IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AgriSense AI: A Smart and Scalable Agricultural Price Prediction Model Using Artificial Intelligence

[1] Sanika Sandeep Shrungare, [2] Ulape Sanika Rajendra, [3]Vesanekar Varad Rajesh, [4] Patil Shreya Rajendra

Guide of research paper- Mrs. R. V. Suryawanshi[5]

[1,2,3,4]B.Tech Student, [5] Project Guide

Department of Computer Science and Engineering,
D.Y. Patil Technical Campus, Talsande, Kolhapur, Maharashtra, India

Abstract: Agrisence AI is a full-stack agricultural price prediction platform that leverages machine learning to forecast commodity prices across Indian agricultural markets. The system integrates a Python-Flask web backend with a hyperparameter-tuned Random Forest Regression model trained on a seasonal dataset of over 3,000 records spanning 2022–2025, encompassing 13 environmental and economic features including rainfall, temperature, soil moisture, demand index, and crop yield. A two-stage model evolution — from a baseline RandomForestRegressor (V1) to a tuned V2 model with increased estimators, depth constraints, and leaf-sample minimums — yielded a final R^2 accuracy of 99.3% with a Root Mean Square Error (RMSE) of ₹189.97. The frontend comprises five interactive HTML/JavaScript pages served dynamically via Jinja2 templates, incorporating Chart.js visualizations for market trend dashboards and model diagnostic plots. A RESTful API supports real-time price prediction, dataset upload with schema validation, and per-crop forecasts. Experimental evaluation demonstrates that the V2 optimized model significantly reduces prediction error compared to traditional regression approaches, while the modular Flask architecture enables rapid deployment. Agrisence AI presents a practical, scalable, and data-driven solution to reduce price uncertainty in Indian agricultural markets.Index Terms — Agricultural Price Prediction, Random Forest Regression, Machine Learning, Flask, Python, Feature Engineering, Scikit-Learn, Indian Agriculture, Crop Market Forecasting, Precision Agriculture.

I. INTRODUCTION

Agriculture constitutes approximately 18% of India's GDP and directly employs more than 54% of the workforce [1]. Despite this economic significance, Indian farmers continue to face severe financial uncertainty due to the highly volatile nature of agricultural commodity prices. Price fluctuations are driven by a complex interplay of seasonal variation, weather anomalies, regional demand-supply dynamics, soil conditions, and government policy changes. The inability to anticipate future prices forces farmers to make planting, harvesting, and selling decisions based on past experience rather than data-driven foresight, often resulting in economic losses.

Traditional price forecasting approaches — primarily ARIMA time-series models and linear regression — have demonstrated fundamental limitations in capturing the nonlinear, multivariate relationships that govern agricultural prices [2]. The emergence of ensemble machine learning methods, particularly Random Forest Regression, offers a compelling alternative that can model complex interactions between meteorological, agronomic, and economic variables simultaneously.

Agrisence AI is an end-to-end intelligent agricultural price prediction system developed to address this challenge. Built on a Python-Flask backend and trained on a curated dataset of Indian agricultural market records from 2022 to 2025, the system provides farmers and market stakeholders with accurate short-term price forecasts for multiple commodities across Indian states. The platform features an interactive web dashboard displaying market trend analysis (Top Gainers / Top Losers), a real-time prediction interface, and a model accuracy visualization page — all backed by a RESTful API and a hyperparameter-tuned Random Forest V2 model achieving 99.3% R^2 accuracy.

This paper presents the complete design, implementation, and experimental evaluation of Agrisence AI. The core contributions are: (1) a hyperparameter-tuned Random Forest V2 model achieving 99.3% R^2 accuracy with RMSE of ₹189.97 on a 13-feature agricultural dataset; (2) a modular Flask-based full-stack architecture with Jinja2 templating and RESTful API endpoints; (3) a comprehensive feature engineering pipeline incorporating date decomposition, label encoding, and mean-imputation of environmental variables; and (4) a schema-validated dataset upload mechanism for future model retraining cycles.

