ISSN: **3107-6513**

AI-Driven Autonomous Decision Engine for Precision Agriculture

[1]*Sri Dhanya M*, [2]*Ramaraj M*

Department of Computer Science, Rathinam College of Arts and Science (Autonomous), Coimbatore,

Tamil Nadu, India

[1]*dsri59326@gmail.com*, [2]*ramaraj.phdcs@gmail.com*

Abstract-*The AI-Driven Autonomous Decision Engine for Precision Agriculture is a smart farming support system designed to improve agricultural productivity through data-driven recommendations. The project integrates machine learning, simulated IoT-style sensor data, weather APIs, and geolocation services to support crop yield prediction, pest risk analysis, and irrigation planning in a single decision-support workflow. The system addresses the limitations of traditional agriculture, where decisions are often made through manual observation and experience without continuous monitoring or predictive analytics. By processing soil parameters, rainfall, temperature, humidity, and nutrient-related inputs, the platform provides timely recommendations that help farmers optimize resource usage and improve crop performance. The implementation uses a React-based frontend, a Flask backend, and machine learning models including Random Forest, Support Vector Machine, regression methods, and decision-tree-based logic for selected prediction tasks. The project also includes fallback heuristics to maintain functionality when live API data is unavailable, making the system more reliable in practical deployment settings.*

Keywords: *Precision Agriculture, Machine Learning, Flask, React, Crop Yield Prediction, Pest Risk Analysis, Irrigation Recommendation, Smart Farming.*

I. Introduction

Agriculture remains a critical sector, but conventional farming still depends heavily on manual judgment, delayed observations, and generalized practices. These limitations often lead to inefficient irrigation, poor pest response, inconsistent crop planning, and waste of resources.

This project proposes an AI-driven precision agriculture system that combines environmental sensing inputs, real-time weather information, and machine learning models to generate actionable insights. The main goal is to help farmers make better decisions related to yield estimation, irrigation scheduling, and pest prevention using a simple digital interface.

The project is especially relevant in the context of climate variability and region-specific farming needs. By using location-aware recommendations and predictive analytics, the system shifts agricultural decision-making from reactive practice to proactive planning.

II. Existing and Proposed System

The existing system mainly relies on traditional agricultural practices, manual observation, and farmer experience. Such methods usually lack real-time data integration and predictive intelligence, which reduces the accuracy and timeliness of farming decisions.

The proposed system introduces an integrated architecture built with a frontend for user interaction, a Flask backend for processing, and predictive modules for crop yield, pest risk, and irrigation recommendation. Weather and geolocation APIs enrich the input data, while simulated sensor values are used when live field hardware is unavailable.

ISSN: 3107-6513

III. Advantages of the Proposed System

- Real-time environmental monitoring through weather and location-aware inputs.
- Data-driven prediction for crop yield, pest occurrence, and irrigation needs.
- Better water conservation through smart irrigation recommendations.
- Reduced dependence on manual estimation and traditional guesswork.
- Scalable architecture suitable for future agricultural platform expansion.

III. System Specification

Hardware Requirements

- Processor: Intel Core i5 or above.
- RAM: Minimum 8 GB, recommended 16 GB or above.
- Storage: 500 GB HDD or SSD.
- Network: Internet connection for cloud deployment and API access.

Software Requirements

- Operating System: Windows 10/11, Linux, or macOS.
- Frontend: React.js, HTML, CSS, JavaScript.
- Backend: Python Flask.
- Tools: Jupyter Notebook or Visual Studio Code.
- Libraries: NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn.
- API: OpenWeatherMap API.

IV. Methodology

The system begins with data collection from multiple sources, including user inputs, simulated agricultural sensor datasets, weather APIs, and browser geolocation services. These inputs are validated and prepared before being sent to the backend for analysis.

The backend applies prediction logic through machine learning modules dedicated to three major tasks: crop yield prediction, pest risk classification, and irrigation recommendation. The system uses regression and ensemble-based models for prediction, while fallback rules are used when trained outputs or live data are not available.

The architecture follows a modular design in which frontend components collect input, backend APIs process requests, and results are returned in a structured format. This design improves maintainability, real-time responsiveness, and ease of future enhancement.

V. Module Description

Weather Data Module

This module collects real-time weather parameters such as temperature, humidity, and rainfall using external APIs. These values are used as important features for agricultural predictions.

Location Detection Module

The system identifies the user's current location through browser geolocation services and uses it to provide region-specific recommendations. This improves the practical relevance of the output.

ISSN: 3107-6513

Prediction Module

This module processes environmental and agricultural features through machine learning models to generate crop yield and pest-risk outputs. It acts as the analytical core of the platform.

Irrigation Recommendation Module

The irrigation module estimates suitable watering amounts using soil moisture and weather conditions. It supports efficient water management and sustainable farming.

Data Simulation Module

When physical sensors are unavailable, this module uses CSV-based datasets to simulate IoT sensor behavior. This allows the project to remain functional during prototype and testing stages.

VI. System Design

The file design follows a modular structure separating frontend, backend, and data resources. React handles interface components, Flask manages API routing and model execution, and CSV files are used as a lightweight data source for sensor simulation.

