COMMODITY PRICE PREDICTION SYSTEM

Using Machine Learning and Deep Learning

Bachelor's Thesis Project Report (BTP1)

Submitted by:
Gaurav Kumar(22AG36012)

Under the supervision of:
Dr. Priyabrata Pradhan

Department of Agriculture and Food Engineering
IIT Kharagpur

INDIAN INSTITUTE OF TECHNOLOGY
Kharagpur

Agricultural and Food Engineering Department
Indian Institute of Technology Kharagpur

Autumn Semester, 2025-26

November 27, 2025

Certificate

This is to certify that the project titled **"Commodity Price Prediction Using Machine Learning and Deep Learning"** submitted by **Gaurav Kumar (22AG36012)** to IIT Kharagpur is a record of bonafide work carried out under my supervision.

Dr. Priyabrata Pradhan
Department of Agriculture and Food Engineering, IIT Kharagpur
Date: November 27, 2025

Abstract

This project develops a machine learning system for predicting agricultural commodity prices in West Bengal, India. Using historical data from 2014-2025 comprising 173,094 records across 18 districts and 61 markets, we implement two models: **XGBoost** and **Deep Neural Network**.

The XGBoost model achieves **5.43% MAPE** with **R^2 = 0.9453**, while the Neural Network achieves **4.64% MAPE** with **R^2 = 0.9601**. For 2025 predictions, the Neural Network achieves excellent **4.44% MAPE** with **R^2 = 0.9724**. The system predicts prices for Rice, Jute, and Wheat with 7-day forecasting capability.

A web application built with Flask and React provides an accessible interface for stakeholders to obtain price predictions.

Keywords: Machine Learning, XGBoost, Neural Networks, Price Prediction, Agriculture

Contents

Chapter 1

Introduction

1.1 Background

Agriculture employs over 50% of India's workforce. Price volatility significantly impacts farmers' livelihoods, leading to distress selling and financial instability. West Bengal, a major agricultural state, produces significant quantities of rice, jute, and wheat with complex pricing patterns.

1.2 Problem Statement

Develop an accurate commodity price prediction system that:

1. Predicts prices for Rice, Jute, and Wheat in West Bengal
2. Provides 7-day ahead forecasts
3. Incorporates weather, economic indicators, and historical patterns
4. Offers an accessible web interface

1.3 Objectives

1. Compile comprehensive historical price data (2014-2025)
2. Design relevant features capturing temporal and economic patterns
3. Implement XGBoost and Neural Network models
4. Develop a production-ready web application

Chapter 2

Methodology

2.1 Dataset

Table 2.1: Dataset Overview

Parameter	Value
Total Records	177,320
Time Period	2014-2025 (11 years)
Districts	18
Markets	61
Commodities	Rice, Jute, Wheat
Database Size	51.14 MB

Table 2.2: Commodity Distribution

Commodity	Records	%
Rice	130,572	75.4%
Jute	34,425	19.9%
Wheat	8,097	4.7%

Data sources include Agmarknet (prices), IMD (weather), RBI (economic indicators), and Ministry of Agriculture (MSP, production data).

2.2 Feature Engineering

We designed 36 features in the following categories:

Temporal Features: Year, month, day, quarter, day of week, weekend indicator, seasonal flags (monsoon, winter, summer).

Categorical Features: District, market, commodity, and variety (label encoded).

Economic Indicators: CPI, per capita income, food subsidy, MSP (Minimum Support Price).

Agricultural Parameters: Temperature, rainfall, area, production, yield, fertilizer consumption, export/import data.

Derived Features: Temperature-rainfall interaction, production efficiency ratios, CPI-MSP ratio.

Price Statistics: Commodity average, market average, district-commodity average, monthly averages.

2.3 Models

2.3.1 XGBoost (Extreme Gradient Boosting)

XGBoost builds an ensemble of decision trees sequentially, optimizing:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2.1}$$

where l is the loss function and Ω is the regularization term.

Key Hyperparameters: 1000 estimators, max depth 8, learning rate 0.05, L1/L2 regularization, GPU acceleration.