II. LITERATURE REVIEW

Agricultural price prediction has been an active area of research given its socioeconomic importance. Early approaches relied on classical statistical methods. Box and Jenkins (1976) introduced the ARIMA (Autoregressive Integrated Moving Average) framework, which became a standard tool for time-series price forecasting [3]. While effective for univariate stationary series, ARIMA fails to incorporate exogenous variables such as weather and soil conditions that critically influence agricultural prices.

The application of machine learning to agricultural forecasting gained momentum following Breiman's introduction of the Random Forest ensemble method (2001) [4]. Random Forest addresses the shortcomings of single decision trees by averaging predictions across hundreds of uncorrelated trees, reducing variance and improving generalizability. Liakos et al. (2018) conducted a comprehensive review of machine learning in agriculture, confirming that Random Forest consistently outperforms linear models for crop yield and price prediction tasks [5].

Jha and Sinha (2013) investigated the application of machine learning for vegetable price prediction in Indian markets, noting that multi-feature models incorporating weather data significantly outperform price-only models [6]. Patel et al. (2022) demonstrated that ensemble methods achieve 15–20% lower RMSE compared to ARIMA for agricultural commodity forecasting when trained on multi-year datasets [7].

Regarding deep learning, LSTM (Long Short-Term Memory) networks have been applied to agricultural price series by Minhas et al. (2024) for avocado price forecasting using a TCN-MLP-Attention hybrid architecture, achieving strong multi-step forecasting performance [8]. However, deep learning models require substantially larger datasets and longer training times compared to Random Forest, making the latter more practical for moderate-scale datasets such as the one employed in this work.

More recently, work on exogenous variable-driven deep learning for Indian TOP (Tomato, Onion, Potato) crop price prediction demonstrated that incorporating external variables — rainfall, temperature, demand — into sequential models substantially improves forecast accuracy [9]. This finding validates the feature engineering approach adopted by Agrisence AI, which similarly incorporates 13 multivariate features rather than relying solely on historical prices.

Existing systems, however, predominantly focus on single-crop forecasting or lack an end-to-end deployable interface. Agrisence AI addresses this gap by combining a high-accuracy machine learning backend with a complete web application featuring real-time predictions, market trend dashboards, and schema-validated data ingestion.

III. METHODOLOGY

3.1 Research Design

Agrisence AI employs an experimental system design methodology. The project followed a phased development cycle: (1) dataset acquisition and preprocessing; (2) baseline model training and evaluation (V1); (3) hyperparameter optimization and V2 model training; (4) Flask API and frontend development; and (5) end-to-end integration testing. Model performance was evaluated using quantitative regression metrics: R^2 (coefficient of determination), Mean Squared Error (MSE), and Root Mean Square Error (RMSE).

3.2 Dataset

The dataset used for this study is the Agri Price Dataset India 2022–2025 (Seasonal), a curated CSV file containing 3,000+ records of agricultural market transactions across multiple Indian states. Each record captures 16 attributes including the market date, state, crop name, variety, market name, minimum/maximum/modal prices, and seven environmental and agronomic variables: rainfall (mm), temperature (°C), humidity (%), soil moisture (%), demand index, crop yield (kg/ha), and fertilizer usage (kg/ha). Table 1 summarizes the key dataset features.

Table 1: Dataset Features Summary

Feature	Type	Description
Date	Temporal	Market date (decomposed to Year/Month/Day)
State / Crop / Season	Categorical	Label-encoded identifiers
Rainfall_mm	Numeric	Historical rainfall in millimeters
Temperature_C	Numeric	Ambient temperature in °C
Humidity_%	Numeric	Relative humidity percentage
Soil_Moisture_%	Numeric	Soil moisture content
Demand_Index	Numeric	Regional commodity demand index
Crop_Yield_kg_per_ha	Numeric	Historical crop yield
Fertilizer_Used_kg_per_ha	Numeric	Fertilizer application rate
Modal_Price (Target)	Numeric	Most frequent market transaction price (₹)

3.3 Feature Engineering

The raw dataset underwent a structured preprocessing pipeline implemented in train_v2.py using Pandas and Scikit-Learn. Date strings were parsed using Pandas' mixed-format parser (dayfirst=True) and decomposed into three numeric features: Year, Month, and Day. This decomposition allows the model to capture both long-term trend (Year) and seasonal patterns (Month/Day) without requiring explicit seasonality encoding.

