The Plant Health Monitoring Web Application using Machine Learning

Sonali Chandrakant More
Komal Gharat
Computer Science, CKT ACS College, New Panvel (Empowered Autonomous) Affiliated to University of Mumbai
Email: sonalimore0303@gmail.com
Email: komalgharat07@gmail.com

How to Cite this Article:

More, S. C. & Gharat, K. (2026). The Plant Health Monitoring Web Application using Machine Learning. International Journal of Creative and Open Research in Engineering and Management, 02(04). https://doi.org/10.55041/ijcope.v2i4.476

https://doi.org/10.55041/ijcope.v2i4.476

Abstract — Plant diseases and unfavorable environmental conditions pose significant challenges to agricultural productivity, often remaining undetected until severe damage occurs. This paper presents a full-stack web application designed to monitor plant health and provide intelligent crop recommendations based on environmental conditions such as temperature, humidity, sunlight, and watering frequency. The system utilizes a Python-based machine learning backend that trains and evaluates three supervised classification models: Decision Tree, Random Forest, and Logistic Regression. The best-performing model is selected and deployed for real-time prediction through a REST API. The frontend is implemented as a responsive multi-page web interface that enables users to perform plant recommendation, health prediction, and leaf image-based disease detection. Experimental results demonstrate that the Decision Tree model achieves the highest accuracy, making it suitable for deployment. The system offers a practical, accessible, and efficient solution for plant health monitoring and agricultural decision support.

Keywords — Plant Disease Detection, Machine Learning, Decision Tree, Random Forest, Logistic Regression, Plant Health Monitoring, Crop Recommendation, Web Application, REST API, Data Preprocessing

I. INTRODUCTION

Plant health plays a vital role in maintaining agricultural productivity and ensuring global food security. Recent advancements in machine learning and deep learning have enabled the development of intelligent systems capable of accurately detecting and classifying plant diseases using structured and image-based data [1].

The application of these technologies in agriculture has gained significant attention, as they provide automated and scalable solutions for crop monitoring and disease management [2]. Studies have shown that plant diseases caused by fungi, bacteria, viruses, and environmental stress conditions can lead to severe crop losses, sometimes exceeding 40% in highly affected regions, making early detection essential [3], [4].

Comparative research on different deep learning models has demonstrated their effectiveness in identifying plant diseases with high accuracy, highlighting the potential of these approaches in real-world agricultural applications [5]. Machine learning libraries such as Scikit-learn further support the development of such systems by providing efficient tools for data preprocessing, model training, and evaluation [6].

To train and evaluate these models, publicly available datasets such as the PlantVillage dataset have been widely used, offering a large collection of labeled plant images for disease detection tasks [7]. Additionally, modern machine learning frameworks and practical implementations have simplified the process of building intelligent applications for real-world use [8].

Building on these advancements, the proposed system is a full-stack web application that provides plant recommendation and health prediction based on environmental inputs, along with a vision-based module for leaf image analysis. Similar approaches using convolutional neural networks have shown strong performance in plant disease recognition tasks [9]

II. RELATED WORK

Mohanty *et al.* [1] demonstrated that convolutional neural networks trained on the PlantVillage dataset can achieve over 99% accuracy in plant disease identification under controlled conditions, establishing the effectiveness of automated plant diagnosis.

Kamilaris and Prenafeta-Boldú [2] surveyed multiple deep learning applications in agriculture and concluded that these approaches outperform traditional methods in image classification tasks, although they require large labeled datasets and significant computational resources.

Ferentinos [3] evaluated various deep learning architectures and reported accuracy rates ranging from 31.4% to 98.1%, highlighting the importance of model selection and training data quality. Barbedo [4] reviewed classical image processing techniques such as color, texture, and shape-based feature extraction, which support alternative approaches using structured data.

Too *et al.* [5] compared fine-tuned deep learning models and found that deeper architectures provide better generalization in real-world conditions but require high computational power. For practical and lightweight implementations, classical machine learning approaches using tools like Scikit-learn offer efficient and interpretable solutions [6].

The PlantVillage dataset [7] has played a key role in training and evaluating plant disease detection models by providing a large collection of labeled plant images. Géron [8] further supports the use of practical machine learning frameworks for building real-world applications.

Li *et al.* [9] proposed convolutional neural network-based methods for plant disease recognition and achieved strong classification performance, demonstrating the effectiveness of deep learning techniques. However, most existing systems focus on a single task, whereas this system integrates both plant recommendation and health prediction into a unified, locally executed system.

III. SYSTEM DESIGN

A. Overall Architecture

The system follows a client-server architecture. The frontend is a multi-page web application that the user interacts with through a browser. The backend is a Python server that handles all machine learning operations and exposes prediction endpoints through a REST API. The frontend communicates with the backend by sending HTTP requests and displaying the returned results to the user.

All machine learning computation — including data preprocessing, model training, and real-time prediction — takes place on the backend server. No machine learning logic runs in the browser. This separation keeps the frontend lightweight and ensures that the ML pipeline can be updated independently of the user interface.

