Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 04 Issue: 10 | Oct-2026
ISSN (O) 3107-6696

AI-POWERED SMART MARKET INTELLIGENCE SYSTEM FOR FARMERS

Saranya L[1], Vinothini J[2], Vishnu Priya K[3], Yaaniga G[4], Nishmitha K[5]

1,2,3,4,5Department of Information Technology, Bachelor of Technology, Sri Shakthi Institute of Engineering and Technology (Autonomous) Coimbatore-641062

Abstract - Farmers often find it difficult to decide when and where to sell their crops because market prices change with season, demand, supply, weather and market location. Existing tools mostly show current or historical prices and do not guide the farmer about future prices or the best selling strategy. The AI-Powered Smart Market Intelligence System for Farmers is a Java-based full-stack application that supports data-driven marketing decisions. The system analyses historical commodity prices together with seasonal patterns, market arrivals, demand indicators and weather conditions, and uses a Java machine learning component to predict future prices. The prediction is then converted into an explainable recommendation such as Sell Today, Wait for Better Price, Store Commodity or Consider Another Market. The system provides a dashboard with current prices, price trends, predicted prices, market comparison, weather, risk level and expected profit. It is built with Java 17, Spring Boot, MySQL, the Smile machine learning library, Vue 3, Spring Security and JWT, and keeps all application data in the database instead of hard-coding it.

KEYWORDS: Smart Agriculture, Market Intelligence, Price Prediction, Machine Learning, Java, Spring Boot, Recommendation System, Decision Support

INTRODUCTION

Agriculture is the main livelihood of a large part of the population, yet the income of a farmer depends heavily on the price received at the time of sale. Commodity prices can change quickly because of seasonal changes, rainfall and temperature, the quantity of produce arriving at the market, and differences between nearby markets. A farmer who sells immediately may miss a better return that was available a few days later, or in a different market.

Farmers today can check prices through local traders, agricultural markets, government price portals, news and mobile applications. However, these sources mainly answer the question "what is the price now?". They do not answer the questions that matter most: Should I sell now or wait? Which market is better after transport cost? How much profit can I expect? Will the price go up or down?

This paper presents the AI-Powered Smart Market Intelligence System for Farmers, a Java-based full-stack decision-support platform. The backend is developed with Java 17 and Spring Boot, the data is stored in MySQL, the frontend is built with Vue 3, and the prediction component is implemented entirely in Java using the Smile machine learning library. Keeping the machine learning pipeline inside the Spring Boot backend allows easy integration between the model, REST APIs, database, authentication and business logic. The key idea is that the system does not only display a predicted number; it converts the prediction into a simple, understandable recommendation for the farmer.

PROBLEM STATEMENT

Farmers face uncertainty when selling agricultural commodities because market prices can change rapidly. The major problems are:

- Lack of future price information and difficulty in understanding historical price trends.
- Different prices in different markets and seasonal price fluctuations.
- Impact of weather on agricultural markets.
- Difficulty in estimating expected profit after transport, storage and other costs.
- Lack of technical knowledge for analysing market data.
- Existing applications show prices but do not give intelligent recommendations, so farmers may sell immediately even when waiting could give a better return.

OBJECTIVE

1. **Current Prices:** Provide farmers with current agricultural commodity market prices.
2. **Trend Analysis:** Analyse historical data to identify increasing, decreasing and stable price trends.
3. **Price Prediction:** Predict future commodity prices using Java-based machine learning, considering

seasonal patterns, weather, demand and market arrivals.

4. **Market Comparison:** Compare prices across markets and identify potentially better nearby markets.
5. **Profit Estimation:** Calculate expected profit from quantity, selling price, transportation and other costs.
6. **Explainable Recommendation:** Give a clear recommendation instead of only a predicted number.
7. **User-Friendly Dashboard:** Provide a simple and professional dashboard for farmers.
8. **History and Alerts:** Maintain prediction history and send notifications for important market changes.
9. **Administration:** Provide an administrator with tools to manage users, commodities, markets and market data.
10. **Security:** Secure the system with JWT authentication and role-based access control.

LITERATURE SURVEY

Market price information in India is publicly available through government portals such as Agmarknet, which publishes daily commodity prices and arrivals from agricultural markets [1]. Such portals are useful for reference, but they present recorded data and leave the interpretation and the selling decision to the farmer.

