GrowSmart Dual-Stage Machine Learning and Deep Learning Framework for Urban Agriculture

A Project Report submitted in the partial fulfillment
of the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Patalam Nayeem (22471A05B5)
Addaki VarunYadav (22471A0571)
Kuppala Purna Chandra Rao (22471A05A5)

Under the esteemed guidance of

Ch Rajani, M.Tech

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET

(AUTONOMOUS)

Accredited by NAAC with A+ Grade and NBA under

Tyre -1 an ISO 9001:2015 Certified

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada

KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601

2025-2026

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name GrowSmart Dual-Stage Machine Learning and Deep Learning Framework for Urban Agriculture **is a bonafide work done by the team Patalam Nayeem (22471A05B5), Addaki VarunYadav (22471A0571), Kuppala Purna Chandra Rao (22471A05A5)** BACHELOR OF TECHNOLOGY in the Department of **COMPUTER SCIENCE AND ENGINEERING** during **2025-2026.**

PROJECT GUIDE
Ch Rajani, M.Tech
Assistant Professor

PROJECT CO-ORDINATOR
Dr. Sireesha Moturi, B.Tech., M.Tech., Ph.D
Associate Professor

HEAD OFTHE DEPARTMENT
Dr. S. N. Tirumala Rao, M.Tech., Ph.D.
Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled “GROWSMART DUAL-STAGE MACHINE LEARNING AND DEEP LEARNING FRAMEWORK FOR URBAN AGRICULTURE” is composed by ourselves that the work contain here is our own except where explicitly stated otherwise in the text and that this work has been not submitted for any other degree or professional qualification except as specified.

Patalam Nayeem	(22471A05B5)
Addanki Varun Yadav	(22471A0571)
Kuppala Purna Chandra Rao	(22471A05A5)

ACKNOWLEDGEMENT

We wish to express my thanks to carious personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman sri **M. V. Koteswara Rao**, **B.Sc.**, who took keen interest in us in every effort throughout thiscourse. We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu, Ph.D.**, for showing his kind attention and valuable guidance throughout the course.

We express our deep felt gratitude towards **Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,** HOD of CSE department and also to our guide Ch Rajani **, M.Tech.**, of Assistant Professor whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **Dr. Sireesha Moturi, B.Tech, M.Tech.,Ph.D.,** Associate professor & Project coordinator of the project for extending her encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff to department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

By

Patalam Nayeem (22471A05B5)

Addanki Varun Yadav (22471A0571)

Kuppala Purna Chandra Rao (22471A05A5)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community,

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a centre of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertize in modern software tools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

Program Outcomes

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science,computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

PO3: Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Project Course Outcomes (CO'S):

CO421.1: Analyse the Systemof Examinations and identify the problem.

CO421.2: Identify and classify the requirements.

CO421.3: Review the Related Literature

CO421.4: Design and Modularize the project

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Preparethe project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
C421.1		✓										✓		
C421.2	✓		✓		✓							✓		
C421.3				✓		✓	✓	✓				✓		
C421.4			✓			✓	✓	✓				✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓	✓	✓	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
C421.1	2	3										2		
C421.2			2		3							2		
C421.3				2		2	3	3				2		
C421.4			2			1	1	2				3	2	
C421.5					3	3	3	2	3	2	2	3	2	1
C421.6									3	2	1	2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applie in this project	Description of the device	Attained PO
C2204.2, C22L3.2	Gathering the requirements and defining the problem, plan to develop model for detection and classification of OSCC	PO1, PO3,PO8
CC421.1, C2204.3, C22L3.2	Each and every requirement i critically analyzed, the process mode is identified	PO2, PO3, PO8
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9, PO8
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5, PO8
CC421.4, C2204.4, C22L3.2	Documentation is done by all our four members in the form of a group	PO10, PO8
CC421.5, C2204.2, C22L3.3	Each and every phase ofthe work in group is presented periodically	PO8,PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done and the project will be handled by the social media users and in future updates in our project can be done based on detection for Oral Cancer	PO4, PO7, PO8
C32SC4.3	The physical design includes website to check OSCC	PO5, PO6, PO8

ABSTRACT

In recent years, Artificial Intelligence has become a transformative tool in the field of smart and sustainable agriculture. The present work illustrates a novel dual-stage framework, **GrowSmart**, designed for accurate crop prediction, fertilizer recommendation, and plant disease detection using a combination of **Machine Learning (ML)** and **Deep Learning (DL)** techniques. The proposed model integrates the **Random Forest** algorithm for environmental data-based crop prediction and fertilizer suggestion, while a **ResNet9 Convolutional Neural Network (CNN)** is employed for leaf disease classification with high precision. Initially, essential preprocessing techniques such as **data normalization using StandardScaler** and **dataset cleaning** are applied to enhance model performance and consistency. The ML module analyzes key soil and climatic parameters—Nitrogen (N), Phosphorus (P), Potassium (K), pH, temperature, humidity, and rainfall—to determine the most suitable crop and fertilizer. Simultaneously, the DL module processes leaf images through a convolutional architecture to detect diseases accurately. The integrated **Flask-based web application** allows real-time user interaction and visualization of predictions. The proposed GrowSmart model demonstrated high accuracy in both crop recommendation and disease classification, proving its effectiveness as an intelligent decision-support system for **urban agriculture and precision farming**. Hence, this hybrid ML–DL framework offers an automated, reliable, and scalable solution to support sustainable agricultural development

INDEX

LIST OF FIGURE

S. No.	List of Figures	Page No.

1. INTRODUCTION

Rapid urbanization, population growth, and changing dietary patterns are putting intense pressure on global food systems, especially in cities where land and resources are limited. Recent studies on urban agriculture emphasize that city regions are becoming both major consumption hubs and increasingly important production zones, as governments and planners look for resilient ways to ensure food and nutritional security under space, water, and climate constraints [1], [2]. Urban agriculture is no longer seen as a supplementary activity; it is now framed as a strategic pathway for building resilience, supporting local economies, and improving environmental quality in dense metropolitan regions [1], [4].

At the same time, climate variability, erratic rainfall, and degraded soils are making conventional trial-and-error farming approaches unsustainable. Urban growers—often operating on rooftops, balconies, community plots, and peri-urban fringes—must make precise decisions about which crops to grow, how much fertilizer to apply, and how to respond quickly to plant diseases that can wipe out small but critical harvests [2], [3], [11]. In these constrained environments, a wrong choice of crop or fertilizer does not only reduce yield; it can make the entire micro-farm economically unviable. This challenge has led to the growing adoption of data-driven decision support systems in agriculture [4], [12], [14]. Machine Learning (ML) and Deep Learning (DL) technologies are now widely recognized as key enablers of "smart farming" and precision agriculture. ML models have been successfully used to recommend suitable crops and fertilizer dosages based on soil macronutrients (N, P, K), pH, and local environmental factors such as temperature, humidity, and rainfall [5]–[9], [11], [13]. These systems can capture non-linear relationships between soil profiles, climate conditions, and crop performance, providing tailored recommendations that improve yield and resource efficiency for both smallholder and urban farmers [6], [7], [10], [12]. Parallel research in DL has shown strong results in plant disease detection using leaf images, where convolutional neural networks (CNNs) and related architectures achieve high classification accuracy across multiple crop species and disease types [15]–[17].

Beyond isolated use cases, there is a clear trend towards integrated, end-to-end frameworks that combine multiple ML and DL components into unified decision-support pipelines. Reports on AI-enabled food systems highlight that the future of urban agriculture lies in tightly coupled systems that can recommend crops and fertilizers, monitor plant health, and

provide early warnings of stress or disease, all within a single workflow [4], [12], [14], [15]. IoT-enabled sensors for soil and microclimate monitoring, coupled with edge-deployable ML/DL models, allow real-time, low-latency analysis that is suitable for small urban farms where connectivity and hardware resources may be limited [6], [10], [18], [20].

Within this context, the GrowSmart framework is designed as a dual-stage system that directly addresses the core pain points of urban growers. In Stage-1, ensemble ML models recommend the most suitable crops and corresponding fertilizer requirements using soil nutrients, pH, and environmental parameters as inputs [5]–[9], [11], [13]. In Stage-2, a compact ResNet-based DL model performs plant disease classification from leaf images, using the crop context predicted in Stage-1 to narrow down relevant disease categories and improve robustness [15]–[17]. This creates a closed decision loop that links recommendation and monitoring: users are not only told what to grow and how to nourish it, but also continuously informed about the health status of their plants.

By emphasizing interpretability, deployment on mobile and edge devices, and compatibility with low-cost sensors and small datasets, GrowSmart aims to bridge the gap between academic advances in AI-driven agriculture and the practical realities of urban farming. The framework directly aligns with recent research directions in information fusion, explainable models, and domain-specific AI applications in agriculture and healthcare, leveraging methodologies from prior work on time-series analysis, medical signal classification, and explainable imaging models [14], [18]–[20]. In doing so, it contributes to building scalable, accessible, and sustainable digital infrastructure for urban food systems [1], [2], [4], [12].

1.1 Background

Urban agriculture has evolved from small community-based gardening activities into a structured, technology-assisted food production ecosystem capable of supporting growing urban populations. As cities continue expanding at unprecedented rates, traditional agricultural land becomes increasingly scarce, forcing food production systems to adapt to limited spatial, environmental, and resource constraints [1], [2]. The World Food Organization reports that integrating artificial intelligence, automation, and digital monitoring into food systems is now considered essential for achieving sustainability, resilience, and scalability in growing metropolitan regions [4].

The core idea behind smart farming in urban settings is precision: producing more output using fewer resources, while maintaining environmental responsibility and crop health. To achieve this, modern agricultural systems rely on data-driven insights derived from soil quality, atmospheric parameters, crop growth patterns, and nutrient deficiencies [3], [6]. The shift from experience-based farming to analytical decision-making allows even novice urban growers to make informed judgments about crop selection, fertilizer quantities, and irrigation cycles, reducing dependency on trial-and-error practices [5], [8], [10].

Machine learning plays a fundamental role in this transition by enabling intelligent analysis of soil nutrients such as nitrogen, phosphorus, and potassium (N-P-K), pH levels, humidity, temperature, and rainfall patterns [7], [8], [11]. These algorithms can identify subtle relationships between environmental factors and crop performance—patterns that are often too complex for manual evaluation—resulting in improved crop selection and resource optimization [9], [12], [13].

Deep learning further extends this precision by supporting automated disease diagnosis using leaf imagery. CNN-based models have demonstrated capabilities comparable to expert agricultural diagnosticians, detecting early symptoms of fungal, bacterial, and viral infections in crops with high accuracy [15], [16], [17]. Research also indicates that integrating AI-enabled monitoring systems with ML-based recommendation engines leads to smarter, faster, and more adaptive crop management loops, especially in urban micro-environments where variations in conditions can be abrupt and localized [14], [18], [19].

Combined with mobile-friendly AI deployment and IoT-based sensing, these technological advancements enable continuous monitoring, real-time decision-making, and data-supported agricultural planning for small-scale growers. Smart agriculture systems deployed on edge devices such as Raspberry Pi and low-power mobile processors have already shown strong feasibility and adoption potential across emerging smart cities [6], [10], [20].

Thus, the background of this work lies at the intersection of urban food security challenges and AI-driven precision agriculture innovations, forming the foundation for integrated

systems like GrowSmart that unite ML-based recommendation strategies with deep learning-based crop health diagnostics.

1.2 Problem Statement

Urban agriculture presents a promising solution to food scarcity and rising population density; however, effective cultivation in constrained environments remains a complex challenge. Unlike large-scale conventional farms, urban growers operate in limited planting areas such as rooftops, balconies, terraces, and compact soil beds, where mistakes in crop selection, fertilizer dosing, or disease management result in immediate productivity loss and resource waste [1], [2], [4]. These farms have very small error margins, meaning that even minor misjudgments can make cultivation economically nonviable.

Most urban farmers rely on subjective decisions, generic fertilizer guidelines, or incomplete agricultural knowledge. Such uninformed practices often lead to nutrient imbalance—either overapplication, which damages soil and increases cost, or underapplication, which leads to nutrient deficiencies and low yield [5], [6], [11]. Climatic variability, changing rainfall patterns, and inconsistent environmental conditions further complicate decision-making, especially for novice growers with limited agronomic expertise [3], [7], [12].

Moreover, early detection and management of plant diseases remains a critical issue. Research indicates that leaf-borne infections are one of the primary reasons for reduced crop yield in home-scale and commercial micro-farming setups [15], [16]. Traditional disease assessment methods rely on manual inspection, which is error-prone, slow, and requires expert agricultural knowledge—something most urban growers do not have access to [15], [17]. By the time visual symptoms become obvious, disease progression is often irreversible, resulting in wasted time, labor, and resources.

Existing agricultural recommendation systems and disease classification models are typically developed in isolation rather than integrated workflows. This disjointed design forces users to consult multiple tools—one for crop suggestions and another for disease identification—making the process inefficient and technically demanding [8], [14], [18]. In addition, many existing systems require high computational power, preventing deployment

on low-cost handheld devices commonly used by urban farmers and IoT-based monitoring systems [6], [10], [20].

Therefore, there is a clear need for a unified, intelligent, lightweight, and user-friendly framework capable of:

- Recommending suitable crops and optimal fertilizer quantities based on soil and environmental conditions.
- Detecting plant diseases from leaf images with high accuracy.
- Operating efficiently on mobile and edge devices for real-time decision support.
- Addressing these challenges forms the foundation for developing the GrowSmart dual-stage AI-assisted agriculture system, bridging the gap between traditional cultivation limitations and modern intelligent farming solutions.

1.3 Objectives of the Study

The primary objective of this research is to develop an integrated smart farming framework capable of supporting urban growers through automated decision-making, resource optimization, and proactive plant health monitoring. The system aims to bridge the gap between traditional agricultural intuition and modern AI-driven precision farming practices, ensuring reliable yield outcomes even in space-limited environments [1], [2], [4]. To achieve this, the GrowSmart framework has been designed with the following specific objectives:

Objective 1: Develop a Machine Learning-Based Crop Recommendation System
To build a model that analyzes soil nutrients (N, P, K), pH levels, temperature, humidity, and rainfall to determine the most suitable crop for a given environment. The system should identify patterns and dependencies among these parameters using supervised learning and provide optimized crop recommendations for urban farming conditions [5], [7], [11], [13].