B. Application Pages

The web application provides six pages accessible through a top navigation bar:

1) Home — Introduces the project, describes the three core features (Leaf Vision, Health Risk Models, Diagnostic Analytics), and lists the four supported crop categories with a call-to-action to train a local model.

2) Recommend — Accepts user inputs for city location, temperature, humidity, sunlight level, and watering frequency. Sends these to the backend and displays the top three recommended plant species ranked by predicted suitability,

along with match confidence scores and care tips.

3) Health — Accepts the same environmental parameters and runs them through the health prediction model. Displays the predicted crop, health status (Healthy or Infected), confidence percentage, and a breakdown of which features most influenced the prediction.

4) Vision — Allows the user to upload a photograph of a plant leaf. The image is sent to the backend for disease classification. The result includes a disease label, confidence score, and a list of maintenance and treatment suggestions.

5) Training — Allows the user to upload a CSV dataset and trigger model training. The backend trains all three classifiers, compares their performance, saves the best model, and returns the accuracy and F1 score to the frontend for display.

6) Dashboard — Shows a summary of all past diagnoses. Includes total inference count, currently active model, a bar chart of healthy versus infected results.

IV. DATASET AND PREPROCESSING

A. Dataset

The system is trained on a structured CSV dataset containing records for multiple plant species and health conditions. Each record captures environmental and cultivation parameters alongside a class label. The dataset covers four crop categories: Tomato, Potato, Pepper Bell, and Corn, spanning both healthy and diseased states including Bacterial Spot, Early Blight, Late Blight, and Common Rust.

Table I shows the structure of the dataset used for training:

Table I: Dataset Feature Structure

Feature	Type	Description	Example Values
temperature	Numerical	Ambient temperature in Celsius	18, 22, 28, 30
humidity	Numerical	Relative humidity in percentage	45, 55, 65, 80
sunlight	Categorical	Sunlight intensity level	Low Indirect, Medium, High Direct
water	Categorical	Watering frequency	Low, Medium, Frequent
label	Categorical	Target class: plant / health state	Tomato Healthy, Potato Blight

B. Preprocessing Steps

Before training, the dataset goes through the following preprocessing steps:

- Missing value handling: Numerical columns use median imputation. Categorical columns use the most frequent value.

- Categorical encoding: The sunlight and water columns are converted from text labels tonumbers using Label Encoding so the classifiers can process them.

- Normalisation: Temperature and humidity values are scaled to the range [0, 1] using Min-Max Scaling to ensure no single feature dominates due to its magnitude.

- Train-test split: The data is divided into 80% for training and 20% for testing, using stratified sampling to maintain class balance in both sets.

The same preprocessing steps are applied consistently during both training and prediction to ensure accurate and fair results.

V. MACHINE LEARNING

A. Models Used

Three supervised classification algorithms were trained and compared:

1) Decision Tree: A Decision Tree splits the data into branches based on feature values to reach a prediction. It is easy to interpret, handles both numerical and categorical features well, and can model complex non-linear patterns. A maximum depth limit was applied to prevent overfitting.

2) Random Forest: A Random Forest builds 100 Decision Trees, each trained on a random subset of the data and features. The final prediction is made by majority vote across all trees. This ensemble approach reduces the risk of overfitting and generally produces more stable predictions than a single tree.

3) Logistic Regression: Logistic Regression predicts class probabilities using a mathematical function and selects the most likely class. It serves as a simple and interpretable baseline. L2 regularisation was applied to avoid overfitting.

B. Model Evaluation

Each model was evaluated on the 20% test set using four standard metrics: Accuracy, Precision, Recall, and F1 Score. Table II shows the results:

Table II: Model Performance Comparison

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Decision Tree	91.7	91.4	91.7	91.5
Random Forest	89.3	89.1	89.3	89.2
Logistic Regression	78.6	78.2	78.6	78.1

The Decision Tree achieved the best accuracy of 91.7% and F1 Score of 91.5%, making it the selected model for deployment. The Random Forest performed close behind at 89.3%, while Logistic Regression scored 78.6% due to the non-linear nature of the plant health data. The winning model is saved to disk and loaded automatically for real- time predictions.

C. Feature Importance

Analysis of the trained Decision Tree revealed that humidity has the highest influence on predictions at approximately 40%, followed by temperature at 27%, watering frequency at 20%, and sunlight intensity at 13%. This is consistent with known plant biology: moisture and temperature are the primary drivers of most fungal and bacterial plant diseases.

VI. IMPLEMENTATION

A. Backend

The backend is written in Python and handles all machine learning and API functionality. It exposes three main endpoints:

POST /train: Accepts a CSV dataset uploaded by the user. Runs the full preprocessing and training pipeline for all three classifiers, evaluates each one, saves the best model to disk, and returns the performance metrics to the frontend.

POST /predict: Accepts environmental parameters (temperature, humidity, sunlight, water) as input. Loads the saved model, preprocesses the input in the same way as training, and returns the top three predicted plant matches with confidence scores and care tips.

POST /predict-image: Accepts a leaf image file uploaded by the user. Processes the image and returns a disease classification label, confidence score, and treatment recommendations.