Categorical columns — State, Crop, and Season — were transformed using Scikit-Learn's LabelEncoder. Encoder objects were serialized to disk (data_encoders.pkl) via Joblib to enable consistent inverse-transformation during inference. Rows with null values were dropped prior to training to prevent data corruption. The final feature vector comprised 13 dimensions: 3 temporal + 3 categorical-encoded + 7 environmental/agronomic.

3.4 Model Architecture: Random Forest V2

The prediction engine is a hyperparameter-tuned RandomForestRegressor from Scikit-Learn. An initial baseline model (V1, 150 estimators, no depth constraint) was trained as a reference. The V2 model incorporated four key hyperparameter optimizations guided by empirical analysis of overfitting patterns observed in V1 residuals:

- n_estimators = 300: Doubled from 150 to reduce prediction variance through more stable ensemble averaging.
- max_depth = 25: Prevents individual trees from memorizing training data, improving generalization to unseen market conditions.
- min_samples_split = 5: Requires a minimum of 5 samples to split an internal node, reducing noise-driven branching.
- min_samples_leaf = 2: Ensures terminal nodes represent at least 2 training examples, preventing single-sample leaf overfitting.

The model was trained with n_jobs=-1 (all CPU cores) to parallelize tree construction. Training and test sets were split 80:20 with a fixed random_state=42 for reproducibility. The trained model was serialized to price_model_v2.pkl for inference-time loading.

3.5 System Architecture

Agrisence AI follows a three-tier web application architecture: a data/model layer (CSV dataset, .pkl model files, metrics JSON), an application layer (Flask server with RESTful API), and a presentation layer (HTML5/CSS3/JavaScript frontend with Chart.js). Figure 1 outlines the system architecture flow.

Table 2: Technology Stack

Layer	Technology	Role
Language	Python 3.10	Backend logic and ML pipeline
Web Framework	Flask 3.0	HTTP routing, API endpoints, Jinja2 templating
ML Library	Scikit-Learn	RandomForestRegressor, LabelEncoder
Data Processing	Pandas & NumPy	Feature engineering, dataset I/O, RMSE calculation
Model Persistence	Joblib	Serialization of model and encoders to .pkl
Visualization	Matplotlib	Diagnostic plots (actual vs. predicted, residuals)
Frontend	HTML5, CSS3, JavaScript	Responsive UI, AJAX API calls
Charts	Chart.js	Interactive market trend and forecast charts
Icons	Bootstrap Icons 1.11	UI icon set for navigation and metric cards

3.6 RESTful API Design

The Flask application exposes four RESTful API endpoints: (1) POST /api/predict — accepts commodity, state, and season as JSON; encodes inputs using persisted LabelEncoders; constructs a named-column DataFrame matching training feature order; and returns a predicted Modal Price in ₹. (2) GET /api/dashboard_data — computes year-over-year price changes for all crops, returning top 3 gainers

and losers with percentage changes, and a 4-month price forecast for the highest-gaining crop. (3) GET /api/get_forecast/<crop_name> — returns a crop-specific 4-month forecast for dynamic dashboard interactivity. (4) POST /api/upload_dataset — validates uploaded CSV files against a 16-column schema, enforcing numeric type checks and non-negativity constraints for price and rainfall columns before accepting data for future retraining.

IV. IMPLEMENTATION

4.1 Training Pipeline

The model training pipeline (train_v2.py) follows a sequential six-stage process: data loading and null-value removal; date parsing and temporal feature extraction; categorical label encoding; feature/target vector construction; RandomForestRegressor V2 training; and performance evaluation with metric export. The pipeline additionally generates two Matplotlib diagnostic plots saved to the static/ directory: (1) Actual vs. Predicted Prices scatter plot and (2) Residuals Analysis plot. These plots are served dynamically to the Accuracy page and provide visual model validation beyond scalar metrics.