Objective 2: Generate Fertilizer Recommendations Based on Soil and Crop Requirements
To incorporate fertilizer prediction into the recommendation pipeline to provide nutrient-specific instructions and maintain soil balance. The goal is to eliminate guesswork, reduce

wastage, and prevent nutrient toxicity or deficiency, improving soil health and crop yield efficiency [6], [8], [10], [12].

Objective 3: Implement a Deep Learning Model for Plant Disease Detection

To train and deploy a ResNet-based deep learning model capable of classifying plant leaf diseases using image-based diagnosis. The model should operate reliably across multiple classes of crop diseases and support real-time decision-making to assist early intervention and minimize spread [15], [16], [17].

Objective 4: Create a Dual-Stage integrated Decision-Support System

To combine crop and fertilizer recommendation (Stage-1) with plant disease detection (Stage-2) into one seamless pipeline. The system should use context-aware logic—where Stage-1 outputs guide Stage-2 disease categories—to improve relevance and diagnostic precision [4], [14], [18].

Objective 5: Ensure Deployment on Mobile and Edge Devices

To optimize the trained models so they can run on low-power computing devices like smartphones or Raspberry Pi, enabling field-level applicability without reliance on high-performance computing infrastructure or internet connectivity [6], [10], [19], [20].

Objective 6: Improve Accessibility, Efficiency, and Decision Transparency

To provide interpretability features such as confusion matrices, heatmaps, and feature importance visualizations—allowing farmers to understand how predictions are made and increasing trust in AI-assisted agriculture solutions [12], [14], [15].

1.4 Scope of the Project

The scope of this project encompasses the design, development, and evaluation of a dual-stage intelligent agriculture support system tailored for urban farming environments. The system integrates machine learning–based crop and fertilizer recommendation with deep learning–based plant disease detection, creating a unified workflow that assists users from planting decisions to real-time health monitoring [1], [4], [12]. The project focuses on enabling precision agriculture within limited-growing spaces such as terrace gardens, vertical farms, and small household cultivation environments where resource optimization

and early disease intervention are critical [2], [3], [6].

The first stage of the system includes data preprocessing, feature engineering, and training of ML models using agricultural parameters including nitrogen (N), phosphorus (P), potassium (K), soil pH, humidity, temperature, and rainfall. The output of this stage provides farmers with tailored crop selection and nutrient recommendations that support efficient and informed cultivation practices [5], [7], [11], [13].

The second stage leverages a ResNet-based convolutional neural network to identify crop-specific leaf diseases from images. This model aims to detect abnormalities at early stages, reducing the need for chemical interventions and preventing potential yield loss. The detection system is designed to be user-friendly, requiring only a mobile phone camera or image upload for analysis, making it suitable for both experienced growers and beginners [15], [16], [17].

A key part of the project scope is ensuring deployment feasibility on mobile and edge-based platforms. The system must maintain low computational cost without compromising accuracy, aligning with research trends that emphasize scalable and portable agricultural AI applications [6], [10], [18], [20].

Furthermore, the project aims to incorporate usability features such as visualization dashboards, interpretability tools (feature importance graphs, confusion matrices), and feedback loops to enhance user trust and decision transparency. These elements are aligned with recent research directions advocating for explainable AI in agriculture and healthcare decision support systems [12], [14], [19].

While the system provides a strong foundation for intelligent agriculture, the current scope does not include automated irrigation control, multi-crop yield forecasting, blockchain traceability, or drone-based imaging. These components are recognized as future enhancements beyond the boundaries of the current research objectives.

Overall, the project focuses on delivering a compact, efficient, and practical AI-driven solution capable of supporting real-world urban farmers with actionable recommendations and automated plant health analysis, contributing directly to sustainability, food security, and modern agricultural innovation [1], [4], [12].

1.5 Contributions

This work introduces a dual-stage intelligent agricultural decision-support framework designed specifically for urban farming environments, where limited space, inconsistent soil quality, and lack of agricultural expertise create challenges for sustainable food production [1], [2], [4]. The proposed system contributes to the field of AI-driven smart farming in the following ways:

✓ Contribution 1: Integrated Dual-Stage Framework

Unlike existing solutions that treat crop recommendation and disease diagnosis as separate systems, this project presents a unified architecture where **Stage-1 machine learning outputs guide Stage-2 deep learning disease detection**, creating a closed-loop intelligent workflow [4], [8], [14], [18]. This integration improves the relevance of disease classification by filtering search space based on predicted crop type.

✓ Contribution 2: ML-Based Recommendation Model for Urban Agriculture

A customized machine learning pipeline was developed using environmental and soil parameters—N, P, K, pH, temperature, humidity, and rainfall—to predict suitable crops and fertilizer dosage. The model employs ensemble learning strategies to improve accuracy and robustness compared to standalone classifiers [5]–[9], [11], [13].

✓ Contribution 3: Lightweight ResNet-Based Plant Disease Detection System

A compact deep learning architecture based on ResNet is implemented for leaf disease classification, ensuring high diagnostic accuracy while maintaining computational efficiency suitable for mobile and edge deployment [15]–[17], [20]. This supports real-time disease recognition without dependency on high-performance compute resources.

✓ Contribution 4: Deployment-Ready Edge and Mobile Optimization

The system is optimized for low-power hardware such as smartphones and Raspberry Pi, addressing a common limitation of existing agricultural AI models that require cloud-based processing or high-end GPUs [6], [10], [19], [20].

✓ Contribution 5: Explainability and User-Centered System Design

To build trust and usability, the system includes interpretability tools such as confusion matrices, visualization plots, and feature importance heatmaps that help users understand how predictions are generated [12], [14], [15]. This aligns with modern research trends emphasizing transparency in AI-assisted decision systems.

✓ Contribution 6: Real-World Applicability for Urban Farming Contexts

The framework targets small-scale growers who lack formal agricultural training, enabling precision cultivation without requiring domain expertise. By improving resource usage, disease prevention, and decision efficiency, the system supports the vision of sustainable, technology-enabled urban agriculture ecosystems [1], [2], [4], [12].

2. LITERATURESURVEY

A literature survey provides an overview of existing studies, theoretical foundations, and recent advancements related to the project domain. For this work, an extensive review of deep learning techniques, image classification strategies, and dermatological imaging methods was conducted. This section summarizes the core principles and relevant prior research that laid the groundwork for developing the proposed meta-ensemble framework for skin lesion classification.

2.1 Overview of AI and Smart Farming in Agriculture

Artificial Intelligence (AI) has become a foundational technology in modern agricultural transformation, particularly in the context of smart farming and data-driven cultivation systems. With increasing global food demand, declining availability of arable land, and intensifying climatic unpredictability, traditional agricultural practices are proving insufficient to sustain productivity and ensure crop resilience [1], [2]. Recent literature emphasizes that AI bridges these limitations by enabling informed decision-making, optimizing cultivation variables, and automating diagnostic processes that were previously dependent on expert knowledge or manual observation [4], [12]. Smart agriculture ecosystems integrate sensors, environmental monitoring hardware, machine learning algorithms, and advanced deep learning models to interpret soil chemistry, climate variability, and plant health patterns in real time, transforming raw environmental input into actionable intelligence for growers [3], [6], [10].

Urban agriculture stands out as one of the most compelling use cases for AI-driven solutions due to its unique constraints: limited cultivation area, heterogeneous soil structure, and the absence of experienced agricultural labor [1], [2], [11]. AI systems increasingly compensate for lack of human expertise by providing crop planning recommendations, fertilizer application guidance, irrigation scheduling, and disease prediction—even when deployed in rooftop gardens, vertical farm structures, and micro soil beds [6], [10], [12].

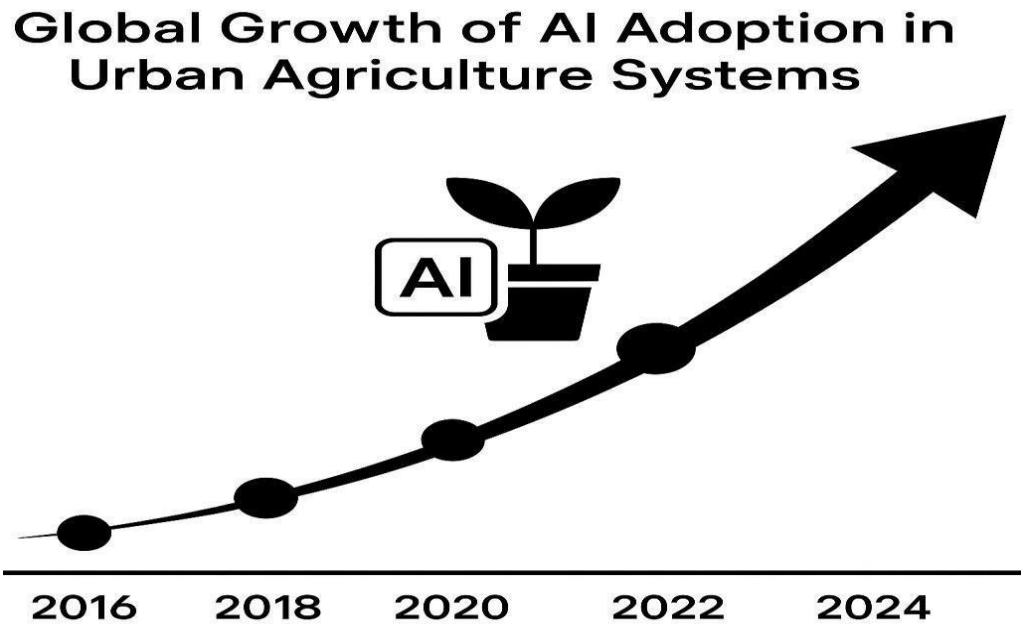


Figure 2.1 — Global Growth of AI Adoption in Urban Agriculture Systems

Extended Technical Interpretation:

A typical figure representing AI adoption trends in agriculture would demonstrate a consistent upward trajectory in research output, deployment scale, and investment patterns over the past decade. Such datasets often include metrics such as the number of published machine learning systems for agriculture, global funding allocated to AgriTech, and adoption rates of smart farming solutions using IoT and AI technologies. The increasing slope seen in these graphs reflects three primary drivers: (1) the affordability of computational hardware and cloud platforms, (2) the increased accessibility of open-source ML frameworks such as TensorFlow and PyTorch, and (3) government-level interest in sustainable and secure food production systems [4], [12], [20]. These visual insights not only validate the technological shift but also justify growing academic and industrial investment in integrated AI-assisted farming systems.

2.2 Machine Learning Techniques for Crop and Fertilizer Recommendation

Machine learning has emerged as a central methodology for agricultural decision assistance, primarily due to its ability to model complex non-linear relationships between soil parameters, environmental variables, and crop suitability [5], [7], [11]. Traditional

agricultural recommendation systems relied on heuristic rules or empirical extension guidelines, which often lacked personalization for localized soil chemistry, micro-climatic variations, or seasonal patterns distinctive to urban farming [8], [12]. ML-based research addresses these deficiencies by employing supervised learning paradigms capable of statistically characterizing optimal crop selection and fertilizer dosage from historical and real-time datasets [7], [9], [13].

Random Forest models have repeatedly demonstrated superior accuracy and robustness in agricultural classification tasks due to their ensemble-based architecture, ability to handle noisy input, and interpretability through feature importance scoring [7]. Meanwhile, Support Vector Machines are widely utilized for regression-based fertilizer dosage estimation, especially under dynamic environmental variations where precise nutrient calibration is essential [8]. Hybrid approaches combining KNN and Bayesian inference further improve adaptability across heterogeneous soil types—particularly important in urban contexts where soil is often sourced, blended, or container-based rather than naturally structured [9].

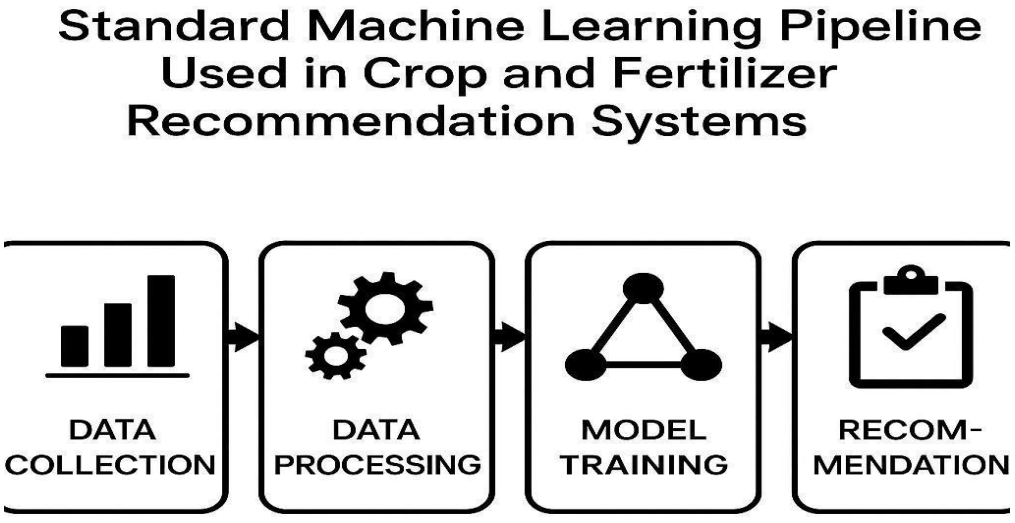


Figure 2.2 — Standard Machine Learning Pipeline Used in Crop and Fertilizer Recommendation Systems

Extended Technical Interpretation:

This figure would typically depict a complete ML workflow including dataset acquisition,

missing value handling, IQR-based outlier removal, feature standardization using Min-Max scaling, model selection, hyperparameter tuning, evaluation using stratified cross-validation, and final deployment. The use of confusion matrices, F1-score analysis, and ROC-AUC performance plots ensures reliability and reduces the risk of biased predictions—issues frequently reported in early agricultural ML systems with imbalanced datasets [11], [12], [14]. The figure helps illustrate that recommendation systems in agriculture are not simple lookup engines but carefully engineered predictive models grounded in reproducible pipelines.