Price forecasting is a time-series problem. Standard forecasting practice builds predictors from past values (lag values), moving averages and seasonal components of the series [3]. For agricultural commodities, the repeating yearly cycle makes seasonal features particularly important, and cyclical (sine/cosine) encoding of month and day is a common way to represent this repetition numerically.

Among machine learning methods, linear regression gives a simple and interpretable baseline, while Random Forest regression combines many decision trees and is well known for handling non-linear relationships between features [2]. Smile is a machine learning library written for the JVM that provides regression models, which makes it possible to build the prediction pipeline completely in Java [4].

On the application side, REST architecture is widely used to connect a web frontend with a backend service [7], Spring Boot simplifies the development of such services [8], JSON Web Tokens give a compact way to carry authenticated claims between client and server [5], and the bcrypt scheme protects stored passwords with an adaptive hashing cost [6]. The proposed work combines these ideas into one decision-support system for farmers.

METHODOLOGY

The system follows a structured approach from data collection to the final recommendation. Historical market records (commodity, market, price, date, source, arrivals and demand indicator) form the primary dataset. Weather values such as temperature, humidity, rainfall and wind speed are obtained from an external weather service; if the service is temporarily unavailable, the latest cached data is used so that the application does not fail.

Data Processing and Feature Extraction:

Records are validated and sorted by date, and the FeatureExtractor converts them into a numerical feature vector. The current design uses 18 features:

- Day index • Price momentum • Market arrivals
- Price lag 1 • Season • Demand index
- Price lag 7 • Month sine • Temperature
- Price lag 30 • Month cosine • Humidity
- 7-day moving average • Day-of-week sine • Rainfall
- 30-day moving average • Day-of-week cosine • Perishable indicator

Lag features represent previous prices (yesterday, about 7 days earlier and about 30 days earlier) and help the model identify historical patterns. **Moving averages** show the general direction of the price; for example, the series ₹200, ₹210, ₹220, ₹225, ₹230, ₹240, ₹250 shows an increasing trend even when only one day is considered. **Seasonal features** (month, season and sine/cosine encoding) represent the repeating nature of months and days. **Price momentum** measures recent movement as (Recent Price − Older Price) / Older Price; a positive value means the price is increasing and a negative value means it is decreasing.

Model Training, Evaluation and Prediction:

The feature matrix is split into a training dataset and the Smile regression model (Linear Regression or Random Forest Regression, according to what the final implementation uses) is trained on it. The model is evaluated with standard regression metrics: **MAE** (Mean

Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 04 Issue: 10 | Oct-2026
ISSN (O) 3107-6696

Absolute Error), the average absolute difference between actual and predicted prices; **RMSE** (Root Mean Squared Error), which penalises large errors more strongly; and the **R2 score**, which shows how well the model explains the variation in the target prices. The trained model is then used to predict the future price for the selected commodity and market. Reported accuracy values are always the ones actually calculated by the implementation.

EXISTING SYSTEM

In the existing approach, farmers obtain market information from local traders, agricultural markets, news and announcements, government market-price portals, mobile applications, manual checking of prices and personal experience. These sources mainly provide the current price, the historical price and general market information. The farmer must still decide manually whether to sell now, whether to wait, which market is better, how much profit to expect and whether the price will increase or decrease. The existing approach does not combine all these factors into one intelligent decision-support system.

Disadvantages:

- Shows only current and historical prices; no future price prediction.
- No intelligent recommendation on selling time or market.
- Weather, season, demand and arrivals are not combined into the analysis.
- Prices of different markets are not compared with transport cost.
- No profit estimation for the farmer.
- Depends on personal experience and local traders, which may be inconsistent.
- Farmers need technical knowledge to analyse data themselves.

PROPOSED SYSTEM

The proposed system combines market data, historical trends, weather, seasonal information, demand and arrival indicators and machine learning. Raw data goes through data processing and feature extraction, the Java ML model produces the future price prediction, and the system then performs trend, risk and market comparison and profit calculation. Finally, the recommendation engine converts the result into a decision that the farmer can easily understand. The complete working flow is shown in Fig. 1.