Input design focuses on collecting soil moisture, NPK values, rainfall, temperature, humidity, and location data. Output design presents crop yield prediction, pest alerts, and irrigation recommendations in a simple and readable way for users with limited technical expertise.

The API architecture connects the frontend, backend, and external services into a single data flow. This integration enables real-time request handling, prediction processing, and result delivery.

VII. Data Flow and ER Design

The data flow begins when the user requests crop-related analysis. The system then gathers weather and sensor-related inputs, processes them through prediction models, and stores or returns the results through a prediction module.

The ER structure links weather data, sensor data, and predictions through identified entities and relationships. This design supports organized storage, better retrieval, and clear interaction among modules.

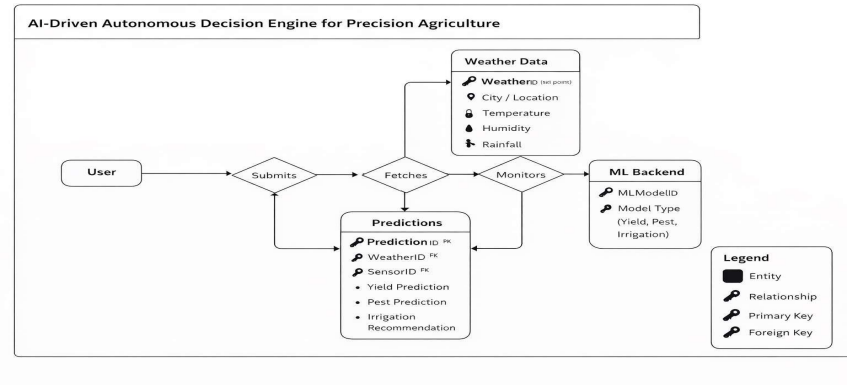

Figure 1. Data Flow Diagram of the proposed precision agriculture system

.

ISSN: 3107-6513

VIII. Testing and Implementation

The project uses Visual Studio Code as the development and testing environment. Testing includes unit testing for individual API endpoints, integration testing for communication between system modules, and system testing for overall performance and reliability.

The implemented architecture combines a React frontend with Flask REST APIs. Example backend endpoints include prediction services for yield, pest analysis, and irrigation recommendation, with data exchanged in structured JSON format.

IX. Results and Discussion

The developed system demonstrates how AI and real-time data integration can improve agricultural decision-making. It supports crop planning, pest monitoring, and water-use optimization by combining prediction modules with environmental inputs.

The project also shows the value of a modular and practical architecture for prototype smart farming systems. Even when real-time sensors are absent, the use of simulated datasets and fallback heuristics helps preserve system continuity and usability.

Table I. Model Performance Metrics

Task	Accuracy	Precision	Recall	Score	Support
Crop	98%	0.98	0.98	0.98	600
Fertilizer	34%	0.34	0.34	0.33	600
Seed	21%	0.22	0.21	0.20	600

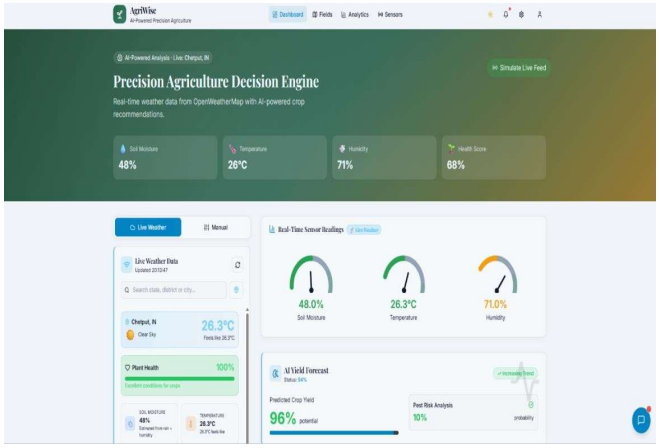

Figure 2. Abstract page from the uploaded final project report.

ISSN: **3107-6513**

X. Conclusion and Future Enhancement

The project successfully demonstrates an AI-enabled decision-support platform for precision agriculture. By integrating machine learning, live weather support, geolocation, and simulated sensing, it provides a structured way to improve productivity, reduce water wastage, and support sustainable farming practices.

Future enhancement can include live IoT sensor integration, a mobile application, multilingual support, cloud deployment, advanced deep learning models, satellite data integration, and support for multiple crop and soil conditions. These additions would improve scalability, prediction quality, and field applicability.

REFERENCES

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.

A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," Computers and Electronics in Agriculture, vol. 147, pp. 70-90, 2018.

M. S. Mekala and P. Viswanathan, "A survey: Smart agriculture IoT with cloud computing," in Proc. Int. Conf. Microelectronic Devices, Circuits and Systems, 2017, pp. 1-7.

S. Pudumalar, E. Ramanujam, R. H. Rajashree, C. Kavya, T. Kiruthika, and J. Nisha, "Crop recommendation system for precision agriculture," in Proc. IEEE Int. Conf. Advanced Computing, 2017, pp. 32-36.

F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825-2830, 2011.

R. Patel, M. Patel, and D. Patel, "Multi-output crop and fertilizer recommendation system using machine learning," Int. J. Eng. Res. Technol., vol. 9, no. 5, 2020.