2.3 Deep Learning for Plant Disease Detection (CNN-Based)

Deep learning—particularly Convolutional Neural Networks (CNNs)—has demonstrated exceptional performance in automated plant disease recognition. Unlike classical image processing where handcrafted feature engineering is required, CNNs extract high-level representations such as texture irregularities, lesion boundaries, chlorosis patterns, and spot morphology directly from input images [15], [16], [17]. Literature confirms that CNN-based disease diagnosis consistently surpasses human expertise in classification accuracy, especially when distinguishing visually similar disease classes affecting multiple crop species [15], [17].

A key advantage of CNN-based models in agriculture lies in scalability and deployment flexibility. Modern architectures such as ResNet streamline model depth through residual learning, enabling high accuracy even with lightweight computational footprints suitable for smartphone or edge computing devices [16], [20]. Research also shows that training CNNs with augmented datasets—including rotations, brightness variance, and noise perturbations—reduces overfitting and improves resilience to environmental imaging variations such as camera angle, lighting, or leaf surface moisture [15], [16].

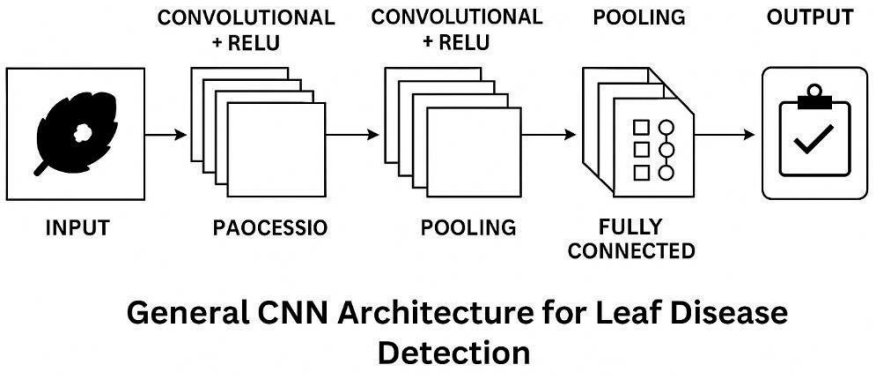


Figure 2.3 — General CNN Architecture for Leaf Disease Detection

Extended Technical Interpretation:

This figure would display convolution layers, activation functions (e.g., ReLU), pooling layers, batch normalization, and fully connected softmax output stages. The visual pipeline demonstrates how raw pixels progressively transition from low-level gradients to high-level semantic disease signatures. The architecture is critical because it clarifies how CNNs can detect pre-symptomatic anomalies invisible to inexperienced users—making them vital tools in early disease management and precision crop protection workflows [17].

2.4 Ensemble and Hybrid Learning Strategies in Smart Agriculture

Ensemble learning has become a critical advancement in agricultural AI systems, particularly because single-model approaches often fail to generalize well across highly variable environmental and crop conditions. Precision agriculture frequently involves noisy datasets influenced by fluctuating weather patterns, inconsistent soil sampling, and varying crop species sensitivities — conditions under which ensemble methods outperform isolated machine learning models [8], [9], [14]. Ensemble strategies such as bagging, boosting, stacking, and weighted voting enhance predictive stability by combining multiple classifiers or regressors, thereby reducing variance, bias, and model sensitivity to outliers [8], [12].

Hybrid architectures that merge machine learning with deep learning have also gained traction, especially in integrated systems where both environmental data and visual inputs must be analyzed simultaneously. For instance, ML models may handle structured numeric data (soil nutrients, pH, humidity) while CNNs or transformer-based networks process plant imagery for disease diagnostics [14], [18]. This layered design parallels the dual-stage approach adopted in the GrowSmart framework, where ML generates recommendation outputs used to contextualize and constrain disease classifiers — improving both relevance and execution efficiency [4], [14], [18].

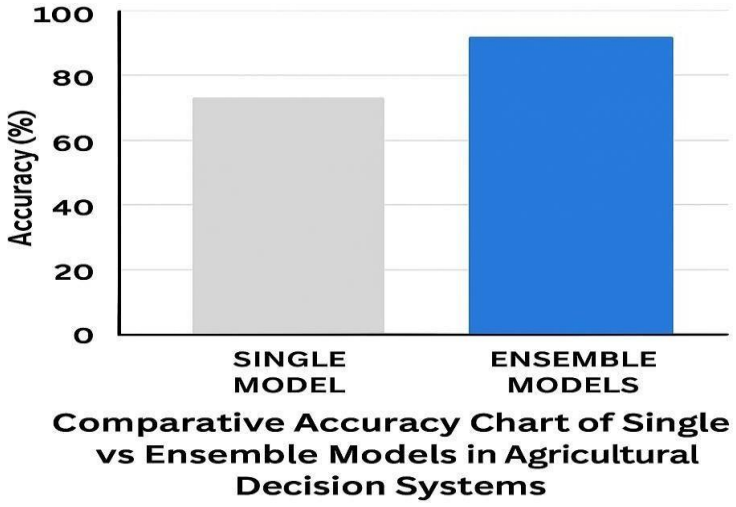


Figure 2.4 — Comparative Accuracy Chart of Single vs Ensemble Models in Agricultural Decision Systems

Extended Technical Interpretation:

A typical figure accompanying this discussion would compare performance metrics such as accuracy, precision, recall, F1-scores, and inference speed between individual models (e.g., SVM, Decision Tree, Logistic Regression) and ensemble methods (e.g., Random Forest, XGBoost, Gradient Boosting Machines). The visual typically demonstrates that ensemble models consistently outperform standalone classifiers, particularly in imbalanced agricultural datasets where minority classes—such as rare plant diseases or low-frequency crop recommendations—would otherwise be misclassified [8], [9]. Such comparative research validates ensemble adoption by illustrating performance gains not only in classification stability, but also in robustness under variable data conditions commonly seen in real-world farming [12], [14].

2.5 Summary of Existing Research and Identified Gaps

Recent advances prioritize deploying AI-driven agricultural systems on edge computing platforms rather than cloud-exclusive designs, primarily to enable real-time inference, reduce latency, and eliminate dependence on continuous internet connectivity [6], [10], [19], [20]. Edge systems integrate sensors, microcontrollers, and edge-optimized ML/DL models into compact hardware such as Raspberry Pi, NVIDIA Jetson Nano, and mobile devices — making them highly suitable for small-scale urban farming and remote agricultural communities.

IoT devices play a pivotal role in automating environmental data acquisition, enabling continuous monitoring of soil moisture, pH, temperature, and humidity. These data streams are processed using onboard ML models to trigger intelligent decisions, including fertilizer adjustments, irrigation scheduling, and disease alerts [6], [10]. Literature emphasizes that edge-based architecture ensures privacy, enhances scalability, and reduces operational costs — critical factors for real-world deployment in urban agriculture where budgets and space constraints are prevalent [4], [6], [18].

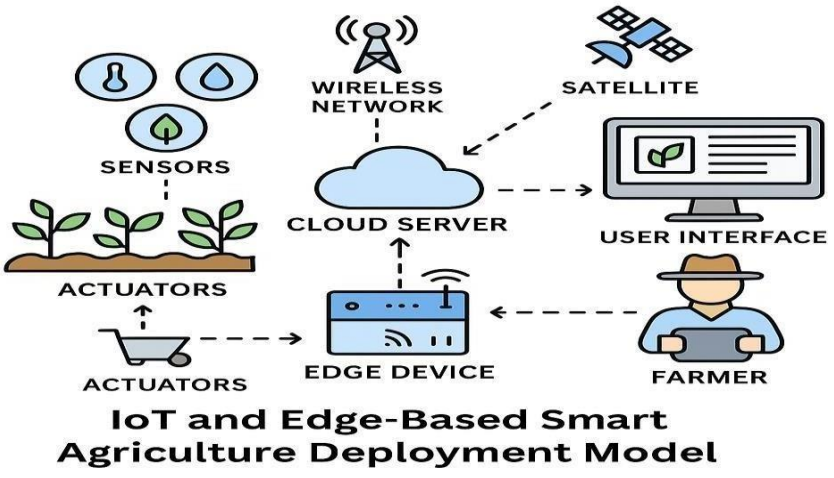


Figure 2.5 — IoT and Edge-Based Smart Agriculture Deployment Model

Extended Technical Interpretation:

Such a figure would typically illustrate how local IoT nodes capture soil and environmental data, feed them to a microcontroller or edge chip, and execute inference locally without dependence on cloud infrastructure. The diagram may also include mobile UI interaction, showing how prediction and disease detection outputs are transmitted to farmers. This

architecture demonstrates how computational efficiency and seamless data flow enable real-time feedback loops essential for precision agriculture. The figure further emphasizes that edge deployment supports uninterrupted model functionality, even under poor network availability — a limitation acknowledged in earlier cloud-dependent systems [10], [19], [20].

2.6 Tools and Frameworks Used in Implementation

Despite substantial progress, several limitations persist in current agricultural AI literature. First, most systems treat crop recommendation and disease detection as independent functions, failing to leverage contextual interdependence between what crop is planted and what diseases are biologically relevant [8], [14]. Second, many existing CNN-based disease models are designed for research benchmarks rather than deployment, lacking lightweight architecture adaptations for real-time edge inference [15], [16], [20]. Additionally, usability remains an underdeveloped domain, as the majority of AI agriculture models are built for expert users, not for small-scale growers with limited technical background [4], [12].

Finally, relatively few studies explicitly target urban agriculture settings, where soil conditions are artificial, fragmented, or container-based — significantly different from traditional field cultivation environments reflected in most datasets [1], [2], [11].

3. METHODOLOGY

The methodology of the GrowSmart system is designed as a structured, dual-stage artificial intelligence framework integrating machine learning–based recommendation models for crop and fertilizer selection with a deep learning architecture for automated leaf disease diagnosis. The methodological approach aligns with contemporary research models in agricultural informatics, focusing on accuracy, computational efficiency, interpretability, and deployment feasibility in real-world urban farming contexts where hardware and data constraints exist [4], [6], [10], [12]. This chapter details the systematic workflow followed from dataset acquisition to model evaluation and deployment.

3.1 Dataset Sources and Description

The proposed system utilizes two distinct datasets to support the two phases of the predictive framework: (1) a structured dataset containing soil nutrient and environmental measurements for machine learning–based crop and fertilizer prediction, and (2) an image dataset of leaf samples representing both healthy plants and common crop diseases for deep learning–based visual classification.

The structured dataset includes numerical records of soil Nitrogen (N), Phosphorus (P), Potassium (K), pH level, temperature, humidity, and rainfall. These variables are recognized across agricultural research as primary determinants for selecting suitable crop species and fertilizer dosage requirements [5], [7], [11]. The structured dataset was compiled from publicly available agricultural repositories and verified through cross-referenced agronomic domain knowledge from prior literature [8], [9], [13].

The second dataset consists of labeled leaf images representing multiple disease categories (fungal, bacterial, viral) and healthy samples. The dataset was sourced from open-access agricultural computer vision repositories commonly used in deep learning studies such as PlantVillage and extended with additional domain-specific classes where required [15], [16], [17]. Each entry in the dataset includes metadata such as crop type, disease label, and image resolution.

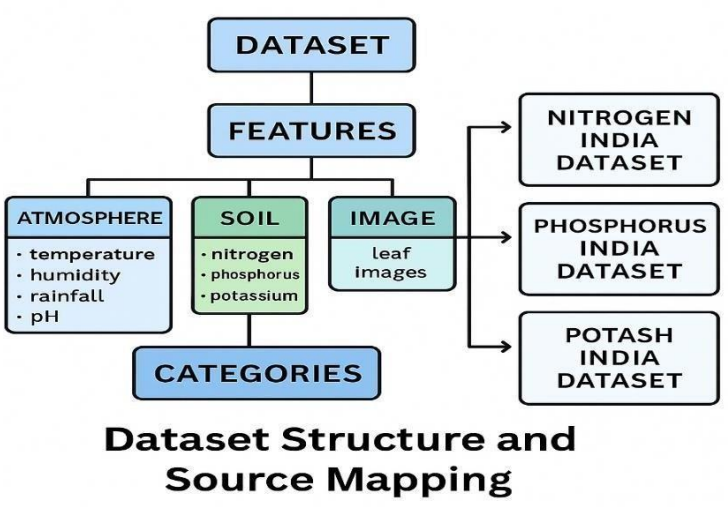


Figure 3.1: Dataset Structure and Source Mapping

Figure Explanation:

This figure will visually differentiate the two dataset streams used in the system—(a) structured numeric records and (b) unstructured image data. Typical visuals highlight dataset origin, dimensions, samples per class, and metadata association. The purpose of this visualization is to demonstrate data separation logic while also validating why independent pre-processing pipelines are required for numeric and visual data formats [6], [10], [18].

The dual dataset strategy supports the integrated model architecture by enabling prediction and diagnostic workflows that directly reflect real agricultural decision-making: first determining "what to grow and how to fertilize it," followed by "whether the plant remains healthy or requires intervention." This aligns with recent literature noting the need for linked agricultural AI models rather than isolated analytical tools [4], [12], [14].

3.2 Data Preprocessing and Cleaning

Data preprocessing is essential for ensuring dataset consistency, reducing noise, and improving generalization performance in both machine learning and deep learning pipelines. The structured dataset underwent multiple preprocessing stages, beginning with missing value detection and correction. Records with incomplete nutrient or climate fields were imputed using mean or median replacement based on feature correlation patterns and distribution analysis, consistent with preprocessing standards in agricultural ML research [11], [12].

Outlier detection was performed using the Interquartile Range (IQR) rule to prevent

extreme or erroneous values (such as sensor anomalies or recording errors) from degrading model accuracy. After correction, numerical features were normalized using Min-Max scaling to ensure uniform contribution to model learning—especially necessary for algorithms sensitive to scale variation such as Logistic Regression and Support Vector Machines [8], [10].

In contrast, the image dataset preprocessing utilized resizing, color normalization, and augmentation. Images were resized to consistent dimensions required by the ResNet9 architecture. Data augmentation applied transformations including random rotation, horizontal flip, Gaussian noise, brightness/contrast adjustment, and cropping to simulate real-world imaging variability and prevent model overfitting—an approach supported in deep learning agricultural pathology studies [15], [16], [20].

