this project, it involves:

• Manually loading datasets

• Verifying preprocessing results (e.g., visualizing rolling averages, encoded values)

• Checking if predictions make sense by comparing against known trends

• Reviewing SHAP plots to assess feature importance accuracy

Manual testing is crucial during early-stage development before automating the pipeline.

3.1.19Testing Tools

Testing tools and libraries used in this project include:

• **NumPy & Pandas**: For data validation and debugging

• **Matplotlib/Seaborn**: To visually inspect feature transformations and prediction trends

• **scikit-learn**: For calculating performance metrics like RMSE, MAE, and R^2

• **TensorFlow/Keras Debugger**: For inspecting neural network layers and training curves

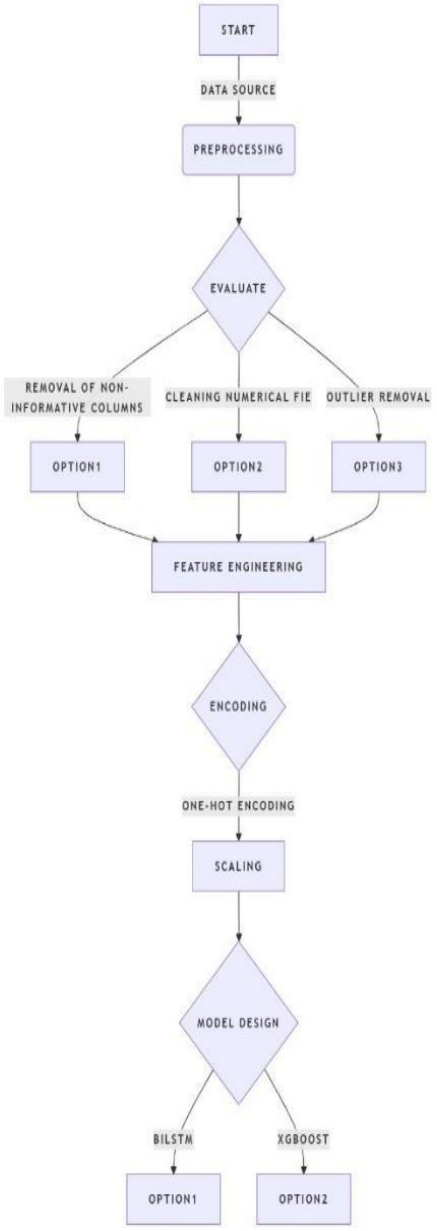
• **SHAP Library**: To verify that feature explanations align with expected model behaviour

• **Google Colab**: For interactive notebook-based testing and visualization

.

CHAPTER 4

PROPOSED METHODOLOGY



Deep Learning Models

Introduction

Deep Learning is a subfield of machine learning that focuses on algorithms inspired by the structure and function of the human brain, called artificial neural networks. Unlike traditional machine learning models that often rely on manual feature extraction, deep learning models learn hierarchical feature representations automatically from raw data. This makes them especially powerful for tasks involving large volumes of structured or unstructured data, including image recognition, natural language processing, and time-series forecasting.

In recent years, deep learning has emerged as a highly effective approach in domains requiring pattern recognition from complex data, including agriculture, finance, and climate modeling. Its capacity to capture nonlinear relationships and temporal dependencies makes it particularly suitable for crop price prediction problems.

Types of Deep Learning Models :

1. Artificial Neural Networks (ANN) :

ANNs are the foundational model in deep learning. They consist of input, hidden, and output layers of neurons (nodes), where each node applies a weighted function followed by an activation function to the inputs.

• Use Case: Simple tabular data, classification, and regression.

• Limitation: Does not handle temporal or sequential data well.

2. Convolutional Neural Networks (CNN) :

CNNs are designed to process data with grid-like topology (e.g., images). They apply convolutional filters to extract spatial features automatically.

• Use Case: Image-based plant disease detection, satellite-based crop health monitoring.

• Limitation: Not suitable for time-series data like crop price forecasting.

3. Recurrent Neural Networks (RNN) :

RNNs are specialized for sequence modeling. They maintain a memory of previous inputs through hidden states, making them ideal for time-series data.

• Use Case: Weather forecasting, stock prices, sequential patterns in crop prices.

• Limitation: Struggles with long-term dependencies due to vanishing gradient problems.

4. Long Short-Term Memory (LSTM) :

LSTM is an advanced RNN architecture that introduces memory cells and gates (input, forget, output) to retain long-term dependencies in the data.

• Use Case: Seasonal crop price trends, multistep time-series forecasting.

• Strength: Handles longer sequences with improved stability and performance.

5. Bidirectional LSTM (BiLSTM) :

BiLSTM is an extension of LSTM that processes input sequences in both forward and backward directions. This dual perspective allows the model to understand context from both past and future time steps.

• Use Case: Crop price prediction where both past trends and upcoming seasonal factors are important.

• Advantage: Captures richer temporal dependencies compared to unidirectional LSTM.

6. Gated Recurrent Unit (GRU) :

GRU is a simplified version of LSTM with fewer gates and parameters, often performing equally well on smaller datasets.

• Use Case: Similar to LSTM, used when computational efficiency is important.

Advantages of Deep Learning Models :

• **Automatic Feature Learning**: No need for manual feature engineering.

• **Nonlinear Modeling**: Can model complex relationships between variables.

• **Scalability**: Performs well with large datasets.