All incoming data is validated before processing. Errors are returned as clear JSON messages with appropriate status codes.

B. Frontend

The frontend is a responsive multi-page web application built with modern web technologies. It provides a clean and intuitive interface for all six pages described in Section III. Forms are validated before submission to catch errors early and give users helpful feedback. Charts on the Dashboard page visualise diagnosis history in an easy-to-read format. The application works on both desktop and mobile browsers.

The frontend sends requests to the backend using standard HTTP calls and displays the results dynamically without requiring a page reload. Loading states and error messages are shown clearly to keep the user informed at every step.

C. System Workflow

The typical user workflow is as follows: the user opens the web application and navigates to the desired page. On the Recommend or Health page, they enter their environmental conditions and click the generate button. The frontend sends this data to the backend, which preprocesses it, runs it through the saved model, and returns a prediction. The result is displayed instantly on the screen with a confidence score and helpful details. On the Vision page, the user uploads a leaf photo and receives a disease diagnosis within seconds. On the Training page, the user can upload a new dataset to retrain and update the model at any time.

VII. RESULTS AND DISCUSSION

A. Training Results

The training pipeline was tested on the plant dataset covering four crop categories. Training completed in under five seconds on a standard laptop. The Decision Tree achieved the highest accuracy of 91.7% and was selected as the active model. The system correctly confirmed that the model was saved and ready for inference after training completed.

B. Recommendation Results

For a test input of 22 degrees Celsius temperature, 50% humidity, Low Indirect sunlight, and Daily watering, the recommendation engine correctly ranked Tomato as the top match at 60% confidence and Potato as second at 50% confidence. The engine also correctly identified relevant care tips such as deep watering preference and high initial nitrogen needs for the top-ranked plant.

C. Health Prediction Results

For an input of 22 degrees Celsius, 45% humidity, Medium sunlight, and Medium watering, the health prediction module correctly classified the crop as Tomato with a Healthy status at 43% confidence. Feature attribution identified humidity (40%), temperature (27%), watering frequency (20%), and sunlight (13%) as the contributing factors.

D. Vision Module Results

A Pepper Bell leaf image with surface spotting was submitted to the Vision module and correctly classified as **Pepper Bell Bacterial Spot** at 97% confidence, with appropriate care recommendations returned

E. Dashboard Results

The Dashboard correctly displayed a total of 34 inference sessions during testing, with 32 classified as Healthy and 2 as Infected. The bar chart accurately visualised this distribution and the history table correctly listed all sessions with their dates, crop names, and health states. Filtering and sorting worked as expected.

F. System Performance

The backend responded to prediction requests in under 200 milliseconds for parametric inputs and under 800 milliseconds for image uploads. The frontend loaded quickly on both desktop and mobile browsers, and all pages responded smoothly to user interactions without noticeable delays.

VIII. CONCLUSION

This paper presented a full-stack web application for plant health monitoring and intelligent crop recommendation. The system combines a Python machine learning backend with a responsive multi-page web frontend to deliver practical agricultural intelligence to everyday users without requiring any specialist knowledge.

The comparative evaluation showed that the Decision Tree classifier achieved the best performance at 91.7% accuracy and 91.5% F1 Score on the plant health dataset, outperforming Random Forest and Logistic Regression. Feature analysis confirmed that humidity and temperature are the dominant factors in plant health classification, which aligns with established agricultural science.

All six application pages — Home, Recommend, Health, Vision, Training, and Dashboard — were validated and found to work correctly, providing a complete end-to-end experience from environmental input to actionable plant health insights. The system runs entirely on local infrastructure with no dependency on any external AI service.

Future work will focus on expanding the dataset to cover more crop species and disease categories, adding real-time weather data integration to auto-fill environmental parameters, and exploring more advanced ensemble methods to further improve prediction accuracy. A mobile application version is also planned to improve accessibility for field-based farmers.

REFERENCES

[1] S. Mohanty, D. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," Frontiers in Plant Science, vol. 7, pp. 1419, 2016.

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

[3] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," Computers and Electronics in Agriculture, vol. 145, pp. 311–318, 2018.

[4] M. Barbedo, "Digital image processing techniques for detecting, quantifying, and classifying plant diseases," SpringerPlus, vol. 2, pp. 660, 2013.

[5] P. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuned deep learning models for plant disease identification," Computers and Electronics in Agriculture, vol. 161, pp. 272–279, 2019.

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

[7] A. Dev, "PlantVillage Dataset," *Kaggle*, 2020. Available: https://www.kaggle.com/datasets/abdallahalidev/plantvill age-dataset

[8] A. Géron, Hands-On Machine Learning with Scikit- Learn, Keras, and TensorFlow, 2nd ed. O'Reilly Media, 2019.

[9] J. Li, X. Chao, and Y. Chen, "Plant disease recognition using deep convolutional neural networks," IEEE Access, vol. 7, pp. 175578–175588, 2019.

[10] H. Brahimi, A. Boukhalfa, and A. Moussaoui, "Deep learning for tomato diseases: Classification and symptoms visualisation," Applied Artificial Intelligence, vol. 31, no. 4, pp. 299–315, 2017.