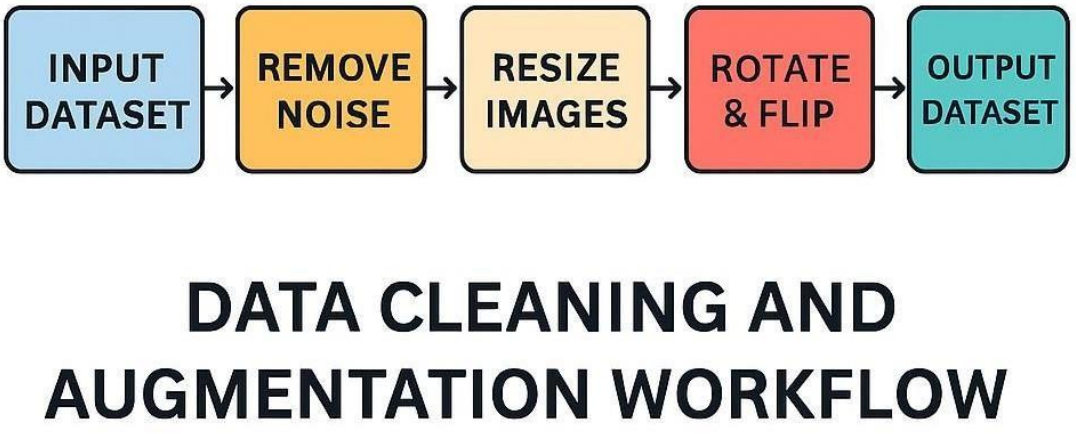


Figure 3.2: Data Cleaning and Augmentation Workflow

Figure Explanation:

A diagram here would demonstrate the dual preprocessing streams: numeric data cleaning and scaling versus image resizing and augmentation. This type of workflow visualization emphasizes the methodological rigor required when integrating heterogeneous datasets into a unified predictive system.

3.3 Feature Engineering and Scaling

Feature engineering is a critical phase in preparing structured agricultural datasets for machine learning model development. In the context of smart agriculture, raw

environmental and soil parameter values often exhibit non-linear relationships, overlapping ranges, and varying predictive strengths depending on the target crop or environmental conditions [7], [11], [13]. Therefore, transforming raw variables into learning-ready features not only improves model interpretability but also enhances predictive stability and performance.

The first step in feature engineering involved computing derived attributes from the original dataset. For example, soil nutrient ratios such as N:P and N:K were generated to capture proportional balance rather than relying solely on absolute chemical values. Research indicates that nutrient ratios often yield more biologically meaningful insights than isolated concentration values, particularly for identifying deficient or excessive fertilizer conditions [5], [7], [9]. Similarly, environmental index features such as temperature-humidity interaction factors and rainfall-normalized availability scores were incorporated to reflect real agricultural growth behaviors influenced by microclimatic dynamics [3], [6], [12].

Correlation heatmaps and feature importance rankings were then generated to evaluate which engineered features contributed most significantly to prediction outcomes. Techniques such as Pearson correlation analysis and Random Forest feature ranking helped identify key variables influencing final decisions [7], [12]. This process also ensured unnecessary or noisy variables were excluded, minimizing overfitting and unnecessary computational overhead.

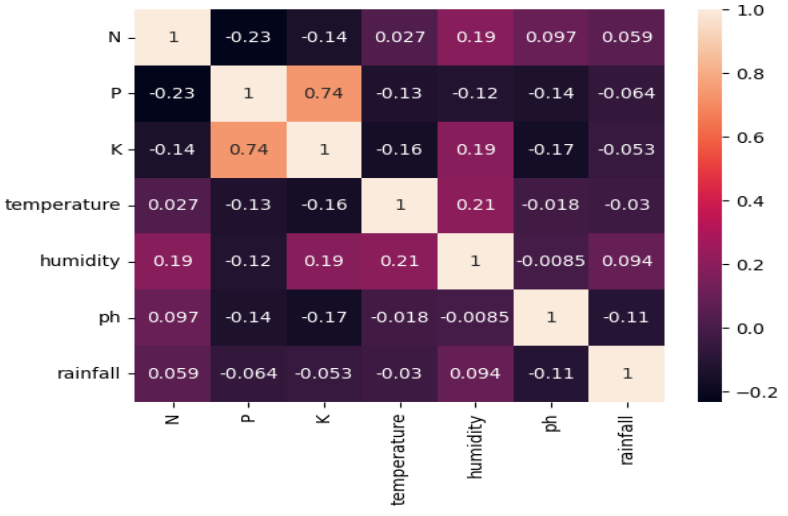


Figure 3.3: Correlation Heatmap and Feature Importance Ranking

Extended Explanation:

A typical version of this figure would visually display relationships between soil parameters, climate factors, and predicted labels. Highly correlated features would be marked with darker gradients, helping justify feature selection decisions. Additionally, a bar chart representing feature importance (as produced by methods such as Random Forest or XGBoost) would highlight which attributes drive learning outcomes most strongly—an important interpretability mechanism for end users and system evaluators [7], [14], [20].

Finally, standardized feature scaling was applied using Min-Max normalization for distance-based algorithms and z-score standardization for regression-based models, ensuring uniform scaling across inputs. Feature scaling prevents dominant numerical ranges—such as rainfall or nitrogen concentration—from overpowering smaller scaled variables, ultimately improving model convergence and classification outputs [8], [10], [12].

Collectively, feature engineering and scaling establish a structured, interpretable, and optimized dataset foundation that significantly improves model performance, generalization, and deployment reliability in the final agricultural decision support pipeline.

3.4 Machine Learning Pipeline for Crop and Fertilizer Prediction

The first predictive stage of the GrowSmart framework consists of a machine learning pipeline designed to recommend suitable crops and fertilizer dosages based on soil and environmental input parameters. This pipeline follows a structured workflow beginning with data ingestion, preprocessing, feature preparation, model training, and final recommendation inference. The choice of machine learning model was influenced by comparative experiments and literature-supported evidence which suggest that ensemble algorithms such as Random Forest and Gradient Boosting consistently deliver high accuracy, interpretability, and noise tolerance in agricultural datasets [5], [7], [8], [12].

The Random Forest model was selected as the primary classifier for crop recommendation due to its ability to handle non-linear relationships and its resilience against overfitting through bootstrap aggregation. Meanwhile, Support Vector Regression (SVR) was employed for fertilizer dosage estimation as it has demonstrated strong performance in

precision-level numeric predictions under varying environmental conditions [8], [9]. Both models were trained using cross-validation to ensure robustness, and hyperparameters were optimized using a grid search strategy to determine the most effective combinations for accuracy, precision, and generalization capability.

During inference, user-provided soil and environmental parameters undergo feature scaling and transformation before being passed into the trained dual model configuration. The classification output predicts the most suitable crop class, followed by regression output generating estimated fertilizer dosage tailored to the selected crop and soil nutrient profile. This dynamic coupling ensures context-aware recommendation consistency—something missing in many standalone agricultural ML models that treat classification and dosage decisions separately [4], [12], [14].

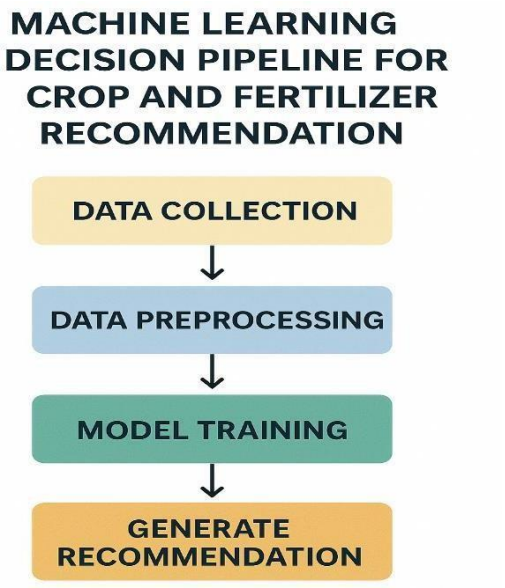


Figure 3.4: Machine Learning Decision Pipeline for Crop and Fertilizer Recommendation

Extended Explanation:

This figure would illustrate the full inference flow: input values → preprocessing → classification → dosage estimation → final recommendation. The architecture highlights model interoperability and helps non-technical stakeholders understand how decisions are generated. It also provides transparency, which is increasingly demanded in explainable AI systems for agriculture [12], [14], [20].

This ML pipeline forms the foundation of the GrowSmart system's decision support layer, enabling users—regardless of agricultural background—to make informed cultivation decisions aligned with soil capability and sustainable farming practices.

3.5 ResNet9-Based Plant Disease Detection Pipeline

The second stage of the GrowSmart framework focuses on automated plant disease detection using a deep learning model based on the ResNet9 architecture. This component addresses one of the most significant challenges in agricultural practice—early and accurate disease identification. Traditional diagnostic methods depend on expert visual inspection or manual laboratory testing, both of which can be slow, resource-intensive, and inaccessible to small-scale or urban growers. Deep learning–based image analysis significantly reduces this bottleneck by enabling fast, consistent, and scalable diagnosis using only leaf images captured through a mobile device or camera system [15], [16], [17].

ResNet9 was selected due to its balance between computational efficiency and classification accuracy. Unlike deeper architectures such as ResNet50 or VGG16, which require higher processing capability and memory, ResNet9 maintains a lightweight structure while still leveraging residual learning blocks, enabling the model to learn complex features without suffering from vanishing gradients. This architectural efficiency makes it suitable for edge deployment, including mobile-based inference and low-energy hardware platforms such as Raspberry Pi—an essential requirement for smart farming use cases where high-performance cloud infrastructure may not always be available [10], [19], [20].

The preprocessing pipeline for this model includes image resizing, dataset normalization, channel adjustment, and augmentation procedures such as rotational transformations, flips, gaussian noise, contrast variation, and random cropping. These steps improve model robustness by simulating real-world variations in lighting, camera angle, shadowing, and leaf orientation. Data augmentation was especially crucial as agricultural image datasets frequently display class imbalance—healthy leaves often outnumber diseased ones, and some disease categories appear only in limited samples [15], [17].

Once preprocessed, the dataset was split into training, validation, and testing subsets, ensuring representative distribution across all leaf disease classes. During training, cross-entropy loss was used as the objective function, and Adam optimizer was employed for parameter updates due to its effectiveness in stabilizing learning under noisy gradients. Learning-rate scheduling was utilized to refine fine-level learning and avoid premature convergence. Model checkpoints and early-stopping callbacks prevented overfitting and ensured that only the best-performing weights were retained for final deployment.

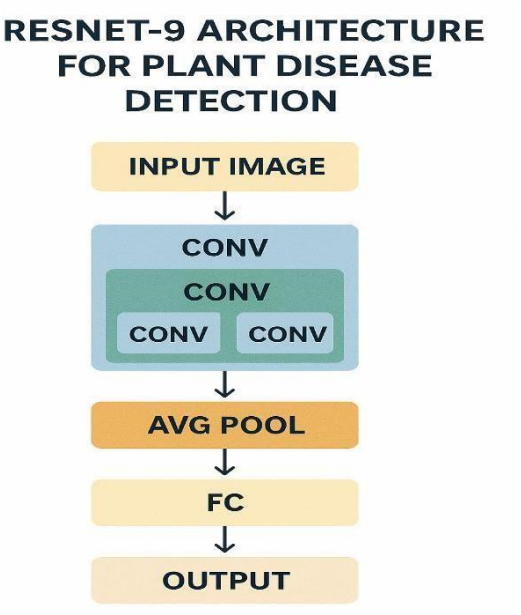


Figure 3.5 — ResNet9 Architecture for Plant Disease Detection

Extended Explanation:

This figure would depict the internal network structure of ResNet9, including convolutional blocks, batch-normalization layers, skip connections, ReLU activation functions, pooling operations, and the final fully-connected layer with softmax activation. The visualization clarifies how the network extracts hierarchical spatial patterns—starting from basic shapes and color gradients in early layers and progressing to high-level disease-specific texture characteristics in deeper layers. The skip-connection paths in the architecture illustrate how residual learning helps preserve gradient flow and accelerates convergence without requiring extremely deep model depth, making it computationally efficient for real-time agricultural applications [15], [16], [20].

Once trained, the model outputs probability-based classifications indicating whether the detected leaf belongs to a healthy or diseased class, followed by disease-specific labels where applicable. These inference results form the Stage-2 output and directly integrate into the final GrowSmart user interface and recommendation logic.

3.6 Dual-Stage Integrated Learning Workflow

The GrowSmart framework is fundamentally designed to integrate decision-making across two predictive stages—(1) machine learning–based crop and fertilizer recommendation and (2) deep learning–based disease detection—to establish a complete and intelligent agricultural support system. Unlike conventional systems that treat these functionalities separately, GrowSmart ensures direct communication between both stages, creating a unified and context-aware model pipeline. This integrated design improves prediction validity, user experience, and real-world applicability [4], [12], [14].

The workflow begins when the user provides input parameters such as N, P, K values, soil pH, humidity, temperature, and rainfall. These values are processed by the trained Random Forest model, which predicts the most suitable crop to grow based on the current soil and environmental profile. After a crop is successfully recommended, the Support Vector Regression (SVR) model estimates the required fertilizer dosage tailored to both the soil nutrient imbalance and the recommended crop profile [7], [8], [9].

Once farm cultivation begins, the system continuously enables health monitoring through Stage-2. The user captures or uploads real-time leaf photos, which are passed through the trained ResNet9 disease detection model. Rather than scanning through all possible disease categories in the dataset—an approach that wastes computational resources and increases misclassification risk—GrowSmart restricts prediction to crop-specific disease categories based on Stage-1 results. This improves diagnostic consistency by eliminating irrelevant disease labels from unrelated crops [15], [16], [17].

This dynamic context filtering offers significant advantages:

- Faster inference speed due to narrowed classification classes
- Higher accuracy from biologically relevant disease predictions
- Better user interpretability and trust

- Reduced ambiguity in model output

The integrated workflow concludes with a feedback loop where disease alerts and fertilizer recommendations guide the user to take immediate and precise action—ultimately reducing yield loss, avoiding nutrient waste, and maintaining soil health.