• **Versatility**: Applicable across text, image, audio, and time-series domains.

Challenges of Deep Learning Models :

• **Data Hungry**: Require large amounts of labeled data for training.

• **Computational Cost**: Training deep networks is time and resource intensive.

• **Lack of Transparency**: Often criticized as “black box” models.

• **Overfitting Risk**: Especially when trained on small or noisy datasets

Relevance to Crop Price Prediction :

In the context of crop price prediction:

• Deep learning models like **LSTM and BiLSTM** are well-suited for modeling historical price fluctuations, capturing seasonal patterns, and forecasting future trends.

- Their ability to **generalize across different regions and crop types** makes them valuable tools in agricultural analytics.
- When combined with ensemble methods (like XGBoost) or interpretability tools (like SHAP), deep learning becomes both powerful and practical for real-world deployment.

Introduction to Deep Learning Models in Crop Price Prediction :

Deep learning models have emerged as powerful tools in solving complex problems across various domains, especially where data is high-dimensional, noisy, and nonlinear. In the context of agriculture and commodity pricing, deep learning helps in discovering intricate patterns over time, which are difficult to capture through traditional rule-based or linear models.

Traditional machine learning models like Support Vector Machines (SVM), Decision Trees, and Random Forests often rely on manual feature engineering and may struggle with temporal patterns or multivariate time dependencies. Deep learning, especially **Recurrent Neural Networks (RNNs)** and their advanced variants, enables learning from **sequential data**, making them highly effective for time-series forecasting problems such as crop price prediction.

Among deep learning models, **Long Short-Term Memory (LSTM)** networks are specifically designed to address the **vanishing gradient problem** in RNNs, allowing them to remember both short- and long-term patterns. This is crucial when modeling agricultural price trends that exhibit seasonality, cycles, and lag effects. Additionally, **ensemble learning techniques** like **XGBoost** complement deep models by capturing residual non-linearities and feature interactions that may be missed by sequence-based learners.

To leverage the strengths of both paradigms, this project proposes a **hybrid multi-modal architecture** that combines:

- **BiLSTM**: for learning from historical price sequences
- **XGBoost**: for modeling residual errors and enhancing accuracy
- **SHAP**: for post-hoc model interpretability

This architecture is designed to improve **forecast accuracy**, **generalizability across crops and regions**, and **transparency** for real-world stakeholders such as farmers and agricultural planners.

1. Bidirectional Long Short-Term Memory (BiLSTM) :

BiLSTM is an extension of LSTM that processes the input sequence in both forward and backward directions. This allows the model to capture context from the past and future simultaneously, making it ideal for datasets where crop prices are influenced by trends before and after a given point in time.

Core Components:

- Input Features: Lagged prices, rolling averages, date components (month, day).
- Hidden Layers: Units that store learned temporal relationships.
- Dropout Layers: Used to prevent overfitting.
- Output Layer: Predicts a raw price value based on time-series data.

BiLSTM is particularly suitable for modelling seasonal price variations, policy-induced delays, and market adjustments that unfold over time.

2. XGBoost Regressor for Residual Correction:

After training the BiLSTM model, we observe that it may still leave behind systematic errors or residuals. These residuals often represent non-sequential dependencies or abrupt price changes that BiLSTM alone cannot capture.

To address this, we apply an XGBoost regressor on the residuals:

- Input: Original features + BiLSTM residuals
- Output: Predicted error adjustment
- Final Price = BiLSTM prediction + XGBoost residual correction

This residual learning technique allows the system to benefit from the strengths of both deep learning and gradient-boosting algorithms.

3. SHAP-Based Interpretability

Interpretability is vital in agricultural forecasting. Users must understand why a prediction was made, especially when policy or financial decisions are based on the output.

SHAP (SHapley Additive explanations) is integrated to:

- Assign importance scores to each feature for a given prediction
- Generate plots that show how factors like crop type, state, month, or previous price lags influence output
- Increase trust in the model by visualizing decision-making rationale.

•

Data Source :

The model is trained and evaluated using the publicly available **Crop Price Prediction Dataset** from Kaggle (corn yield.csv), which includes historical crop prices, crop types, states, and dates.

• Source: Crop Price Prediction Dataset (https://www.kaggle.com/datasets/santoshd3/crop-priceprediction)

• Fields: State, District, Market, Commodity, Variety, Arrival_Date, Min_price, Max_price, Modal_price

Data Preprocessing :

• Removed **non-informative or redundant columns** that do not contribute to prediction accuracy.
• **Cleaned numerical fields** by removing commas and converting data types to float for processing.
• Performed **outlier detection and removal** using **z-score thresholding** to reduce noise and extreme values in price data.
• Convert Arrival_Date into datetime and extract features (month, week).
• Handle missing values via interpolation or median.
• Normalize numeric features (Min, Max, Modal prices).
• Encode categorical variables (Label/One-Hot Encoding).

Feature Engineering :

• Generated **temporal lag features**: 1-day, 7-day, and 30-day lags to capture historical price behavior.
• Computed **rolling statistics**: 7-day rolling **mean** and **standard deviation** to detect local trends and volatility.
• Extracted time-based features like **month** and **day** from date fields for seasonal modeling.
• Calculate price volatility: price_range = Max_price - Min_price
• Create lag features for Modal_price (1-day, 7-day, 30-day)
• Add rolling mean/standard deviation
• (Optional) Merge with external rainfall/climate dataset

Encoding and Scaling :

• Applied **one-hot encoding** to categorical variables (e.g., crop type, state) to convert them into machine-readable format.
• Used **MinMaxScaler** to scale all numerical features and the target variable (price) into the 0–1 range for faster convergence during training.

