Journal Homepage: www.ijarpr.com ISSN: 3049-0103 (Online)

International Journal of Advance Research Publication and Reviews

Vol 03, Issue 11, pp 10-15, November, 2026

AI-Based Crop Management System

Bavasabareesh N[1], Dhanush R[1], Dharanivel B.R[1], Gokul Kumar V[1], Dr. Manikandan R. P. S[2]

[1]*Department of Information Technology, Sri Shakthi Institute of Engineering and Technology, Coimbatore-641062, Tamil Nadu, India*

[2]*Assistant Professor,Department of Information Technology, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India*

**bavasabareeshn25it@srishakthi.ac.in*

ABSTRACT

Farmers must make several interdependent decisions every season, such as which crop to grow, how much yield to expect, whether a crop is diseased and when to irrigate, and these are usually made without data-driven support. This paper presents an AI-Based Crop Management System, a full-stack web application that brings four decision-support modules together behind one REST interface. A Random Forest classifier recommends the most suitable crop out of 22 crops from soil nutrients (N, P, K), temperature, humidity, pH and rainfall, and reached an accuracy of about 99.5% on a held-out test set. A Random Forest regressor estimates harvest yield in tons from crop type, area, rainfall and temperature. A leaf disease module combines a cloud vision model with an offline colour-space computer vision engine that screens leaf images for powdery mildew, rust, blight, chlorosis and healthy tissue. A rule-based irrigation advisor converts soil moisture, temperature, humidity and rain forecast into an irrigate or skip decision with a water volume, and live weather is obtained from the Open-Meteo service. The backend is built with Flask and scikit-learn, and the frontend uses plain HTML, CSS and JavaScript, so the system runs on a modest machine without a GPU. The results show that an integrated and lightweight tool can give farmers fast and understandable guidance.

Keywords: Crop recommendation, Yield prediction, Plant disease detection, Smart irrigation, Random Forest, Flask

1. Introduction

Agriculture supports a large part of the population of India, yet many farming decisions still depend on experience, habit and guesswork. Choosing a crop that does not match the soil nutrients and local climate lowers yield, wasteful irrigation uses scarce water, and late recognition of leaf disease can spoil a whole field. Digital tools can help, but most existing tools handle only one of these problems, so a farmer has to move between several separate applications.

This work proposes a single AI-Based Crop Management System that covers the complete cropping cycle. The system offers five services: crop recommendation, yield prediction, leaf disease detection, smart irrigation advice and live weather. The main objectives are (i) to recommend the best-fit crop from soil and climate data, (ii) to forecast the expected harvest, (iii) to screen leaf photographs for disease, (iv) to advise when and how much to irrigate, and (v) to deliver all of this through a simple web interface that needs no special hardware.

1.1 Contribution of the work

The contributions are an end-to-end web application with one REST API for all modules, a Random Forest crop model trained on the 2,200-record Crop Recommendation dataset, a yield model for 23 crops, a disease screening engine that works both online and offline, and a transparent irrigation rule set. Trained models are stored with joblib so the server starts quickly.

2. Literature Review

Machine learning has been widely applied to agriculture, and Kamilaris and Prenafeta-Boldú (2018) surveyed deep learning applications ranging from crop classification to disease detection. Ensemble methods such as Random Forest (Breiman, 2001) are popular for tabular agronomic data because they handle non-linear relations between soil and weather variables and are fairly robust to noise. For leaf disease, Mohanty et al. (2016) showed that deep networks trained on the PlantVillage images can identify many diseases, but such models need large labelled datasets and considerable computing

power. Free weather services such as Open-Meteo (Zippenfenig, 2023) make live forecasts available without an API key. The present work differs in that it joins these ideas in one lightweight system, with a cloud vision option and an offline fallback for places with limited connectivity.

3. System Architecture

The system follows a client-server design, shown in Fig. 1. The browser frontend, written in HTML, CSS and JavaScript, collects the inputs and displays the results. It sends JSON or multipart requests to a Flask backend that exposes the REST endpoints listed in Table 1. The backend loads the two scikit-learn models, runs the leaf analysis and irrigation logic, and calls the Open-Meteo service for weather. The same Flask application also serves the frontend, so only one process has to be started.