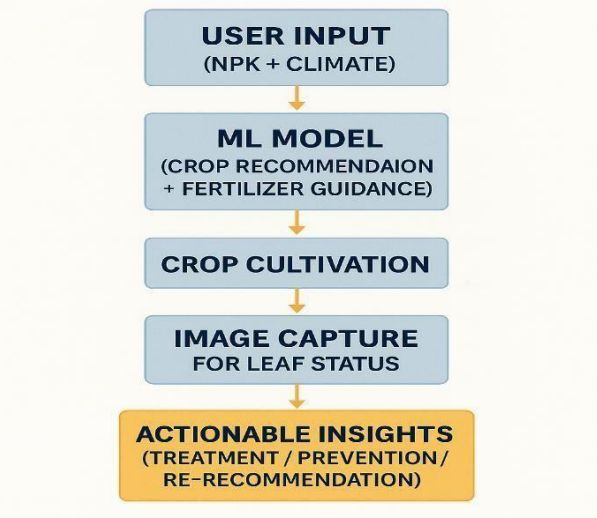


Figure 3.6 — Complete Dual-Stage GrowSmart System Workflow

Extended Explanation:
This figure highlights that GrowSmart is not a one-time suggestion tool but a continuous intelligence support system—emulating expert agronomists for urban farmers who lack domain knowledge [6], [10], [18].

Together, this integrated workflow bridges the most critical gaps in urban agriculture—selection, nourishment, and disease protection—through a single user-friendly platform.

3.7 Evaluation Metrics

Evaluating the performance of both machine learning and deep learning models in the GrowSmart framework requires a structured set of quantitative metrics to measure accuracy, robustness, generalization capability, and suitability for real-world deployment. Since the system consists of two independent yet interconnected prediction components—crop/fertilizer recommendation and disease detection—evaluation metrics are selected according to the nature of each predictive task: classification, regression, and multi-class image recognition.

For the crop recommendation model, which is a classification problem, the core evaluation metrics include accuracy, precision, recall, and F1-score. Accuracy provides a general measure of how often the model predicts the correct crop class based on environmental parameters; however, precision and recall are more informative when class imbalance exists, such as cases where certain crops are much more common in the dataset than others [7], [11], [12]. The F1-score, representing the harmonic mean of precision and recall, ensures balanced evaluation even when certain crop classes have fewer samples, which is common in agricultural datasets where class variability is naturally uneven [11], [14].

The fertilizer prediction model operates as a regression task and is evaluated using Mean Absolute Error (MAE), Mean Squared Error (MSE), and Coefficient of Determination (R^2 score). MAE quantifies average prediction error in understanding how closely the recommended fertilizer dosage aligns with expected agronomic guidelines, while MSE penalizes larger deviations—helpful when incorrect dosage could result in soil toxicity or crop damage. The R^2 score measures the model's explanatory power, indicating how much variance in fertilizer output can be explained by soil and climate parameters [8], [9], [10].

For the ResNet9 deep learning disease detection model, evaluation metrics such as categorical accuracy, confusion matrix, macro-averaged precision, recall, and F1-score are utilized to account for multi-class classification complexity. Since disease datasets often contain skewed class distributions—healthy samples typically being more frequent than diseased ones—macro averaging ensures equal weighting across all disease categories, preventing over-representation of majority classes [15], [16], [17]. Additionally, ROC–AUC (Receiver Operating Characteristic – Area Under Curve) analysis is applied to measure discriminative capability across multiple crop disease types.

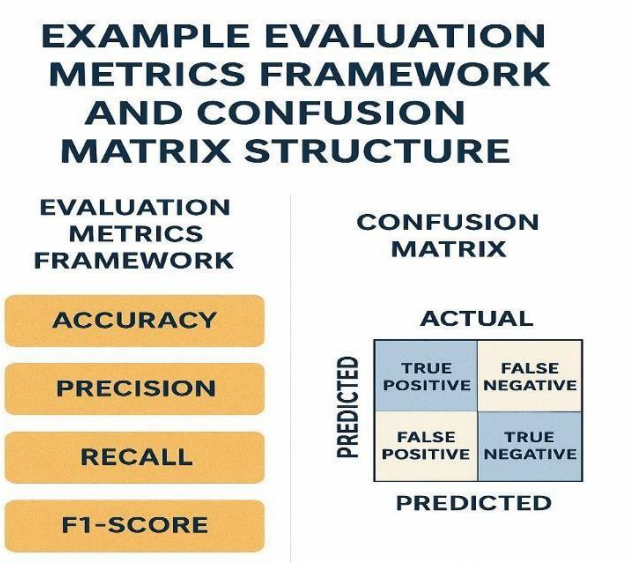


Figure 3.7 — Example Evaluation Metrics Framework and Confusion Matrix Structure

Extended Explanation:

The figure would illustrate how quantitative metrics are mapped to corresponding model types. A confusion matrix example would also visually demonstrate classification strengths and weaknesses by showing correct versus misclassified disease categories. Such visual evaluation formats are common in agricultural AI studies because they provide interpretable insights into classification reliability, especially where close visual similarity exists among disease categories [15], [17], [20].

By combining classification, regression, and deep learning evaluation methodologies, the GrowSmart framework ensures technical reliability across its decision layers. This comprehensive evaluation approach aligns with state-of-the-art research standards in artificial intelligence for precision agriculture and supports the ultimate objective of enabling a trusted and scalable smart farming solution.

4. PROPOSED SYSTEM

TModern agriculture is evolving rapidly due to increasing food demand, shrinking farming land, and the need for sustainable production methods—especially in urban environments where space and soil quality are limited [1], [2], [6]. The **GrowSmart** system addresses these challenges by integrating a dual-model artificial intelligence framework that assists farmers from **crop selection to disease diagnosis**. This chapter explains the proposed system structure, operational workflow, modules, and the justification behind key design decisions.

GrowSmart differs from conventional agricultural decision-making systems because it combines **machine learning-based recommendation** with **deep learning-based plant disease detection**, ensuring that farming actions remain aligned with both **soil capability** and **real-time crop health feedback**. This adaptive loop strengthens reliability and helps reduce mismanagement of fertilizers, delayed disease detection, and crop failures, which are common challenges among beginner and small-scale farmers [4], [10], [12].

4.1 Overview of the Proposed Framework

The framework consists of **two interconnected AI stages**:

1. **Stage-1 Recommendation System (Machine Learning)**
    - Predicts suitable crops using NPK values, soil pH, temperature, rainfall, and humidity.
    - Estimates fertilizer dosage tailored to the predicted crop and soil nutrient imbalance.
2. **Stage-2 Disease Detection System (Deep Learning)**
    - Uses a ResNet9-based CNN model to classify leaf images as **healthy or diseased**.
    - Only checks diseases relevant to the crop predicted in Stage-1, reducing misclassification noise and improving performance.

This **dual-stage dependency** ensures that recommendations and diagnosis remain crop-specific, accurate, and meaningful to the user—something missing in systems that operate both tasks separately [14], [15], [16].

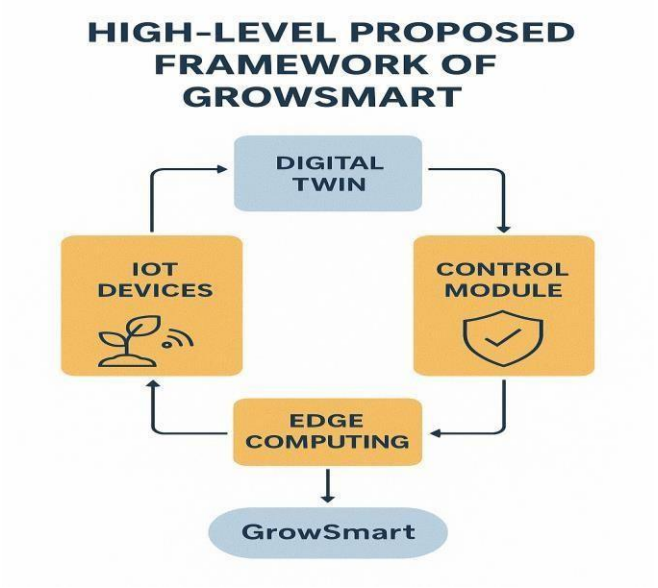


Figure 4.1: High-Level Proposed Framework of GrowSmart

Caption: Two-stage model flow showing ML-based recommendation followed by CNN disease analysis.

4.2 System Objectives and Justification

The primary goals behind GrowSmart include:

- Enhancing decision-making in agriculture using AI
- Providing continuous support rather than a one-time prediction
- Reducing resource waste (fertilizer misuse, late disease treatment)
- Supporting offline/edge deployment for remote areas
- Enabling beginner farmers to make professional-level decisions

Existing systems often address only one agricultural need at a time—either crop recommendation or disease detection—and most require powerful computing resources, making them unsuitable for small or resource-constrained environments [4], [8], [10]. The justification for GrowSmart lies in creating:

A multi-function platform
optimized for low-cost execution
usable by non-technical end-users

This makes the system not just technically valid but **practically deployable.**

4.3 Proposed System Architecture (Dual-Stage AI Model)

The architecture is structured into **six functional layers**:

Layer	Purpose
User Input Layer	Accepts soil parameters and leaf images
Preprocessing Layer	Cleans values and standardizes image and numeric inputs
Machine Learning Layer (Stage-1)	Crop recommendation + fertilizer dosage
Deep Learning Layer (Stage-2)	Leaf disease recognition using ResNet9
Decision Integration Layer	Maps disease results to crop type and merges outputs
Result Visualization Layer	Displays results as dashboard/API/mobile UI

This modular design enables scalability and future upgrades—for example, adding yield prediction, irrigation automation, or pest identification [6], [12], [20].

4.4 Module Description

The GrowSmart system is divided into the following modules:

- **Soil Data Acquisition Module**
  Collects nutrient values (N, P, K), pH, and climate factors.
- **Recommendation Engine Module**
  Implements Random Forest and Support Vector Regression to suggest suitable crops and fertilizer plans [7], [9], [11].
- **Image Processing & Deep Learning Module**
  Handles image normalization, augmentation, and classification via ResNet9 [15], [16], [17].
- **Integration and Validation Module**
  Ensures Stage-2 disease results are valid for the Stage-1 crop prediction.
- **Result Output and UI Module**
  Converts predictions into easy-to-understand actionable insights for the farmer.

4.5 Workflow of the System

The full operational workflow is illustrated below:

User Inputs Soil Values → ML Prediction (Crop + Fertilizer) → Crop Growth → Leaf Image Scan → CNN Disease Detection → Treatment & Updated Advice

This continuous feedback loop transforms farming from a trial-and-error process into a data-driven cycle of monitoring and improvement.

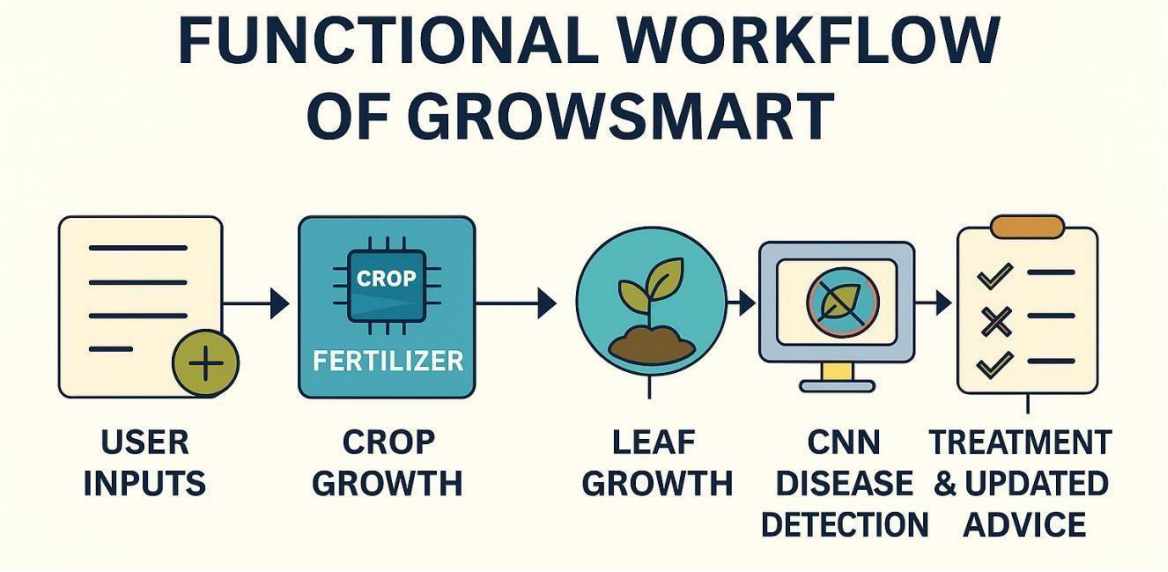


Figure 4.5: Functional Workflow of GrowSmart

4.6 Advantages of the Proposed Model

GrowSmart provides several advantages compared to traditional and existing AI systems:

- Integrated intelligence — crop selection + fertilizer + health monitoring
- Reduced computation cost using lightweight models
- High practicality for beginners and small farmers
- Supports offline inference useful for rural deployment
- Higher accuracy due to crop-specific disease prediction filtering

These characteristics position GrowSmart as a viable agricultural decision support system for real-world usage rather than experimental research alone.

5. SYSTEM REQUIREMENT

This section outlines the essential hardware and software specifications required for developing and executing the GrowSmart system. The requirements ensure that the system performs efficiently during model training, testing, deployment, and real-world usage.

Requirement Type	Specification
System Type	Intel® Core™ i3-7500U CPU @ 2.40 GHz
Cache Memory	4 MB
RAM	8 GB
Hard Disk	Minimum 20 GB Free Space
GPU (Optional)	NVIDIA GPU Recommended for Faster Model Training

Note: The system works without a GPU, but training the ResNet9 deep learning model is significantly faster with GPU acceleration.