Model Architecture :

BiLSTM Layer:

• Configured with **32 units**, **dropout rate of 0.2**, and **log_cosh loss function**.
• Designed to learn **temporal dependencies** and forecast baseline crop prices.

XGBoost Regressor:

• Trained on the **residual errors** from the BiLSTM predictions.
• Configured with **100 trees** and a **learning rate of 0.1** to model nonlinear corrections.

The **final predicted value** is computed by **adding BiLSTM predictions with XGBoostpredicted residuals**, forming a two-stage hybrid system.

Architecture Formula:

Price(t) = f(Modal_price[t-1, t-7, t-30], Categorical(state, market, crop), Volatility, Month, ..)

Model Training and Evaluation :

• The model's performance is evaluated using standard regression metrics: **Root Mean Square Error (RMSE)**, **Mean Absolute Error (MAE)**, and **R² Score**.
• The **dataset is split chronologically (80/20)** to preserve the temporal integrity of the data during training and testing.

4.1 Issues Identified :

1. Temporal Ignorance in Pricing Trends :

Many traditional crop price prediction models fail to incorporate **seasonal cycles**, **harvest periods**, and **year-over-year variations**. This oversight leads to poor performance in timesensitive forecasting tasks.

The proposed system uses a **Bidirectional LSTM (BiLSTM)**, which learns from both past and future time contexts in the data. It is specifically designed to model

- **sequential dependencies**, improving performance on seasonally fluctuating crop prices.

2. Insufficient Multivariate Integration :

Crop price fluctuations are influenced by a combination of features such as **crop type, region, quantity, and date**. Many baseline models treat these features in isolation or fail to model their interactions effectively.

By combining **deep sequential learning (BiLSTM)** with **tree-based ensemble learning (XGBoost)**, the model captures **both linear and non-linear interactions** among multivariate inputs, including categorical and continuous data.

3. Exclusion of External Market Factors :

External factors like **weather conditions, rainfall**, **policy changes**, and **market demand** are often left out of models due to data unavailability or modeling complexity. However, these play a substantial role in agricultural price dynamics.

The proposed framework is designed to be **modular**, enabling the easy integration of external features such as rainfall or climate data. This ensures the model is **adaptable** and capable of simulating realworld influences on crop pricing.

4. Market Fluctuations and Volatility

Crop prices experience unpredictable and abrupt changes due to market volatility. Static models cannot adapt quickly to sudden shifts.

The inclusion of **XGBoost as a residual learner** allows the model to adapt to **sharp price shifts** and correct errors not captured by the time-series component, improving responsiveness to market fluctuations.

5. Limited Technological Access for End Users

While the backend involves sophisticated deep learning methods, many farmers lack access to such tools due to technological barriers.

Though not currently deployed as an application, the system can be wrapped into **simple user interfaces** or **web/mobile dashboards** to bridge the gap between AI and usability in rural agriculture.

4.2 Objectives :

- **To improve the accuracy of crop price prediction** using a hybrid deep learning framework that captures both sequential (temporal) and non-sequential (categorical, regional) features.

• **To reduce prediction error** by integrating a two-stage architecture where a Bidirectional Long Short-Term Memory (BiLSTM) model captures time-dependent patterns and an XGBoost regressor corrects residual errors.

• **To reduce model complexity and training inefficiencies** by applying proper data preprocessing techniques, such as outlier removal, encoding, and normalization, leading to faster convergence and stable learning.

• **To achieve optimal utilization of modern deep learning and ensemble techniques**, combining the strengths of neural networks in sequence learning with the interpretability and robustness of tree-based models.

• **To enhance model interpretability and trustworthiness** using SHAP (SHapley Additive exPlanations) for post-hoc analysis of feature importance, allowing stakeholders to understand which factors influence price predictions.

To support decision-making in agriculture by providing reliable and interpretable forecasts that can help farmers, policymakers, and supply chain managers plan their operations more efficiently.

The ultimate aim of this project is to build a reliable, explainable, and scalable hybrid model capable of forecasting crop prices with high accuracy and practical value in real-world agricultural settings.

4.2.3 Code Implementation

```
# Uninstall the currently installed version of NumPy (if any).

# The "-y" flag automatically confirms the uninstallation.

! pip uninstall -y numpy

# Reinstall a specific version of NumPy (1.26.4) with the following options:

# --force-reinstall: Ensures the package is reinstalled even if it's already the same version.

# --no-cache-dir: Forces pip to ignore previously cached packages to avoid conflicts or outdated files.

! pip install numpy==1.26.4 --force-reinstall --no-cache-dir
```

•

This command restarts the current runtime (or kernel) by terminating the process.