2.3.2 Neural Network

A 5-layer deep neural network with architecture:

- Input: 36 features
- Hidden layers: 256 $\rightarrow$ 128 $\rightarrow$ 64 $\rightarrow$ 32 $\rightarrow$ 16 neurons
- Activation: ReLU with BatchNorm and Dropout
- Output: 1 neuron (price prediction)
- Optimizer: Adam (learning rate 0.001)
- Early stopping with patience 20

2.4 Evaluation Metrics

- **MAE:** Mean Absolute Error (Rs)
- **RMSE:** Root Mean Squared Error (Rs)
- **MAPE:** Mean Absolute Percentage Error (%)
- **R^2:** Coefficient of Determination

Chapter 3

Results and Analysis

3.1 Model Performance

Table 3.1: Model Performance Comparison

Metric	XGBoost	Neural Network
MAE (Rs)	167.42	**143.73**
RMSE (Rs)	285.36	**253.66**
MAPE (%)	5.43	**4.64**
R^2 Score	0.9453	**0.9601**

Table 3.2: Prediction Accuracy Distribution

Error Threshold	XGBoost	Neural Network
Within 5%	89.2%	**92.4%**
Within 10%	97.5%	**98.1%**
Within 15%	99.2%	**99.5%**

3.2 Commodity-wise Performance

Table 3.3: MAPE by Commodity (%)

Commodity	XGBoost	Neural Network
Rice	5.21	**4.42**
Jute	5.89	**4.95**
Wheat	5.45	**4.68**

3.3 Validation Results

Sample predictions compared against actual database values:

Table 3.4: Sample Predictions vs Actual Prices

Commodity	District	Actual	XGBoost	NN
Wheat	North 24 Parganas	Rs 2,060	Rs 2,180	Rs 2,105
Jute	Murshidabad	Rs 5,880	Rs 6,120	Rs 5,894
Rice	Medinipur(W)	Rs 3,800	Rs 3,950	Rs 3,811

3.4 Feature Importance

Top features influencing predictions (XGBoost):

1. Commodity average price (18.7%)
2. Market average price (15.6%)
3. MSP - Minimum Support Price (13.4%)
4. Variety average price (9.8%)
5. CPI (8.7%)

3.5 Discussion

Neural Network Advantages:

- Superior accuracy (4.64% MAPE vs 5.43%)
- Excellent 2025 predictions (4.44% MAPE, R^2 = 0.9724)
- Effectively captures temporal and seasonal patterns
- Better generalization on unseen data

XGBoost Observations:

- Good performance (5.43% MAPE)
- Handles categorical features effectively
- Provides feature importance insights
- Useful as fallback model

Chapter 4

System Implementation

4.1 Architecture

The system follows a three-tier architecture:

- **Presentation Layer:** React.js frontend
- **Application Layer:** Flask REST API
- **Data Layer:** SQLite database + trained models

4.2 Technology Stack

Table 4.1: Technology Stack

Component	Technology
Frontend	React.js 18
Backend	Flask 3.1, Waitress WSGI
ML Framework	XGBoost 3.1.2, TensorFlow 2.20
Database	SQLite
Language	Python 3.10

4.3 Web Application Features

1. **Model Selection:** Choose between XGBoost or Neural Network
2. **Input Form:** Select commodity, district, market, variety, date
3. **7-Day Forecast:** Display predictions for current day + 6 days
4. **Responsive Design:** Works on desktop and mobile devices

4.4 API Endpoints

Table 4.2: REST API

Endpoint	**Method**	**Description**
/	GET	Main page
/predict	POST	Get price predictions
/get_markets	POST	Markets for district
/get_varieties	POST	Varieties for commodity

Chapter 5

Conclusion

5.1 Summary

This project successfully developed a commodity price prediction system achieving:

- Neural Network with **4.64% MAPE** and **R^2 = 0.9601**
- XGBoost model with **5.43% MAPE** and **R^2 = 0.9453**
- 2025 predictions: Neural Network achieves **4.44% MAPE**, **R^2 = 0.9724**
- Coverage of 18 districts, 61 markets, 3 commodities
- Production-ready web application

5.2 Contributions

1. Novel feature set combining economic and agricultural indicators
2. Comparative analysis of gradient boosting vs deep learning
3. Deployed system accessible to farmers and policymakers

5.3 Limitations

- Limited to West Bengal region
- Does not account for sudden policy changes or disasters
- Requires continuous data updates

5.4 Future Work

1. Implement LSTM for better temporal pattern capture
2. Extend coverage to pan-India
3. Add more commodities (vegetables, pulses)

4. Develop mobile application
5. Integrate real-time weather data APIs

Bibliography

[1] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *KDD*.

[2] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*.

[3] Srivastava, N. et al. (2014). Dropout: Preventing Overfitting. *JMLR*.

[4] Kingma, D. P., & Ba, J. (2014). Adam Optimizer. *arXiv:1412.6980*.

[5] Breiman, L. (2001). Random Forests. *Machine Learning*.

[6] Abadi, M. et al. (2016). TensorFlow. *OSDI*.

[7] Pedregosa, F. et al. (2011). Scikit-learn. *JMLR*.