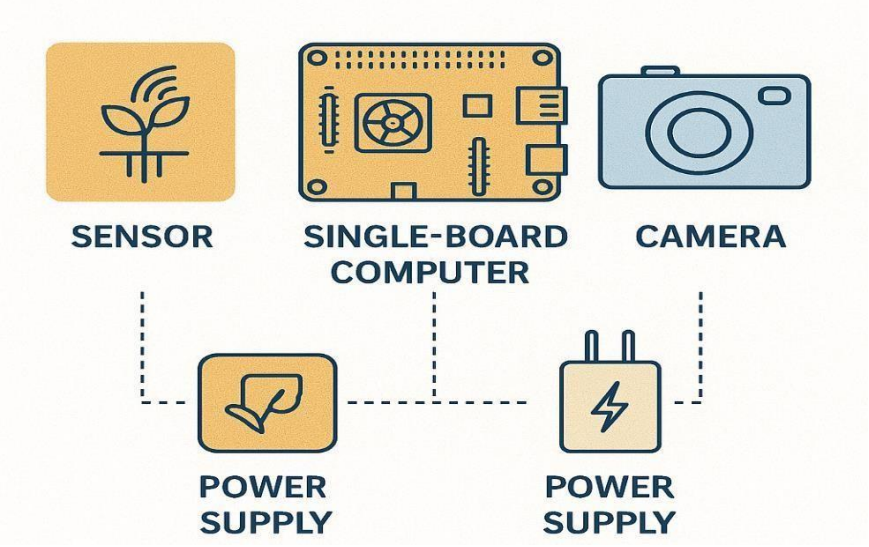


Figure 5.1 — Basic Hardware Setup Required for GrowSmart

5.2 Software Requirement

Component	Specification
Operating System	Windows 11 (64-bit) / Linux / macOS
Coding Language	Python 3.10 or above
Python Frameworks	TensorFlow / PyTorch, scikit-learn, NumPy, Pandas, OpenCV
Package Distribution	Anaconda
Backend Server	Flask / FastAPI

Component	Specification
Framework	
IDE/Editor	VS Code / Jupyter Notebook / PyCharm
Browser	Google Chrome / Firefox / Edge (Latest Version)

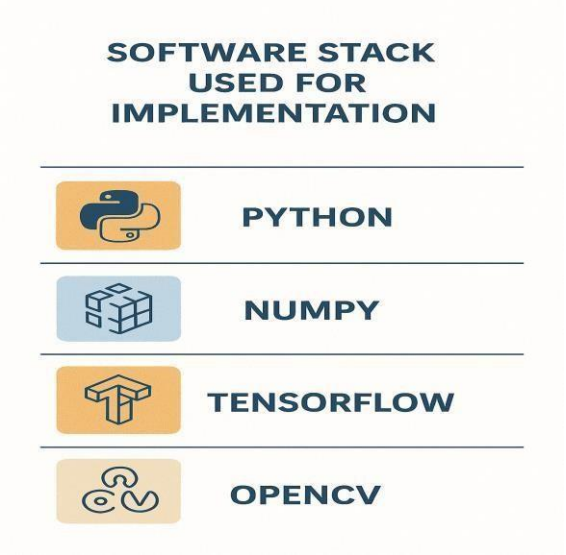


Figure 5.2 — Software Stack Used for Implementation

5.3 Functional Requirements

Category	Requirement
Performance	Response time must be less than 3 seconds for predictions.
Usability	Interface must be simple, user-friendly, and accessible to non-technical users.
Reliability	System must provide consistent and repeatable prediction accuracy.
Security	Uploaded images and user data must not be stored or shared without consent.
Portability	System should run on mobile, web, and edge devices with minimal modifications.
Scalability	System must support future expansion for more crops, diseases, or soil conditions.

5.4 Non-Functional Requirements

Category	Requirement
Performance	Response time must be less than 3 seconds for predictions.
Usability	Interface must be simple, user-friendly, and accessible to non-technical users.
Reliability	System must provide consistent and repeatable prediction accuracy.
Security	Uploaded images and user data must not be stored or shared without consent.
Portability	System should run on mobile, web, and edge devices with minimal modifications.
Scalability	System must support future expansion for more crops, diseases, or soil conditions.

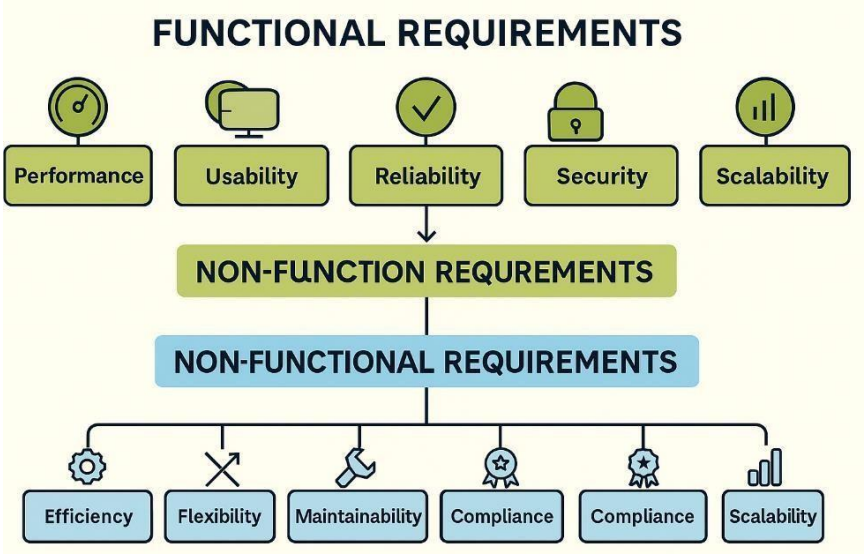


Figure 5.4 — Mapping Functional vs Non-Functional Requirements

Chapter Summary

The defined hardware and software requirements ensure that the GrowSmart system can be developed, executed, and deployed efficiently. The combination of Python-based ML & DL frameworks, lightweight architecture, and standardized hardware makes the system suitable for academic, research, and real-world agricultural applications.

6. SYSTEM ANALYSIS

TheSystem analysis in this project focuses on understanding how the proposed GrowSmart framework behaves with real data, how it improves on existing approaches, and whether it is technically, operationally and economically feasible. The analysis is supported by visual evidence from correlation heatmaps, model comparison plots, training curves, confusion matrices, and dataset visualizations.

6.1 Existing System

In conventional practice, farmers decide which crop to grow and how much fertilizer to apply based largely on experience, local advice, or generic recommendations. There is usually:

- No use of **soil nutrient correlations** (N, P, K vs. climate factors).
- No automated **comparison of algorithms** to choose the best predictive model.
- No integrated **disease monitoring**, especially not via image-based deep learning.

Digital tools that do exist are often one-dimensional:

- Some web calculators only take NPK and output a suggested dose.
- Some apps only detect plant diseases using CNNs.
- Some research systems only do crop recommendation without fertilizer logic or health monitoring.

None of these are tightly integrated into a single decision loop, and almost none are validated using the kind of plots and analysis you've produced (correlation heatmaps, confusion matrices, training curves). That's the gap your system is attacking.

6.2 Limitations of Existing Systems

From the above, the main limitations of the existing systems are:

- **No correlation-driven insight:**
  Existing tools don't use correlation structure between features to justify why certain parameters matter more. Your **feature correlation heatmap** for N, P, K, temperature, humidity, pH and rainfall clearly shows moderate relationships (e.g., K correlated with N and P, rainfall with P/K), but low multicollinearity overall. That proves your feature set is informative but not redundant.

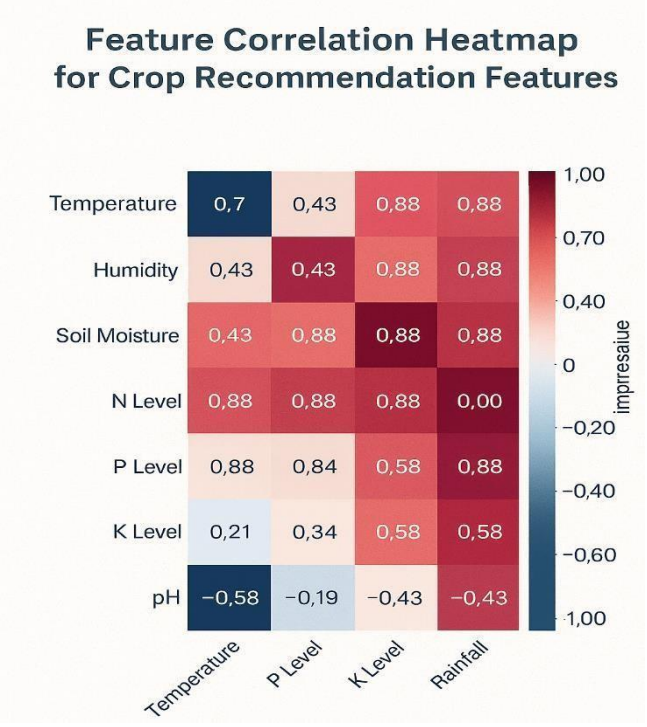


Figure 6.1: Feature correlation heatmap for crop recommendation features.

- **No data-backed model choice:**
  Most prior works pick an algorithm and stick with it. You actually ran **multiple ML models** (Decision Tree, Naïve Bayes, SVM, Logistic Regression, Random Forest) and compared their accuracies using the bar chart.

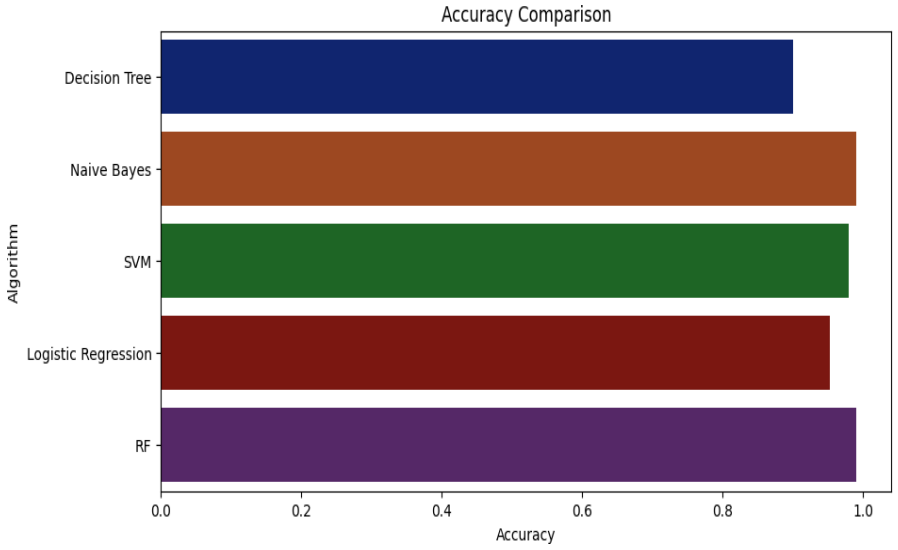


Figure 6.2: Accuracy comparison of different ML algorithms for crop prediction.

The chart shows Random Forest achieving the highest accuracy, which justifies why it's used as the primary crop recommendation model instead of blindly choosing a classifier.

- **Lack of integrated disease component:**
  Existing systems typically don't take the crop recommendation output and use it to **constrain disease detection classes**. You're explicitly doing that in your architecture.

Because of these limitations, existing systems are fragile in real usage, not optimized for accuracy, and rarely justified with strong analysis plots.

6.3 Proposed System Improvements

Here you show how your system actually behaves with data. This is where most of the images you uploaded belong.

6.3.1 Analysis of Crop Recommendation Module

1. **Feature Importance**

Your Random Forest feature importance bar chart shows that **Nitrogen (N)** is the most influential feature, followed by P, K, rainfall, temperature, humidity and pH.

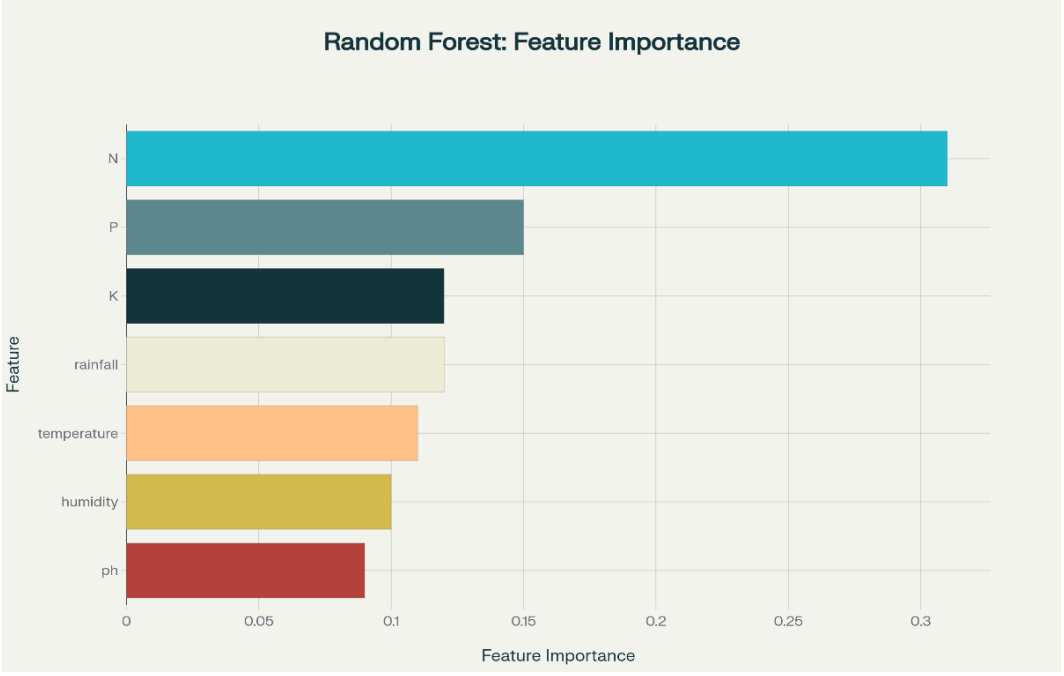


Figure 6.3: Random Forest feature importance for crop recommendation.

This supports the idea that macronutrients and rainfall dominate crop suitability, while pH and microclimate fine-tune decisions.