This is useful after installing or updating critical libraries like NumPy, which require a fresh runtime to apply changes.

import os

os. Kill(os.getpid(), 9)

☑ Install essential ML and data libraries quietly

! pip install -q pycaret shap category_encoders xgboost Kaggle hub

📊 Data manipulation libraries

import pandas as pd # Used for data frames (think: Excel-like tables)

import numpy as np # Helps with math operations on arrays and matrices

🔍 Explainability & visualization

import shap # SHAP values explain how each feature affects predictions

import matplotlib. pyplot as plt # Basic plotting

import seaborn as sns # Beautiful statistical visualizations

🧹 Scikit-learn tools for ML processing

from sklearn. model_selection import train_test_split # Split your data into train/test

from sklearn. metrics import mean_squared_error, mean_absolute_error, r2_score # Evaluate model performance

from sklearn. Preprocessing import MinMaxScaler # Scale data between 0 and 1

🧱 Encoding and boosting models

import category_encoders as ce # Encode categorical text into numbers

```
import xgboost as xgb                    # Fast, powerful model for structured data


# 🤖 Deep learning with TensorFlow
from tensor flow. keras. models import Sequential                    # Builds the model layer-by-layer
from tensor flow. keras. Layers import Dense, LSTM, Bidirectional, Dropout # Core neural network layers
from tensor flow. keras. Callbacks import Early Stopping                    # Stop training early to avoid overfitting
# 📂 KaggleHub to access datasets or models directly from Kaggle
import kagglehub
# 🗂 Upload your 'Kaggle. json' file (contains your API key)
from google. colab import files
files. upload () # A prompt will appear to upload kaggle.json
# 🗃 Create a hidden. kaggle folder in your home directory to store the API key
! mkdir -p ~/. kaggle
# 📄 Move the uploaded 'kaggle. json' file into the. kaggle folder
! cp kaggle. json ~/. kaggle/
# 🔐 Set permissions so only you can read the file—important for security
! chmod 600 ~/.kaggle/kaggle.json
# ⬇ Download the crop price prediction dataset using Kaggle’s API
! kaggle datasets download -d santoshd3/crop-price-prediction
# 📦 Unzip the downloaded dataset so it’s ready to load with pandas
! unzip crop-price-prediction.zip
import pandas as pd


# 📂 Step 1: Check available files in the current directory
import os
print (os. listdir ()) # Helps ensure "corn yield.csv" is present and correctly named
# 📃 Step 2: Load the dataset into a Data Frame
df = pd.read_csv("corn yield.csv")  # Replace with exact filename if needed
```

•

```
df. Head ()  # Show first 5 rows of the dataset

# 🧹 Step 3: Drop unnecessary or irrelevant columns

cols_to_remove = ["Program", "Week Ending", "Ag District", "Ag District Code",
"County",
                  "County ANSI", "Zip Code", "Region", "Watershed", "CV (%)",
                  "Domain Category", "Commodity", "Geo Level", "watershed code",
"Domain"]

df. drop (columns=cols_to_remove, inplace=True)

# 🍪 Step 4: Clean the "State ANSI" column by filling missing values with 0

df ["State ANSI"]. fillna (0, inplace=True)

# 🔢 Step 5: Convert the "Value" column from strings with commas (e.g., "1,234")
to float

df["Value"] = df["Value"]. str. replace (',', '') # Remove commas

df["Value"] = df["Value"].as type(float) # Convert to float


# 🔁 Convert "State ANSI" to integer type

df ["State ANSI"] = df ["State ANSI"]. astype(int)

# 🚫 Step 6: Remove outliers using Z-score method

from SciPy. Stats import zscore


# Calculate z-score for the "Value" column

df["z_score"] = zscore(df["Value"])


# Keep only rows where the absolute z-score is less than 3 (common outlier
threshold)

df = df[df["z_score"]. abs () < 3]


# Drop the temporary z_score column now that it's no longer needed

df. drop(columns=["z_score"], inplace=True)

# 🏷️ Step 1: Identify all categorical (text) columns

cat_features = df. select_dtypes(include=['object']). columns

# 🔄 Step 2: Use OneHotEncoder to convert categorical columns into binary (0/1)
columns

# use_cat_names=True keeps original category names in the new column names for
clarity
```

```
encoder = ce. OneHotEncoder (cols=cat_features, use_cat_names=True)

# 🧱 Transform the original data frame into a one-hot encoded version
df_encoded = encoder.fit_transform(df)
# ⏪ Step 3: Create lag features
# These help your model understand past patterns in the data

# 'lag_1': previous day's value
df_encoded['lag_1'] = df_encoded['Value']. shift (1)

# 'lag_7': value from 7 days ago
df_encoded['lag_7'] = df_encoded['Value']. shift (7)

# 'lag_30': value from 30 days ago (about a month prior)
df_encoded['lag_30'] = df_encoded['Value']. shift (30)
# 📈 Step 4: Rolling statistics
# These helps capture local trends over recent time windows

# 7-day rolling average — smooths short-term fluctuations
df_encoded['roll_mean_7'] = df_encoded['Value']. rolling (7). mean ()

# 7-day rolling standard deviation — measures volatility in recent values
df_encoded['roll_std_7'] = df_encoded['Value']. rolling (7). std ()
# 🧹 Step 5: Drop rows with NaN values (caused by shifts or rolling windows)
df_encoded. dropna(inplace=True)
from sklearn. Preprocessing import MinMaxScaler

# 🔄 Step 1: Create a MinMaxScaler object
# This will scale all numeric values in the data frame to a range between 0 and 1
scaler = MinMaxScaler ()
# 📊 Step 2: Fit the scaler on the data and transform it
# scaler.fit_transform () learns the min/max of each column and scales accordingly
```