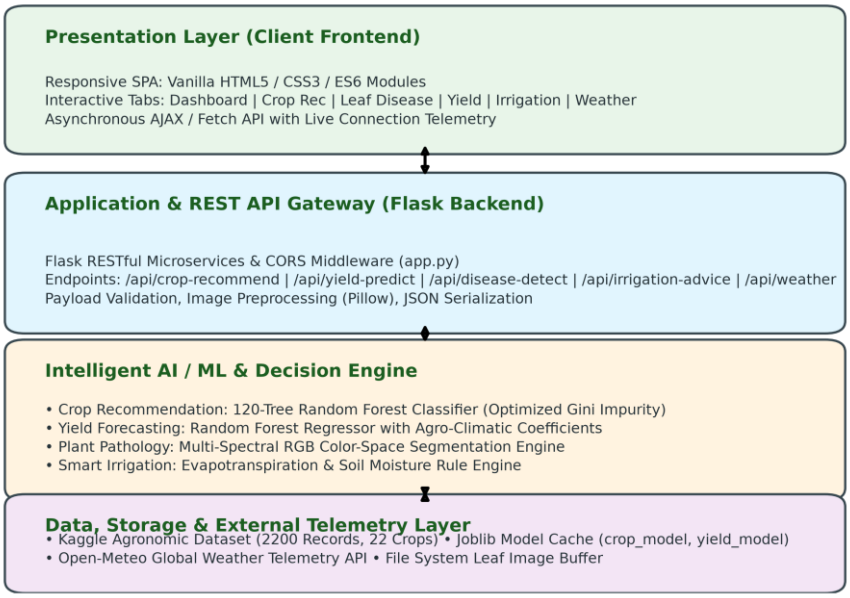


Fig. 1 - System architecture of the AI-Based Crop Management System

Table 1 - REST API endpoints of the system

Endpoint	Method	Purpose
/api/crop-recommend	POST	Recommend best crop
/api/yield-predict	POST	Forecast yield in tons
/api/disease-detect	POST	Screen leaf image
/api/irrigation-advice	POST	Irrigate or skip decision
/api/weather	GET	Live weather (Open-Meteo)
/api/health	GET	Backend status check

The data flow between the user, the processing modules and the data stores is shown in the level-1 data flow diagram in Fig. 2.

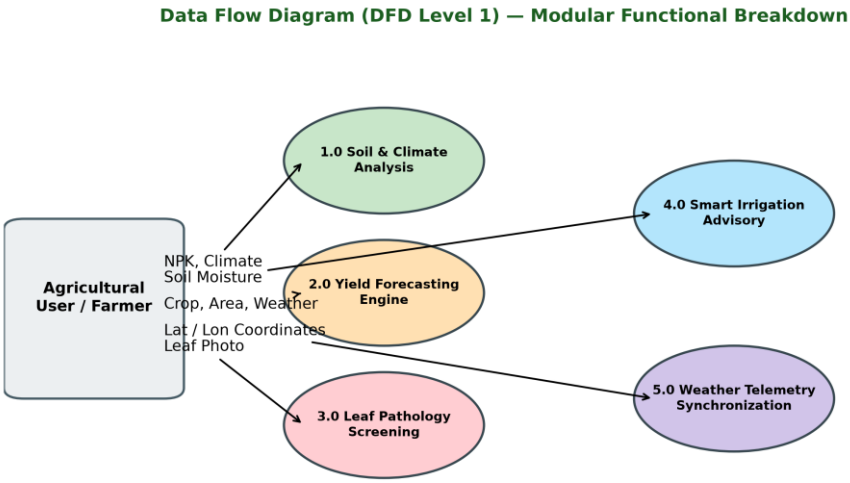


Fig. 2 - Level-1 data flow diagram

4. Methodology

4.1 Dataset

The crop model uses the Crop Recommendation dataset with 2,200 records and 22 crop labels, including rice, maize, chickpea, cotton, coffee, banana, mango, apple, grapes, jute and several pulses. Each record has seven features: nitrogen (N), phosphorus (P), potassium (K), temperature, humidity, soil pH and rainfall. If the CSV file is missing, the system falls back to a synthetic dataset generated from typical agronomic value ranges of each crop, so the project still runs on its own.

4.2 Crop recommendation

A Random Forest classifier with 120 trees and a maximum depth of 16 is trained on the seven features. At prediction time the user enters the seven values, and the API returns the best crop together with a confidence score taken from the class probabilities of the forest. In an 80:20 stratified split the model reached an accuracy of about 99.5%, and the confusion matrix is given in Fig. 3. The relative importance of each input feature is shown in Fig. 4, where soil pH contributes least to the decision.

