Indian Institute of Technology Kharagpur

Department of Computer Science and Engineering

Commodity Price Prediction System Using Machine Learning and Deep Learning

Bachelor of Technology Project (BTP) Report

Submitted by:

Gaurav Kumar
B.Tech, Computer Science
IIT Kharagpur

Supervisor:

Prof. P. Pradhan
Department of CSE
IIT Kharagpur

November 2025

Certificate

This is to certify that the project titled **"Commodity Price Prediction System Using Machine Learning and Deep Learning"** submitted by **Gaurav Kumar** to the Indian Institute of Technology Kharagpur, is a record of bonafide work carried out under my supervision.

Prof. P. Pradhan
Department of CSE, IIT Kharagpur
Date: November 26, 2025

Abstract

Agricultural commodity price prediction is critical in developing economies like India. This project presents a comprehensive machine learning-based system for predicting commodity prices in West Bengal, utilizing historical data from 2014-2025.

We developed two predictive models: **XGBoost** and a **Deep Neural Network**. The system incorporates 36 features including temporal patterns, economic indicators, and agricultural parameters.

Key Results:

- XGBoost: MAPE = 1.46%, $R^2 = 0.9999$
- Neural Network: MAPE = 2.93%, $R^2 = 0.9573$

The system predicts prices for Rice, Jute, and Wheat across 18 districts and 61 markets.

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

Contents

Chapter 1

Introduction

1.1 Background

Agriculture employs over 50% of India's workforce. Price volatility significantly impacts farmers' livelihoods. West Bengal produces significant quantities of rice, jute, and wheat.

1.2 Problem Statement

Develop an accurate commodity price prediction system that can:

1. Predict prices for Rice, Jute, and Wheat in West Bengal
2. Provide 7-day forecasts
3. Incorporate weather, economic indicators, and historical patterns
4. Offer an accessible web interface

1.3 Objectives

1. Data Collection: Historical price data from 2014-2025
2. Feature Engineering: 36 features capturing temporal and economic patterns
3. Model Development: XGBoost and Neural Network models
4. Web Application: User-friendly prediction interface

Chapter 2

Literature Review

2.1 Traditional Approaches

ARIMA: Box and Jenkins (1970) - captures linear dependencies but struggles with non-linear patterns.

GARCH: Bollerslev (1986) - models volatility but has limitations in complex market dynamics.

2.2 Machine Learning Approaches

XGBoost (Chen & Guestrin, 2016):

- Winner of numerous ML competitions
- Handles missing values naturally
- Built-in regularization

Neural Networks:

- Universal approximation capability
- Backpropagation for training
- Dropout for regularization

Chapter 3

Methodology

3.1 Dataset

Table 3.1: Dataset Overview

Parameter	Value
Total Records	173,094
Time Period	2014-2025
Districts	18
Markets	61
Commodities	3 (Rice, Jute, Wheat)
Database Size	51.14 MB

Table 3.2: Commodity Distribution

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

3.2 Feature Engineering

36 Features organized as:

Temporal Features:

- year, month, day, quarter, day_of_week
- is_weekend, month_start, month_end
- is_monsoon, is_winter, is_summer

Economic Indicators:

- CPI, Per Capita Income, Food Subsidy, MSP

Agricultural Parameters:

- Temperature, Rainfall, Area, Production, Yield
- Fertilizer Consumption, Export, Import

3.3 XGBoost Model

Objective function:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{3.1}$$

Hyperparameters:

- n_estimators = 1000
- max_depth = 8
- learning_rate = 0.05
- GPU accelerated training

3.4 Neural Network Model

Architecture: 5 hidden layers (256-128-64-32-16 neurons)

```
Input (36 features)
  -> Dense(256) + BatchNorm + Dropout(0.3)
  -> Dense(128) + BatchNorm + Dropout(0.3)
  -> Dense(64) + BatchNorm + Dropout(0.2)
  -> Dense(32) + BatchNorm + Dropout(0.2)
  -> Dense(16) + BatchNorm + Dropout(0.1)
  -> Output (1 neuron)
```

Training: Adam optimizer, MSE loss, Early stopping

Chapter 4

Results and Analysis

4.1 Model Performance

Table 4.1: Performance Comparison

Metric	XGBoost	Neural Network
MAE (Rs)	48.23	113.15
RMSE (Rs)	89.45	212.11
MAPE (%)	**1.46**	2.93
R^2 Score	**0.9999**	0.9573

Table 4.2: Accuracy Distribution

Error Threshold	XGBoost	Neural Network
Within 5%	98.7%	85.6%
Within 10%	99.8%	97.0%
Within 15%	99.9%	99.3%

4.2 Sample Predictions

Table 4.3: Predictions vs Actual Prices

Commodity	District	Actual	XGBoost	NN
Jute	Malda	3,600	3,916	3,703
Jute	Murshidabad	3,500	3,825	3,575
Rice	Nadia	3,200	3,418	3,362

4.3 Feature Importance (XGBoost)

Top features:

1. commodity_avg_price (18.7%)

2. market_avg_price (15.6%)
3. MSP (13.4%)
4. variety_avg_price (9.8%)
5. CPI (8.7%)

Chapter 5

System Implementation

5.1 Technology Stack

Table 5.1: Technologies Used

Component	**Technology**
Frontend	React.js 18
Backend	Flask 3.1.0
ML Framework	XGBoost 3.1.2
Deep Learning	TensorFlow 2.20.0
Database	SQLite
Server	Waitress WSGI

5.2 API Endpoints

Table 5.2: REST API

Endpoint	**Method**	**Description**
/	GET	Serve frontend
/predict	POST	Get predictions
/get_markets	POST	Get markets for district
/get_varieties	POST	Get varieties

5.3 Web Interface Features

- Model selector (XGBoost / Neural Network)
- Cascading dropdowns (District → Market → Commodity → Variety)
- Date picker
- 7-day forecast display
- Responsive design

Chapter 6

Conclusion

6.1 Summary

This project successfully developed a commodity price prediction system achieving:

- XGBoost: **1.46% MAPE**, $R^2 = 0.9999$
- Neural Network: **2.93% MAPE**, $R^2 = 0.9573$
- Coverage: 18 districts, 61 markets, 3 commodities
- Production-ready web application

6.2 Contributions

1. Multi-source feature engineering combining economic and agricultural data
2. Comparative analysis of gradient boosting vs deep learning
3. Production-ready deployed system

6.3 Future Work

- LSTM networks for temporal patterns
- Pan-India coverage
- Mobile application
- Real-time data integration
- Price alert notifications

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: A Simple Way to Prevent Overfitting. JMLR.

[4] Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv.

[5] Abadi, M., et al. (2016). TensorFlow: Large-scale Machine Learning. OSDI.