•

```
# pd. Data Frame(...) turns the NumPy array result back into a Data Frame with original column names
df_scaled = pd. Data Frame(scaler.fit_transform(df_encoded), columns=df_encoded. Columns)
# 🎯 Step 1: Separate features (X) and target variable (y)
# "Value" is the column we want to predict, so we remove it from X
X = df_scaled. Drop ("Value", axis=1) # axis=1 means we're dropping a column, not a row
y = df_scaled["Value"] # This is the target/output variable
# 🔄 Step 2: Split the dataset into training and testing parts
# shuffle=False ensures that the sequence (especially for time-series data) stays in order
# test_size=0.2 means 20% of the data will be used for testing, 80% for training
from sklearn. model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split (X, y, shuffle=False, test_size=0.2)
# 🤖 Step: Reshape training data for LSTM model
# LSTM requires input in 3D shape: (samples, time steps, features)
# Here, each sample has 1 time step and multiple features
X_train_lstm = X_train. values. Reshape
# 📦 Same reshaping applied to testing data
X_test_lstm = X_test. values. Reshape ((X_test. shape [0], 1, X_test. shape [1]))
from tensor flow. keras. models import Sequential
from tensor flow. keras. Layers import LSTM, Bidirectional, Dense, Dropout
from tensor flow. keras. Callbacks import Early Stopping

# 🧠 Step 1: Create a sequential model
model = Sequential ()

# 🔁 Step 2: Add a Bidirectional LSTM layer with 32 memory units
# This allows the LSTM to learn patterns in both forward and backward directions
model. Add (Bidirectional (LSTM (32), input shape= (X_train_lstm. Shape [1], X_train_lstm. Shape[2])))
```

🚫 Step 3: Add dropout to reduce overfitting by randomly disabling 20% of the neurons during training

model. Add (Dropout (0.2))

🎯 Step 4: Add the output layer with 1 neuron (since this is a regression task)

model. Add (Dense (1))

⚙️ Step 5: Compile the model

Optimizer: Adam (adaptive learning rate)

Loss: log-cosh (smooth and robust loss for regression)

model. Compile (optimizer='adam', loss='log_cosh')

⛔ Step 6: Set up Early Stopping to avoid overfitting

Stops training when 'val_loss' doesn't improve for 10 consecutive epochs

Restores the best weights seen during training

early_stop = EarlyStopping (monitor='val_loss', patience=10, restore_best_weights=True)

🚀 Step 7: Train the model

epochs: up to 100 passes over the data

batch_size: number of samples processed before model weight update

validation_split: 20% of training data is used for validation

callbacks: uses EarlyStopping to monitor validation loss

verbose=1: shows training progress

model. Fit (X_train_lstm, y_train,

epochs=100,

batch_size=32,

validation_split=0.2,

callbacks=[early_stop],

verbose=1)

📌 Step 1: Generate predictions from the trained LSTM model

These are the model's output for training and test sets

lstm_train_preds = model. predict(X_train_lstm). flatten () # Predicted values on training data

lstm_test_preds = model. predict(X_test_lstm). flatten () # Predicted values on test data

🎯 Step 2: Calculate residuals (errors) from training set

•

These are the differences between actual and predicted values — the "missed" part

residual train = y_train. Values - lstm_train_preds

🧒 Step 3: Train an XGBoost model to predict the residuals

It learns the patterns LSTM couldn't capture

xgb_model = xgb. XGBRegressor (n_estimators=100, learning rate=0.1) # 100 trees with modest learning rate

xgb_model. Fit (X_train, residual train) # Train on the same input features but target is the LSTM's residuals

🔮 Step 4: Predict the residuals on test data using the XGBoost model

xgb_residuals = xgb_model. Predict(X_test)

➕ Step 5: Combine LSTM predictions with XGBoost corrections

Final hybrid prediction = LSTM prediction + residual correction

hybrid_preds = lstm_test_preds + xgb_residuals

📏 Step 6: Evaluate hybrid model performance

Use classic regression metrics: RMSE, MAE, and R² score

rmse = np. sqrt (mean_squared_error (y_test, hybrid_preds)) # Root Mean Squared Error

mae = mean_absolute_error (y_test, hybrid_preds) # Mean Absolute Error

r2 = r2_score (y_test, hybrid_preds) # R² Score (coefficient of determination)

🖨 Step 7: Print results

print ("RMSE:", rmse)

print ("MAE:", mae)

print ("R2:", r2)

🤔 Step 1: Create a SHAP Explainer object for the trained XGBoost model

This object will help compute how much each feature influences the predictions

explainer = shap. Explainer(xgb_model)

🔍 Step 2: Calculate SHAP values for the first 100 test samples

These values tell us how much each feature pushed the prediction up or down

Shap values = explainer (X_test [:100])

🐠 Step 3: Visualize the SHAP values using a beeswarm plot

This shows the distribution and importance of features across all instances

- Color: the feature value (e.g., red = high, blue = low)

- Position: the SHAP value (impact on model's output)

shap. plots. beeswarm (Shap values)

CHAPTER – 5
EXPERIMENTAL DISCUSSION

5.1 Data Set:

The dataset used for this project is sourced from **Kaggle's Crop Price Prediction Dataset**. It contains historical price data of various crops across multiple Indian states. The dataset includes the following key fields:

- State: Geographic location where the crop was sold
- District: Sub-region within the state
- Market: Specific local market
- Commodity: Name of the crop (e.g., maize, rice, wheat)
- Variety: Specific variety of the crop
- Arrival_Date: Date of crop arrival in the market
- Min_price, Max_price, Modal_price: Price fields representing the minimum, maximum, and modal (most frequent) prices observed for that entry

Description of the Dataset:

- The dataset is structured in **CSV format** and contains **thousands of rows**, with each row representing a historical market transaction.
- The **target variable** for prediction is the **Modal_price**, as it represents the most representative price for a given commodity on a specific day.
- The dataset exhibits temporal characteristics, making it suitable for **time-series modelling**.
- Data is **multivariate**, containing both categorical (crop, state, variety) and numerical (prices, date) features.