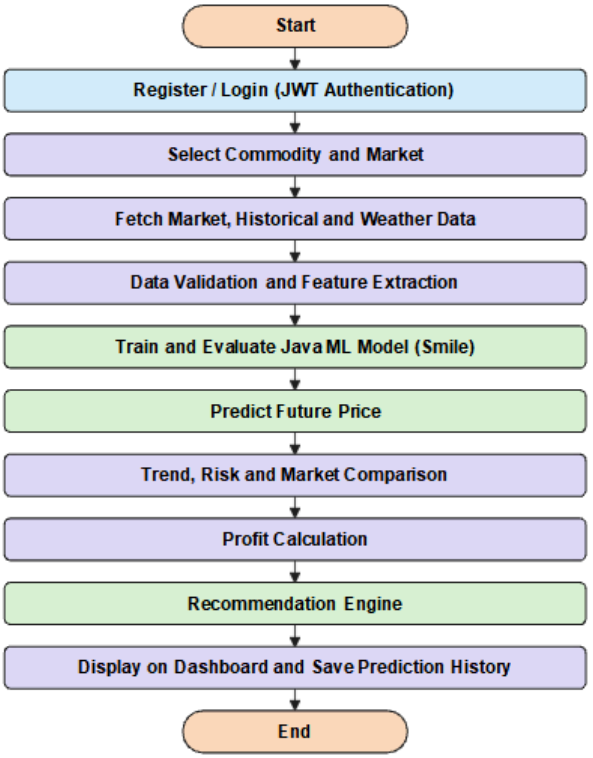

Fig. 1. Complete working flow of the proposed system

SYSTEM ARCHITECTURE

The system uses a layered architecture. The Vue 3 frontend communicates with the Spring Boot backend through REST/JSON using Axios. Inside the backend, Spring Security (JWT and BCrypt) handles authentication, the business services handle market, profit and notification logic, and the AI/ML module (Smile) feeds the recommendation engine. All data is stored in MySQL, and an external Weather API supplies weather data through the backend. A typical request passes through the Vue component, Axios, REST controller, service layer, repository and MySQL; for prediction the chain is PredictionController, PredictionService, MarketPriceRepository, FeatureExtractor, ModelTrainer, PredictionEngine and RecommendationEngine.

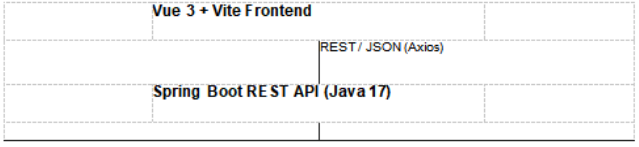

Fig. 2. System architecture

TECHNOLOGY STACK

Layer	Technologies	Purpose
Frontend	Vue 3, Vite, Axios, Vue Router, CSS, chart library	User interface, REST communication, navigation and visualisation of prices and predictions
Backend	Java 17, Spring Boot 3.2.4, Spring MVC, Spring Data JPA, Hibernate, Maven	REST APIs, business logic, database access and build management
Security	Spring Security, JWT, BCrypt	Authentication, authorization, role-based access and password hashing
Database	MySQL	Users, commodities, markets, prices, predictions, weather, favourites and notifications
AI / ML	Smile Java ML (Linear / Random Forest Regression)	Price prediction implemented fully in Java
Support	Lombok, Swagger/OpenAPI, Actuator, SLF4J/Logback	Less boilerplate code, API documentation, monitoring and logging

The ML component is implemented completely in Java so that the prediction pipeline remains inside the Spring Boot backend, which gives easier integration between the model, REST APIs, database, authentication and business logic than a separate Python service.

SYSTEM REQUIREMENTS

Hardware Requirements:

- Computer or laptop with Intel Core i5 processor or equivalent.
- Minimum 8 GB RAM recommended for smooth operation.
- At least 1 GB of free storage space for the application and data.
- Internet connection (for the weather service and web access).

Software Requirements:

- Java 17 and Spring Boot 3.2.4 (Maven)
- MySQL database
- Vue 3, Vite and Axios
- Smile Java Machine Learning library
- Eclipse / IntelliJ IDEA and a web browser

MODULE DESCRIPTION

1. User Authentication:

- User registration and login with BCrypt password hashing and JWT token generation.
- Role-based access control with two roles: FARMER and ADMIN.