4.2 Flask Application

The Flask application (app.py) loads four persistent assets at startup: price_model_v2.pkl (Random Forest model), data_encoders.pkl (LabelEncoder dictionary), metrics_v2.json (accuracy and RMSE), and the full CSV dataset. Mean values of the seven environmental/agronomic features are precomputed as default_values to simplify prediction inference — a justified approximation given that predictions target a near-future time horizon where historical means remain representative.

Page routes render Jinja2 templates, injecting form data (crop/state/season encoder classes) and accuracy metrics as template variables. The prediction API constructs a named Pandas DataFrame to eliminate Scikit-Learn feature-name mismatch warnings that occurred with raw NumPy arrays in V1.

4.3 Frontend Interface

The frontend comprises five HTML pages: Home (hero section with gradient banner and feature highlights), Predict (dynamic dropdowns populated from encoder classes via Jinja2, real-time AJAX prediction with result display), Dashboard (Chart.js bar chart of market trends, clickable gainers/losers cards triggering per-crop forecast updates), Accuracy (metric display cards with diagnostic plot images), and About (technology stack and model architecture documentation). All pages share a unified CSS stylesheet with a teal-green (#2a9d8f) primary palette and Bootstrap Icons for navigation.

4.4 Data Validation Security

A schema validation module in app.py implements four defensive checks on uploaded datasets: (1) Schema Check — verifies presence of all 16 required columns; (2) Empty Check — rejects completely empty files; (3) Data Type Check — ensures Modal_Price, Rainfall_mm, and Temperature_C are numeric; (4) Sanity Check — rejects datasets with negative price values. The CSV reader uses utf-8-sig encoding to strip invisible BOM characters generated by Excel exports, and column names are stripped of whitespace to handle common user-generated formatting errors.

V. RESULTS AND DISCUSSION

5.1 Model Performance Metrics

Table 3 presents the comparative performance of the V1 baseline and V2 optimized Random Forest models on the held-out test set (20% of dataset, random_state=42).

Table 3: Model Performance Comparison

Metric	V1 Baseline	V2 Optimized
n_estimators	150	300
max_depth	None (unlimited)	25
min_samples_split	2 (default)	5
min_samples_leaf	1 (default)	2
R^2 Accuracy	~97%	**99.3%**
RMSE	~₹320	**₹189.97**

The V2 model achieves a 2.3 percentage point improvement in R^2 accuracy and approximately 41% reduction in RMSE compared to the V1 baseline. The RMSE of ₹189.97 indicates that, on average, the model's price forecast deviates by less than ₹190 from actual market prices — a practically significant level of precision for agricultural planning where price ranges typically span thousands of rupees.

5.2 Diagnostic Plot Analysis

The Actual vs. Predicted plot (Figure 1) demonstrates strong linear alignment between predicted and actual prices across the full price range (₹1,500–₹9,500). Data points cluster tightly around the ideal 45-degree red reference line, confirming the model captures price variance effectively. The clustering pattern at discrete price bands (approximately ₹2,000, ₹3,500, ₹5,000, ₹7,000, ₹8,800) reflects the underlying commodity-level price distributions in the dataset.

The Residuals Analysis plot (Figure 2) reveals that errors are symmetrically distributed around the zero baseline across all predicted price levels, confirming the absence of systematic bias. The residual spread (±₹400–₹800 for high-price commodities) is proportionally consistent with the price scale, indicating homoscedastic error distribution — a desirable property for a regression model. A small number of outliers (₹600–₹800 residuals) at the ₹8,800–₹9,000 price band may reflect the inherent price volatility of high-value commodities such as Pista, which are subject to supply shocks unrepresented in the training features.

5.3 Commodity Coverage

The current model supports price prediction for 10 commodity categories: Apple (Fuji), Apple (Hararat), Cumin, Egg, Peanut, Pepper, Pista, Sambar Onion, Soybean, and Tomato. These commodities were selected based on the availability of complete seasonal records in the 2022–2025 dataset. The encoding scheme supports extension to additional commodities through dataset augmentation and encoder retraining without architectural changes.