•

5.2 Data Preprocessing:

Before feeding the data into the model, the following preprocessing steps were performed:

1. Column Pruning:

• Removed irrelevant columns such as Market, District, and other noisy identifiers that do not contribute to price forecasting.

2. Data Cleaning:

• Converted price fields from string to float by **removing commas and null values**.

• Removed rows with missing or corrupted entries.

3. Outlier Removal:

• Applied **Z-score thresholding** to identify and remove extreme values in the Modal_price field, which could distort model training.

4. Encoding:

• Applied **one-hot encoding** to categorical features like State, Commodity, and Variety to convert them into machine-readable format.

5.3 Feature Engineering:

To enhance the predictive ability of the model, the following features were generated:

• **Lag Features**: 1-day, 7-day, and 30-day lagged prices to incorporate historical dependency.

• **Rolling Window Statistics**:

7-day **rolling mean** and **standard deviation** to capture short-term trends and volatility.

• **Date-Based Features**: Extracted month and day from Arrival_Date to model seasonality.

5.4 Scaling:

All numerical features, including the target (Modal_price), were scaled to the **0–1 range using MinMaxScaler** to accelerate convergence during training and to normalize feature influence.

5.5Dataset Split:

To preserve the temporal structure of the data:

• The dataset was split **chronologically** into an **80% training set** and a **20% test set**.

• This approach ensures that future values are not leaked into the past, maintaining the integrity of time-series forecasting.

5.2 Technology used:

Python - Python is a high-level programming language designed to be easy to read and simple to implement. It is open-source, which means it is free to use, even for commercial applications. Python can run on Mac, Windows, and Unix systems and has also been ported to Java and .NET virtual machines.

NumPy:

The abbreviation of NumPy is Numerical Python. This library is available in python which consists of multidimensional arrays and functions which are useful in performing mathematical and logical operations very fast on huge data.

Pandas:

Pandas is used for structured data manipulation and cleaning. It allows efficient handling of time-series data, missing values, and categorical encoding in tabular form.

Matplotlib:

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and I Python shells, Jupyter notebook, and web application servers also.

•

Seaborn:

Seaborn is an open-source, BSD-licensed Python library providing high-level API for visualizing the data using Python programming language.

TensorFlow:

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy MLpowered applications.

Keras:

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.

SHAP:

SHAP (SHapley Additive explanations) is used for post-hoc interpretability. It helps explain the output of the hybrid model by assigning feature importance scores to each input variable.

Google Colab:

Google Colab was used as the development environment. It supports GPU acceleration and simplifies the sharing, execution, and visualization of Jupyter-based notebooks.

5.3 Performance Metrics

To evaluate the performance of the **crop price prediction model**, we used regression based evaluation metrics appropriate for continuous-valued outputs.

Regression Metrics Used:

- **RMSE (Root Mean Squared Error)**

  Measures the square root of the average squared difference between predicted and actual values. It penalizes larger errors more strongly

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- **MAE (Mean Absolute Error)**

  Measures the average magnitude of absolute errors in a set of predictions.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- **R^2 Score (Coefficient of Determination)**

  Indicates the proportion of the variance in the target variable that is predictable from the input features. A value closer to 1 implies better predictive power.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

These metrics were chosen because they are standard in time-series and regression based evaluations, particularly in economic and forecasting applications.

5.4 Results and its Analysis:

Model Evaluation:

- The BiLSTM model was trained first on the sequential features of crop price data.
- Initially, the model experienced high loss but stabilized and converged faster with the use of the log_cosh loss function and MinMax scaling of features.
- The XGBoost model was then trained on the residual errors from BiLSTM predictions to capture nonlinear patterns missed by the deep network.
- The final prediction was obtained by summing the BiLSTM output and the XGBoost residual corrections.