2. Farmer Dashboard:

- Shows current price, predicted price, price trend, risk level, market comparison, weather, expected profit and the recommendation.
- All values come from the backend, database or API rather than being hard-coded.

3. Commodity and Market Management:

- Maintains commodity name, category, unit and description (for example rice, wheat, tomato, onion, potato).
- Maintains market name, location, district, state, latitude and longitude for price comparison.

4. Market Price Management:

- Stores commodity, market, price, date, source, arrivals and demand indicator.
- Historical records form the primary dataset for the ML component.

5. Weather Module:

- Obtains temperature, humidity, rainfall and wind speed and uses them as additional prediction features.
- Uses cached/latest available data if the external weather service is unavailable.

6. AI Prediction Module:

- Performs validation, sorting, feature extraction, training, evaluation and prediction with the Smile model.

7. Recommendation Engine:

- Combines current price, predicted price, trend, risk, weather, market comparison and expected profit to produce the final recommendation.

8. Prediction History and Notifications:

- Stores commodity, market, predicted price, recommendation, date and model information so past predictions can be reviewed.
- Notifies users about significant price changes, prediction updates and favourable market opportunities.

9. Admin Module:

- Manages users, roles, commodities, markets and market data (add, update, delete, data-quality monitoring).
- Monitors application health, logs, model status and data availability.

RECOMMENDATION ENGINE, MARKET COMPARISON AND PROFIT CALCULATION

The recommendation engine is what makes the system more useful than a simple prediction application. It considers the current price, predicted price, trend, risk, weather, market comparison and expected profit, and produces one of four recommendations:

- **Sell Today** – when current market conditions are favourable.
- **Wait for Better Price** – when the prediction and trend indicate a potentially higher future price.
- **Store Commodity** – when storage is feasible and the expected future price may justify waiting.
- **Consider Another Market** – when another market offers a better return after relevant costs.

The highest market price alone is not automatically the best choice. For example, if Market B offers ₹2,650 and the transport cost is ₹200, the effective return is ₹2,450, so the system helps decide whether moving the commodity is actually beneficial. The profit calculator takes quantity, selling price, transport cost, storage cost and other costs: Revenue = Quantity × Selling Price; Total Cost = Transport + Storage + Other Costs; Estimated Profit = Revenue − Total Cost.

DATABASE DESIGN AND SECURITY

The important entities are User, Commodity, Market, MarketPrice, PredictionResult, WeatherData, FavoriteCommodity and Notification. A user can have favourite commodities, prediction results and notifications; commodities and markets are linked through MarketPrice records, which hold the historical data used for prediction.

After login, the username and password are verified by Spring Security and a JWT token is returned to the frontend. For later API calls the Vue application sends the token in the Authorization: Bearer header, Spring Security validates it, and only then the request reaches the controller. Passwords are never stored as plain text; they are hashed with BCrypt.

CONCLUSION

The AI-Powered Smart Market Intelligence System for Farmers moves beyond simply displaying prices. By combining historical prices, seasonal patterns, weather, arrivals and demand with a Java-based machine learning model, it predicts future prices and turns the prediction into an explainable recommendation supported by trend, risk, market comparison and profit calculation. The Java 17, Spring Boot, MySQL and Vue 3 architecture is dynamic and database-driven, secured with JWT and BCrypt, and can be extended with more commodities, markets and data sources. The model quality will be reported using the MAE, RMSE and R2 values calculated by the final implementation. In future, the system can be extended with mobile application support, wider market and weather coverage and more advanced prediction models.

REFERENCES

[1]. Agmarknet, Directorate of Marketing and Inspection, Ministry of Agriculture and Farmers Welfare, Government of India. https://agmarknet.gov.in

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

[3]. R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 3rd ed., OTexts, 2021. [4]. H. Li, "Smile: Statistical Machine Intelligence and Learning Engine." https://haifengl.github.io

[5]. M. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, IETF, 2015.

[6]. N. Provos and D. Mazières, "A Future-Adaptable Password Scheme," *Proc. USENIX Annual Technical Conference*, 1999.

[7]. R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[8]. Spring Boot Reference Documentation. https://spring.io/projects/spring-boot