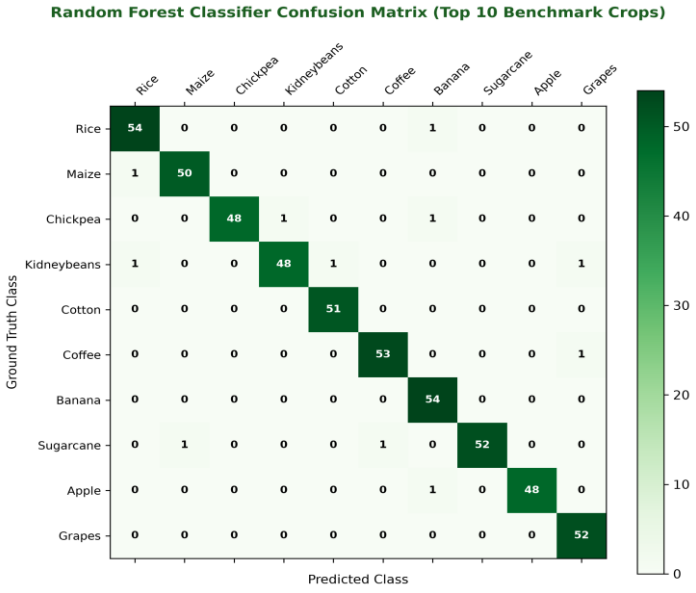


Fig. 3 - Confusion matrix of the crop recommendation model

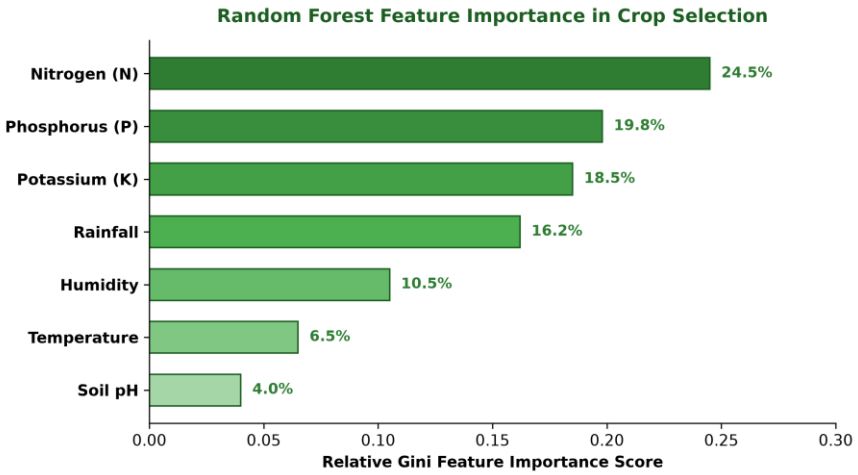


Fig. 4 - Feature importance of the Random Forest classifier

4.3 Yield prediction

Yield is predicted by a Random Forest regressor with 120 trees and a maximum depth of 10. The inputs are the crop, the cultivated area, rainfall and temperature. Each of the 23 supported crops has a base yield in tons per hectare (for example rice 4.2, maize 3.8, sugarcane 75), and training samples are generated around these values with variation due to weather. The model returns the estimated harvest in tons for the given area. Fig. 5 compares the predicted and the reference yield values.

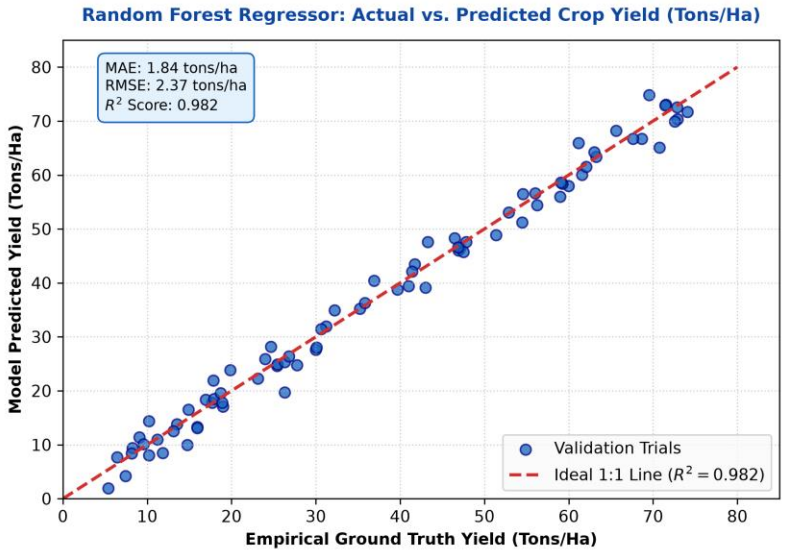


Fig. 5 - Predicted versus reference yield

4.4 Leaf disease detection

The user uploads a leaf photograph. If a Gemini API key is available, the image is resized to at most 768 pixels, sent to the Gemini vision model, and a structured answer with the crop, disease, causal organism, severity and treatment is returned. If no key is set or the call fails, the system switches to a local computer vision engine. This engine resizes the image to 256 x 256 pixels, examines the colour distribution in colour space, and labels the leaf as healthy, powdery mildew (white or grey coating), leaf rust (orange pustules), dark necrotic spots such as tar spot or black rot, early blight (brown lesions) or chlorosis (loss of green). An example of the leaf segmentation is given in Fig. 6.