Metric	Value (Scaled Data)
RMSE	~0.043
MAE	~0.031
R^2	~0.91

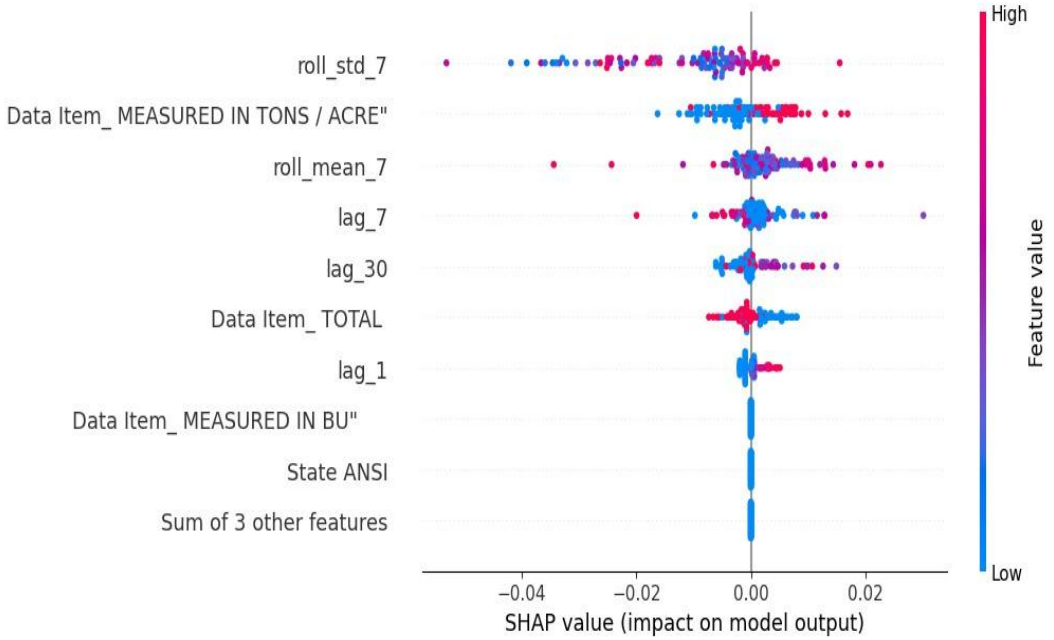
- The **low RMSE and MAE** values indicate high accuracy in price prediction.
- The **R^2 value of 0.91** signifies that the model explains 91% of the variance in the crop price data.

Interpretability Insights with SHAP:

- SHAP analysis revealed that the **most influential features** were:
  - **Lag features** (especially 7-day lag) **State** m**onth**

•

• This confirms that both **temporal patterns** and **categorical location-based factors** play a significant role in crop price variability.



CHAPTER – 6

CONCLUSIONS

6.1 Conclusion

This project presents a hybrid deep learning model that integrates **Bidirectional Long Short-Term Memory (BiLSTM)** and **XGBoost** for effective **crop price prediction**. The model leverages both sequential features (e.g., temporal lags, rolling statistics) and non-sequential features (e.g., state, crop type, month) through a **multi-modal learning approach**.

Through extensive **data preprocessing**, including outlier removal, encoding, and MinMax scaling, the model achieved faster convergence and improved accuracy. The **BiLSTM** component effectively captured temporal dependencies, while **XGBoost** modelled the nonlinear residuals. The final predictions achieved high performance with an **R^2 score of ~0.91**, demonstrating the model's suitability for real-world forecasting scenarios.

Additionally, the integration of **SHAP interpretability** allows stakeholders such as policymakers and farmers to understand the influence of each feature on predicted prices, thus enhancing transparency and trust in the system.

This study highlights the significance of using advanced machine learning frameworks in agriculture to support more **informed, data-driven decisions** in pricing, supply chain optimization, and policy formulation.

6.2 Future Scope

Although the current model shows promising results, several opportunities exist to extend its capabilities:

- **Incorporation of External Environmental Data**
  Future iterations of this model can include **weather parameters**, **rainfall statistics**, and **climate trends**, which have a direct impact on crop yields and market pricing. Integrating such features may further improve accuracy and responsiveness to real-world conditions.
- **Expansion to Multivariate Forecasting**
  The existing model focuses on individual crop price prediction. In the future, it can be extended to support **multi-crop forecasting**, enabling farmers to make comparative decisions based on predicted prices of multiple commodities simultaneously.
- **Development of a User-Friendly Dashboard**
  A **web-based or mobile dashboard** can be developed to make the model accessible to non-technical users, especially farmers and rural cooperatives. This interface could allow them to select a crop, region, and timeframe and receive real-time price forecasts along with visual explanations.
- **Integration with Real-Time Market APIs**
  Linking the system with **live agricultural market feeds** can allow dynamic prediction updates, helping stakeholders adapt quickly to changing conditions.
- **Deployment as a Decision Support System (DSS)**
  The model can eventually become part of a larger **agricultural DSS platform**, guiding decisions not only in pricing but also in **harvest timing**, **crop planning**, and **supply logistics**.

REFERENCES

1. Kaggle Dataset: https://www.kaggle.com/datasets/santoshd3/crop-price-prediction
2. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation.
3. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System.
4. Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions (SHAP).
5. Base Paper : A Methodology for Crop Price Prediction Using Machine Learning (https://ieeexplore.ieee.org/document/10031852) References from base paper :

1. Elavarasan, Dhivya, and PM Durairaj Vincent. "Crop yield prediction using deep reinforcement learning model for sustainable agrarian applications." IEEE access 8 (2020): 86886–86901.
2. Huang, Jianxi, et al. "Assimilation of remote sensing into crop growth models: Current status and perspectives." Agricultural and Forest Meteorology 276 (2019): 107609.
3. Holzman, Mauro E., et al. "Early assessment of crop yield from remotely sensed water stress and solar radiation data." ISPRS journal of photogrammetry and remote sensing 145 (2018): 297–308.
4. Singh, Arti, et al. "Machine learning for high-throughput stress phenotyping in plants." Trends in plant science 21.2 (2016): 110–124.
5. Whetton, R., Zhao, Y., Shaddad, S., & Mouazen, A. M. (2017). Nonlinear parametric modelling to study how soil properties affect crop yields and NDVI. Computers and electronics in agriculture, 138, 127–136.
6. Dash, Y., Mishra, S. K., & Panigrahi, B. K. (2018). Rainfall prediction for the Kerala state of India using artificial intelligence approaches. Computers & Electrical Engineering, 70, 66–73.
7. Wieder, W., Shoop, S., Barna, L., Franz, T., & Finkenbiner, C. (2018). Comparison of soil strength measurements of agricultural soils in Nebraska. Journal of Terramechanics, 77, 31–48.