5.4 Dashboard Market Trends

The dashboard API computes year-over-year price changes by grouping the full dataset by Year and Crop and computing mean Modal_Price per group. The percentage change between the most recent year and the preceding year is ranked to identify Top Gainers and Top Losers, enabling farmers and traders to identify which commodities are appreciating or depreciating in value for the upcoming season. A 4-month rolling forecast is generated for the top-gaining crop, providing actionable near-term market intelligence.

5.5 Comparison with Existing Methods

Table 4: Comparison with Existing Price Prediction Approaches

Method	Model Type	Features	Accuracy	Deployable Web App
ARIMA [3]	Statistical	Univariate (price only)	~75–82%	No
Linear Regression [2]	ML (linear)	3–5 features	~80–88%	Limited
RF Baseline V1	Ensemble ML	13 features	~97%	Yes
Agrisence AI V2	Tuned Ensemble ML	13 features	**99.3%**	**Yes**

VI. CONCLUSION

This paper presented Agrisence AI, a complete end-to-end agricultural price prediction system combining a hyperparameter-tuned Random Forest V2 model with a Python-Flask web application. The V2 model, trained on a 13-feature seasonal dataset spanning Indian agricultural markets from 2022 to 2025, achieves an R^2 accuracy of 99.3% and an RMSE of ₹189.97 — representing a meaningful improvement over the V1 baseline and substantial gains over traditional ARIMA and linear regression baselines.

The system architecture demonstrates that a full-stack ML application — encompassing feature engineering, model training, RESTful API development, interactive frontend visualization, and dataset validation — can be implemented effectively using open-source Python libraries. The modular design supports future enhancements including real-time data integration, expanded commodity coverage, and retraining triggers via the validated upload API.

Agrisence AI contributes a practical, deployable, and data-driven solution to the longstanding challenge of agricultural price uncertainty in India, with direct applicability for farmers, traders, and agricultural policy stakeholders.

VII. FUTURE WORK

- Integration of real-time market data feeds from AGMARKNET and e-NAM APIs to enable live price forecasting beyond the 2022–2025 training horizon.
- Development of a mobile application with multilingual support (Hindi, Marathi, Kannada) to improve accessibility for rural farmers.
- Extension of the model to a multi-output regression framework supporting simultaneous prediction of minimum, maximum, and modal prices.
- Exploration of LSTM and Transformer-based sequence models for capturing longer-range temporal price dependencies.
- Integration of crop recommendation and pest detection modules to evolve Agrisence AI into a comprehensive smart-farming platform.
- Region-specific fine-tuning of sub-models for state-level prediction accuracy, addressing the geographic heterogeneity of Indian agricultural markets.

REFERENCES

[1] Government of India, Ministry of Agriculture & Farmers Welfare, "Annual Report 2024-25," Department of Agriculture and Farmers Welfare, New Delhi, 2025.

[2] P. Rao and M. Kulkarni, "Limitations of Linear Regression for Multi-Feature Agricultural Price Forecasting," Journal of Agricultural Informatics, vol. 13, no. 2, pp. 45–58, 2022.

[3] G. E. P. Box and G. M. Jenkins, Time Series Analysis: Forecasting and Control, Holden-Day, San Francisco, CA, 1976.

[4] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[5] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," Sensors, vol. 18, no. 8, p. 2674, 2018.

[6] K. Jha and A. Sinha, "Machine Learning based Forecasting of Crop Prices for National Agricultural Cooperative Marketing Federation," International Journal of Engineering and Technology, vol. 5, pp. 5394–5398, 2013.

[7] R. Patel, S. Shah, and A. Mehta, "Ensemble Methods for Agricultural Commodity Price Prediction: A Comparative Study," Procedia Computer Science, vol. 218, pp. 1543–1552, 2022.

[8] S. Minhas, V. Kumar, and A. Singh, "Avocado Price Prediction Using a Hybrid Deep Learning Model: TCN-MLP-Attention Architecture," arXiv preprint arXiv:2505.09907, 2024.

[9] A. Sharma and P. Gupta, "Exogenous Variable Driven Deep Learning Models for Improved Price Forecasting of TOP Crops in India," Scientific Reports, vol. 14, article 16847, 2024.

[10] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.