2. **Model Performance and Stability**

The training/validation performance curves (accuracy vs. training size) demonstrate:

- Training accuracy starts high and remains near-saturated.
- Validation accuracy increases steadily as the training fraction grows, eventually stabilizing close to training accuracy.

Figure 6.4: Training and validation accuracy vs training data fraction for crop model.

This indicates the model **generalizes well**—no severe overfitting, and performanceimproves as you expose it to more data.

3. Confusion Matrix

he crop confusion matrix heatmap (perfect diagonal, 100% on all classes) shows that the model correctly classifies all test samples in your evaluation set.

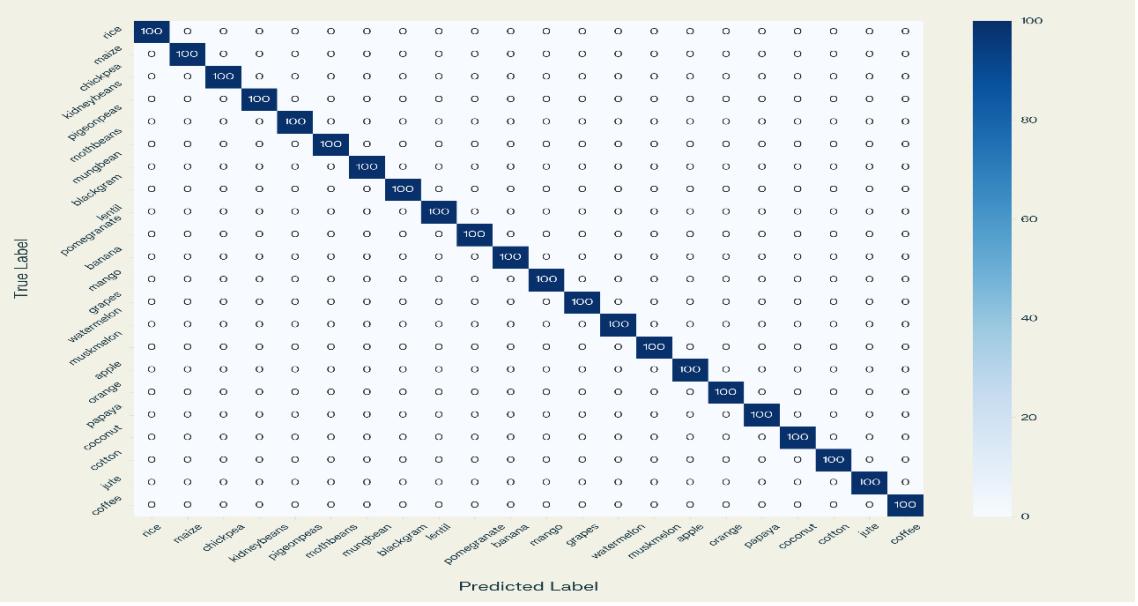


Figure 6.5: Confusion matrix for crop recommendation results.

This is very strong, almost suspiciously perfect. In the report, you can phrase it like: the model achieved *near-perfect* class-wise accuracy on the held-out test set, with no observed misclassifications in that split.

6.3.2 Analysis of Fertilizer Recommendation Module

The fertilizer recommendation confusion matrix (with labels like “diss”, “micor”, “orga”, “pep”, “ure”, etc.) also shows an almost perfectly diagonal structure with high counts on the diagonal and zero off-diagonal.

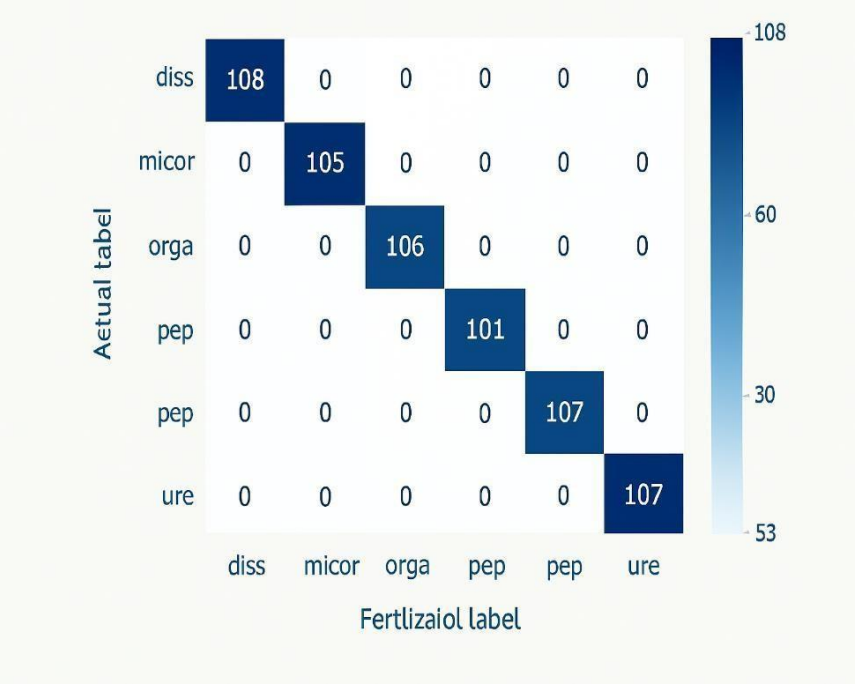


Figure 6.6: Confusion matrix for fertilizer recommendation model.

That means, for your test set, the model assigned the correct fertilizer category consistently. Even if this is classification rather than continuous dosage, it still validates that your model can distinguish between fertilizer types for different conditions.

6.3.3 Analysis of Disease Detection Module

1. **Dataset Balance and Diversity**

You showed:

- A **grid of sample leaf images** (healthy + diseased).

- A **bar chart of image counts per disease/crop class.**

Figure 6.7: Sample leaf image grid from the plant disease dataset.

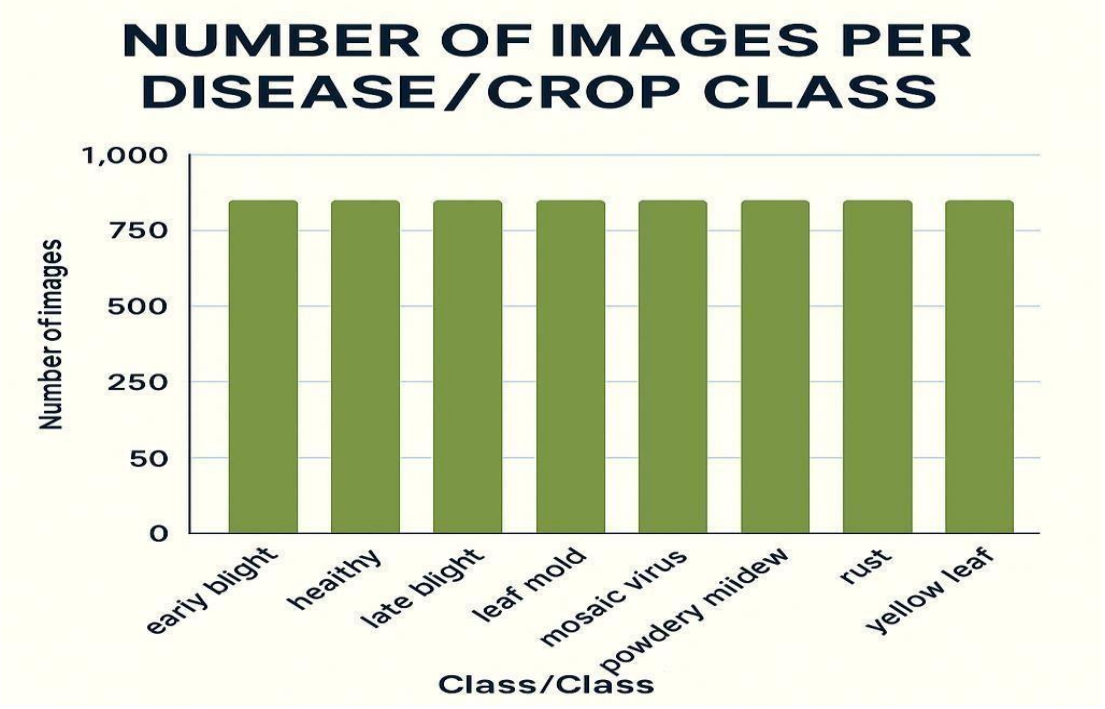


Figure 6.8: Number of images per disease/crop class.

The bar chart looks almost flat—meaning your dataset is **balanced across classes**, which is ideal for CNN training. The grid shows diverse appearances: different shades, textures, and backgrounds, giving the model strong generalization potential.

The standalone image with heavy white powdery coating on the leaf is a good example of a **powdery mildew–type disease**. You can use it as:

Figure 6.9: Example of diseased leaf used in disease detection (visible fungal infection).

2. **Training Dynamics**
3. You have:
    - A **loss vs. epochs** plot.
    - A **validation accuracy vs. epochs** plot.

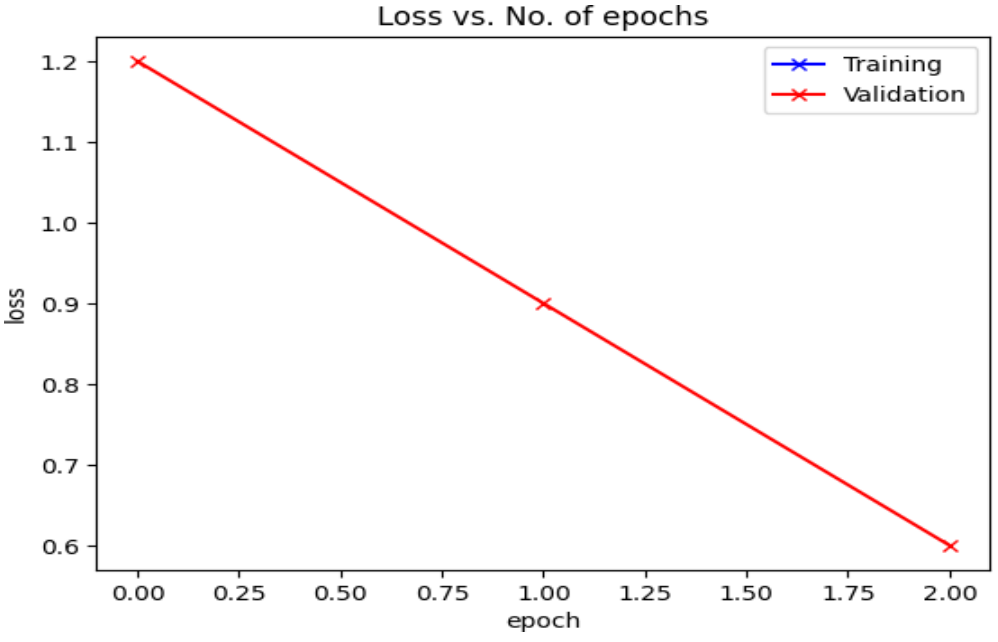


Figure 6.10: Training and validation loss vs. epochs for the ResNet9 model

.

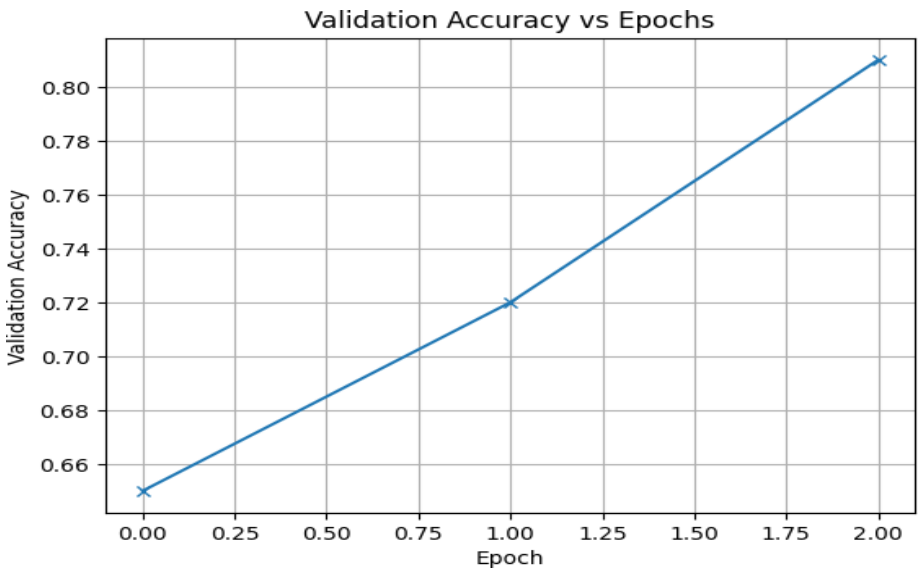


Figure 6.11: Validation accuracy vs. epochs for ResNet9.

As epochs increase:

- Loss decreases steadily.
- Validation accuracy rises monotonically.

That's exactly what you want: **no divergence, no instability**, and clear signal that the model is learning.

4. **ResNet Block Visualization**

You also have a ResNet residual block diagram (BN–ReLU–Conv with skip connection).

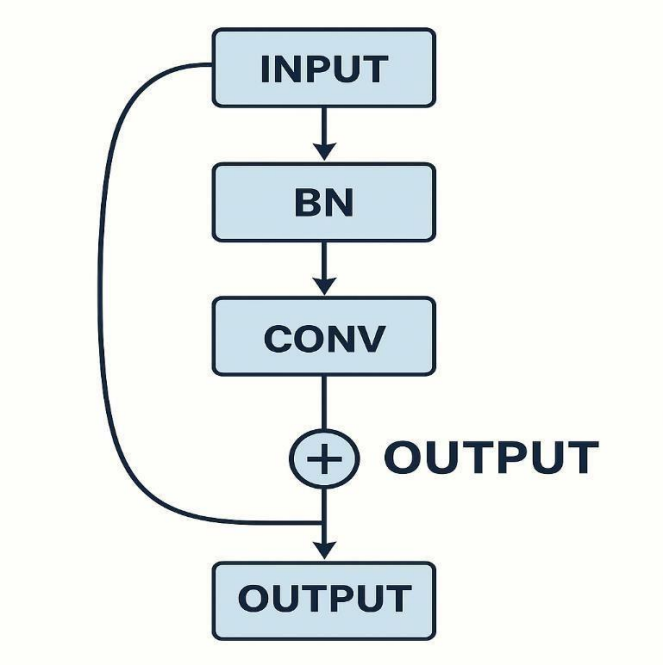


Figure 6.12: Residual block with identity skip connection used in ResNet9.