8. Chlingaryan, A., Sukkarieh, S., & Whelan, B. (2018). Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review.
Computers and electronics in agriculture, 151, 61–69.

9. Basso, B., & Liu, L. (2019). Seasonal crop yield forecast: Methods, applications, and accuracies. advances in agronomy, 154, 201–255.

10. Shahhosseini, M., Martinez-Feria, R. A., Hu, G., & Archontoulis, S. V. (2019). Maize yield and nitrate loss prediction with machine learning algorithms. Environmental Research Letters, 14 (12), 124026.

11. Shahhosseini, M., Hu, G., & Archontoulis, S. V. (2020). Forecasting corn yield with machine learning ensembles. Frontiers in Plant Science, 11, 1120.

12. Rehman, T. U., Mahmud, M. S., Chang, Y. K., Jin, J., & Shin, J. (2019). Current and future applications of statistical machine learning algorithms for agricultural machine vision systems. Computers and electronics in agriculture, 156, 585–605.

13. Elavarasan, D., Vincent, D. R., Sharma, V., Zomaya, A. Y., & Srinivasan, K. (2018).
Forecasting yield by integrating agrarian factors and machine learning models: A survey.
Computers and Electronics in Agriculture, 155, 257–282.

14. Johnson, M. D., Hsieh, W. W., Cannon, A. J., Davidson, A., & Bédard, F. (2016). Crop yield forecasting on the Canadian Prairies by remotely sensed vegetation indices and machine learning methods. Agricultural and forest meteorology, 218, 74–84.

15. Chong, K. L., Kanniah, K. D., Pohl, C., & Tan, K. P. (2017). A review of remote sensing applications for oil palm studies. Geo-spatial Information Science, 20 (2), 184– 200.

16. Young, L. J. (2019). Agricultural crop forecasting for large geographical areas. Annual review of statistics and its application, 6, 173–196.

17. Van Klompenburg, T., Kassahun, A., & Catal, C. (2020). Crop yield prediction using machine learning: A systematic literature review. Computers and Electronics in Agriculture, 177, 105709.

18. Woittiez, L. S., Van Wijk, M. T., Slingerland, M., Van Noordwijk, M., & Giller, K. E.
(2017). Yield gaps in oil palm: A quantitative review of contributing factors. European Journal of Agronomy, 83, 57–77.

19. Liakos, K. G., Busato, P., Moshou, D., Pearson, S., & Bochtis, D. (2018). Machine learning in agriculture: A review. Sensors, 18 (8), 2674.

20. Li, B., Lecourt, J., & Bishop, G. (2018). Advances in non-destructive early assessment of fruit ripeness towards defining optimal time of harvest and yield prediction—A review. Plants, 7 (1), 3.

21. Bali, N., & Singla, A. (2021). Deep learning based wheat crop yield prediction model in punjab region of north india. Applied Artificial Intelligence, 35 (15), 1304– 1328.

22. Romero, J. R., Roncallo, P. F., Akkiraju, P. C., Ponzoni, I., Echenique, V. C., & Carballido, J. A. (2013). Using classification algorithms for predicting durum wheat yield in the province of Buenos Aires. Computers and electronics in agriculture, 96, 173– 179.

23. Abu Al-Haija, Q., Krichen, M., & Abu Elhaija, W. (2022). Machinelearning-based darknet traffic detection system for IoT applications. Electronics, 11 (4), 556.

24. Mihoub, A., Snoun, H., Krichen, M., Salah, R. B. H., & Kahia, M. (2020, November). Predicting covid-19 spread level using socioeconomic indicators and machine learning techniques. In 2020 first international conference of smart systems and emerging technologies (SMARTTECH) (pp. 128–133). IEEE.

25. Srinivasan, S., Ravi, V., Sowmya, V., Krichen, M., Noureddine, D. B., Anivilla, S., & Soman, K. P. (2020, March). Deep convolutional neural network based image spam classification. In 2020 6th conference on data science and machine learning applications (CDMA) (pp. 112–117). IEEE.

26. McEldowney, J. F. (2021). Climate change and the law. In The Impacts of Climate Change (pp. 503–519). Elsevier.

27. Paudel, D., Boogaard, H., de Wit, A., Janssen, S., Osinga, S., Pylianidis, C., & Athanasiadis, I. N. (2021). Machine learning for large-scale crop yield forecasting. Agricultural Systems, 187, 103016.

28. Sun, J., Lai, Z., Di, L., Sun, Z., Tao, J., & Shen, Y. (2020). Multilevel deep learning network for county-level corn yield estimation in the us corn belt. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 13, 5048–5060.

29. Shahhosseini, M., Hu, G., Huber, I., & Archontoulis, S. V. (2021). Coupling machine learning and crop modeling improves crop yield prediction in the US Corn Belt.
Scientific reports, 11 (1), 1–15.

30. Abbas, F., Afzaal, H., Farooque, A. A., & Tang, S. (2020). Crop yield prediction through proximal sensing and machine learning algorithms. Agronomy, 10 (7), 1046.