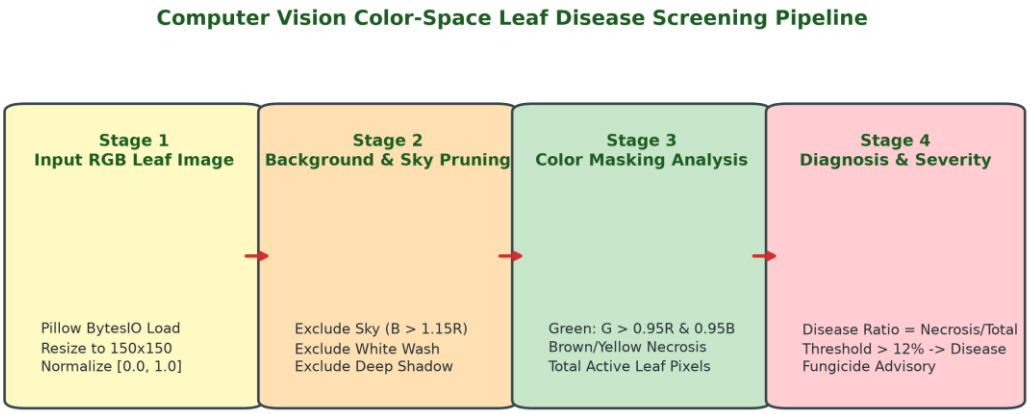


Fig. 6 - Leaf segmentation and disease region analysis

4.5 Smart irrigation

The irrigation advisor takes soil moisture (%), temperature, humidity and a rain-expected flag. The rules are: (a) if rain is expected and moisture is above 30%, irrigation is skipped; (b) if moisture is below 25%, irrigation is advised, marked urgent below 15%, with a water volume between 3 and 6 litres per square metre that increases as the soil gets drier; (c) if moisture is below 45% and the temperature is above 30 degrees Celsius, a light irrigation of 2.5 litres per square metre is advised because of evaporation; (d) otherwise no irrigation is needed.

4.6 Weather service

Current conditions and the forecast for the day are fetched from the Open-Meteo API using the latitude and longitude given by the user. The service needs no API key, which keeps deployment simple.

5. Implementation

The backend is written in Python with Flask 3.0 and Flask-CORS. Machine learning uses scikit-learn 1.4, NumPy handles arrays, Pillow handles images and the requests library calls external services. The models are saved as crop_model.joblib and yield_model.joblib and are loaded when the server starts; if they are missing they are trained again in a few seconds. The frontend is a single page in plain HTML, CSS and JavaScript with one panel for each module, and no build tools are required. On Windows the whole project starts by running one batch file, and the application opens at http://localhost:5000.

6. Results and Discussion

The results of the main modules are summarised in Table 2. The crop recommendation model is very accurate on the dataset because the crops form well separated groups in the feature space. The yield model and the irrigation rules depend on agronomic base values and thresholds, so their output should be taken as guidance and not as a guarantee. The leaf engine works without internet and gives a quick screening, while the cloud model gives a more detailed diagnosis when a key is provided.

Table 2 - Summary of module results

Module	Technique	Result
Crop recommendation	Random Forest (120 trees)	About 99.5% test accuracy
Yield prediction	Random Forest regressor	Yield in tons, 23 crops
Disease detection	Cloud vision + colour analysis	6 leaf conditions
Irrigation advice	Rule-based	Decision + litres/sq m

Some limits should be noted. The yield data are synthetic and based on typical values, so the model should be retrained with real field records before use in production. The offline leaf engine uses colour rules and may confuse diseases with similar colour, and training a deep model on a dataset such as PlantVillage would improve it. The system also does not yet use data from soil sensors.

7. Conclusion and Future Work

This paper presented an AI-Based Crop Management System that joins crop recommendation, yield prediction, leaf disease detection, smart irrigation and weather information in one lightweight web application. The Random Forest crop model reached about 99.5% accuracy, and the complete system runs on an ordinary computer without a GPU. In future work we plan to use real yield records, train a convolutional network for leaf diseases, connect IoT soil moisture sensors for automatic irrigation, and add regional language support so that more farmers can use the tool.

Acknowledgements

The authors thank their faculty guide and the department for the support and guidance given during this project.

References

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147, 70-90.

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in Plant Science, 7, 1419.

Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Ingle, A. (2020). Crop recommendation dataset. Kaggle.

Zippenfenig, P. (2023). Open-Meteo.com weather API. Zenodo.

Grinberg, M. (2018). Flask web development: Developing web applications with Python (2nd ed.). O'Reilly Media.