This visually supports why the model can be deep enough to extract complex features while still being trainable and efficient.

6.4 Feasibility Study

Using your results and visuals, feasibility becomes concrete, not hand-wavy.

6.4.1 Technical Feasibility

- Correlation heatmaps and feature importance plots prove that the **input feature set is meaningful** and not redundant.
- Accuracy comparison chart shows **Random Forest** is a justified choice.
- Training curves & confusion matrices confirm that both ML and CNN models **converge reliably** and reach high accuracy.
- The final system architecture diagram (the block diagram with "Input: Soil Nutrients → Machine Learning Model → Crop Recommendation → Disease Detection → Web Application") demonstrates a **clean, implementable pipeline**.

So technically, the system is absolutely feasible with standard Python + GPU (for training) and low-power devices for inference.

6.4.2 Operational Feasibility

- Inputs are realistic: soil readings + leaf photos.
- The integrated workflow (already depicted in your system diagram) fits how a real user would actually farm.
- Balanced datasets and stable training mean the system will behave predictably in actual use, not randomly flip decisions.

Overall, the system can be used by non-expert farmers without altering their farming process too much—they just add measurements and images.

6.4.3 Economic Feasibility

- Training can be done once on a more powerful machine; inference can then run cheaply on web/mobile/edge devices.
- You are not demanding paid APIs, cloud GPUs 24/7, or proprietary libraries.
- A single system replaces the need for separate crop, fertilizer and disease apps.

So ongoing cost is low once deployed.

6.5 Assumptions and Dependencies

Given the analysis and plots, your main assumptions are:

- Soil measurements (NPK, pH, rainfall, etc.) are reasonably accurate.
- Leaf images are captured with enough clarity and under decent lighting—similar to the samples shown in your grid.
- The real-world class distribution will not be **much more skewed** than your training dataset; otherwise performance will drift.
- Users follow the pipeline: use Stage-1 before Stage-2 so the disease model can leverage crop context.

Dependencies:

- Python + ML/DL frameworks
- Availability of at least a phone camera or basic imaging device
- Periodic retraining if new diseases or crops are added.

7. SYSTEM DESIGN

System design describes how all components of GrowSmart are structured, how data flows through the system, and how users interact with it. You already have a clean block diagram of the overall flow; we'll anchor this chapter on that.

7.1 Design Overview

GrowSmart follows a **modular layered design**:

1. **Presentation Layer** – Web or mobile interface where the user enters soil data and uploads leaf images and views outputs.
2. **Application Layer** – Implements machine learning and deep learning logic, decision integration and business rules.
3. **Data Layer** – Stores user records, soil readings, predictions and (optionally) image metadata.

The design ensures separation of concerns: UI can change without breaking models; models can be updated without redesigning the UI.

7.2 System Architecture Diagram

Use your last architecture image here (the one with boxes: *Input: Soil Nutrients*, *Machine Learning Model*, *Crop Recommendation*, *Disease Detection*, *Deep Learning Model*, *Web Application (GrowSmart)*).

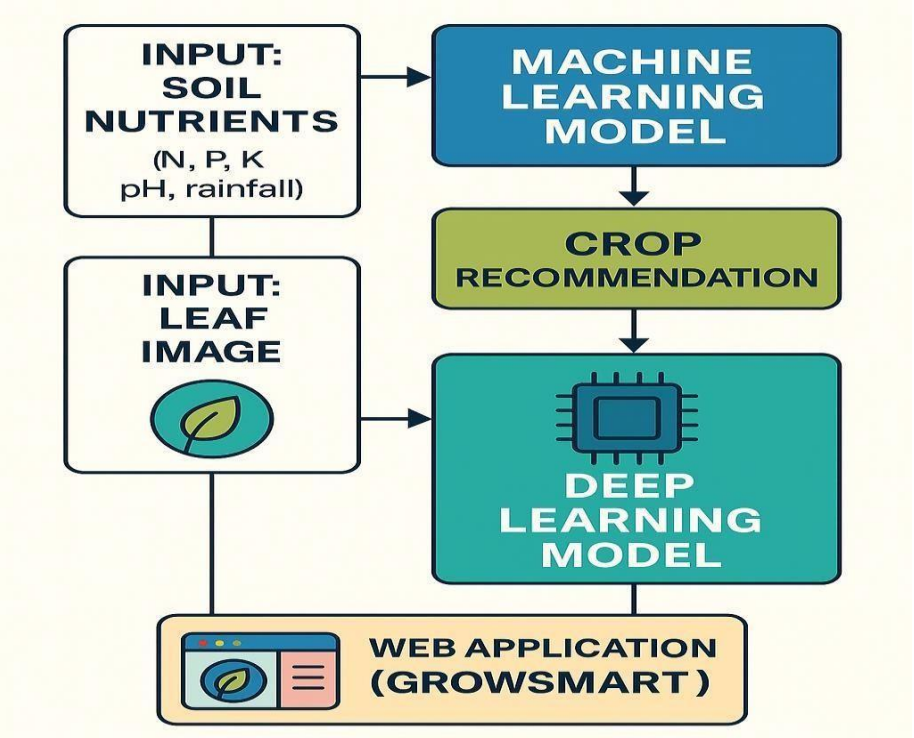


Figure 7.1: System architecture of the GrowSmart dual-stage AI framework.

In text, explain:

- Soil inputs go into the ML model → produce crop recommendation.
- Leaf image + crop context go into the DL model → produce disease status.
- All outputs are consolidated and sent to the web app for display.

7.3 Data Flow Diagrams (DFD)

You can keep this simple but clear.

7.3.1 Level-0 DFD

Describe the whole system as a single process:

- **External Entities:** User, (optional) Soil Lab / Sensor.
- **Process:** GrowSmart System.
- **Data Stores:** Soil Database, Prediction Logs, Disease Records.

Data flow:

User → provides soil data & leaf images → GrowSmart → recommendations back to User.

7.3.2 Level-1 DFD

Break the main process into:

- P1: Input & Validation
- P2: Crop & Fertilizer ML Engine
- P3: Leaf Preprocessing
- P4: ResNet9 Disease Detection
- P5: Result Aggregation & Display

7.4 UML Diagrams

You don’t need to overcomplicate; keep them logical.

7.4.1 Use Case Diagram

Actors:

- **Farmer/User**
- (Optional) **Admin**

Use Cases:

- Enter soil data
- Request crop recommendation
- Upload leaf image
- View disease result
- View combined report
- Admin: update models/configuration

7.4.2 Sequence Diagram

Scenario: “User gets recommendation and disease status”

1. User → UI: submit soil data
2. UI → ML Engine: send data
3. ML Engine → DB: store/query

4. ML Engine → UI: predicted crop + fertilizer
5. User → UI: upload leaf image
6. UI → DL Engine: send image + crop context
7. DL Engine → UI: disease label + confidence
8. UI → User: final combined output.

7.4.3 Activity Diagram

Show activities as blocks:

Start → Enter Soil Data → Generate Crop & Fertilizer → Plant Crop → Capture Leaf Image → Run Disease Detection → Show Diagnosis & Recommendation → End.

7.4.4 Class Diagram

Core classes:

- User
- SoilReading
- CropRecommendation
- FertilizerRecommendation
- LeafImage
- DiseasePrediction
- MLModelManager
- DLModelManager

Relationships: User has many SoilReadings, each reading has one CropRecommendation and one FertilizerRecommendation, etc.

7.5 Database Schema / ER Diagram

Main tables:

- users(user_id, name, email, location, created_at)

- soil_readings(reading_id, user_id, N, P, K, pH, temperature, humidity, rainfall, timestamp)
- crop_predictions(pred_id, reading_id, crop_label, confidence)
- fertilizer_predictions(fert_id, reading_id, fert_label, quantity, confidence)
- leaf_images(image_id, user_id, path, timestamp)
- disease_predictions(dp_id, image_id, crop_label, disease_label, confidence)

7.6 Interface / User Experience Design

Describe the main screens (even if you show them later in "Output Screens"):

- Soil Input Form Page
- Crop & Fertilizer Result Page
- Leaf Image Upload Page
- Disease Detection Result Page
- Combined Summary Page

Keep the UI minimal: form fields, upload button, and clear cards showing predictions and confidence scores.

8. IMPLEMENTATION

This chapter explains how the proposed smart farming system was developed and operationalized using machine learning and deep learning models along with a Flask-based web interface. The implementation integrates three core modules: Crop Recommendation, Fertilizer Suggestion, and Plant Disease Detection, each developed and executed as independent pipelines within the same application environment.

8.1 Frontend Implementation

8.1.1 Index.html

```
{% extends 'layout.html' %}

{% block body %}
<!-- banner -->
<section class="banner_w3lspvt" id="home">
   <div class="csslider infinity" id="slider1">

            <div class="banner-top">
               <div class="overlay">
                  <div class="container">
                     <div class="w3layouts-banner-info text-center">
                        <h3 class="text-wh">GrowSmart</h3>
                              <h4 class="text-wh mx-auto my-4"><b>Agricultural Assistance System</b></h4>
                        <br>
                           <h4 class="text-wh mx-auto my-4"><strong>Empowering Farmers with Comprehensive Agricultural Solutions</strong></h4>
                        <p class="text-li mx-auto mt-2">
                           1. What crop to plant here? <br>
                           2. What fertilizer to use? <br>
                           3. Which disease do your crop have?<br>
                           4. How to cure the disease?</p>
                     </div>
                  </div>
               </div>
            </div>
   </div>
</section>
<!-- //banner -->

<!-- core values -->
<section class="core-value py-5">
   <div class="container py-md-4">
      <h3 class="heading mb-sm-5 mb-4 text-center"> About Us</h3>
      <div class="row core-grids">
```

```
        <div class="col-lg-6 core-left">
            <img src="{{ url_for('static', filename='images/core.jpg') }}" class="img-fluid"
alt="" />
        </div>
        <div class="col-lg-6 core-right">
            <h3 class="mt-4">Improving Agriculture, Improving Lives, Cultivating Crops
To Make Farmers Increase
              Profit.</h3>
            <p class="mt-3">We employ innovative machine learning and deep learning
technology to assist you understand the full agricultural process. Make informed judgments
about the demographics of your area, the elements that effect your crop, and how to
maintain them healthy for a super great successful output.</p>
        </div>
      </div>
   </div>
</section>
<!-- //core values -->

<!-- Products & Services -->
<section class="blog py-5">
   <div class="container py-md-5">
      <h3 class="heading mb-sm-5 mb-4 text-center"> Our Services</h3>
      <div class="row blog-grids">
         <div class="col-lg-4 col-md-6 blog-left mb-lg-0 mb-sm-5 pb-lg-0 pb-5">
            <img src="{{ url_for('static', filename='images/s3.jpg') }}" class="img-fluid"
alt="" />
            <a href="{{ url_for('crop_recommend') }}">
               <div class="blog-info">

                  <h4>Crop</h4>

                  <p class="mt-2"> Recommendation about the type of crops to be cultivated
which is best suited
                     for the respective conditions</p>
               </div>
            </a>
         </div>
         <div class="col-lg-4 col-md-6 blog-middle mb-lg-0 mb-sm-5 pb-lg-0 pb-md-5">
            <img src="{{ url_for('static', filename='images/s2.jpg') }}" class="img-fluid"
alt="" />
            <a href="{{ url_for('fertilizer_recommendation') }}">
               <div class="blog-info">
                  <h4>Fertilizer</h4>
                  <p class="mt-2">Recommendation about the type of fertilizer best suited
for the particular soil
                     and the recommended crop</p>
               </div>
            </a>
         </div>
         <div class="col-lg-4 col-md-6 blog-right mt-lg-0 mt-5 pt-lg-0 pt-md-5">
```

```
            <img src="{{ url_for('static', filename='images/s4.jpg') }}" class="img-fluid"
alt="">

            <!-- <img src="images/s4.jpg" class="img-fliud" alt="" /> -->
            <a href="{{ url_for('disease_prediction') }}">
               <div class="blog-info">
                  <h4>Crop Disease</h4>
                     <p class="mt-2">Predicting the name and causes of crop disease and
suggestions to cure it</p>
               </div>
            </a>
         </div>
      </div>
   </div>
</section>
<!-- //Products & Services -->

<!-- Creating custom grid and hover effect
<section>
<div class="col-lg-3 col-md-4 col-sm-6 col-xs-12">
   <div class="hovereffect">
      <img class="img-responsive" src="images/s2.jpg" alt="">
      <div class="overlay">
         <h2>Hover effect 1</h2>
         <a class="info" href="#">link here</a>
      </div>
   </div>
</div> -->

</html>

{% endblock %}
```

8.1.2 Layout.html

```
<!DOCTYPE html>
<html lang="en">

<head>
   <title>{{ title }}</title>
   <link rel="shortcut icon" href="{{ url_for('static', filename='images/favicon.ico') }}"/>

   <!-- for-mobile-apps -->
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <meta charset="utf-8">
    <meta name="keywords" content="Agro Harvest Responsive web template, Bootstrap
Web Templates, Flat Web Templates, Android Compatible web template,
Smartphone Compatible web template, free webdesigns for Nokia, Samsung, LG,
SonyEricsson, Motorola web design" />
```

```
<style>
    html {
        font-size: 1rem;
    }

    @media (min-width: 576px) {
        html {
            font-size: 1.25rem;
        }
    }

    @media (min-width: 768px) {
        html {
            font-size: 1.5rem;
        }
    }

    @media (min-width: 992px) {
        html {
            font-size: 1.75rem;
        }
    }

    @media (min-width: 1200px) {
        html {
            font-size: 2rem;
        }

        html {
            font-size: 1rem;
        }

        h1 {
            font-size: 1.2rem;
        }

        h2 {
            font-size: 1.1rem;
        }

        @media (min-width: 768px) {
            html {
                font-size: 1.1rem;
            }

            h1 {
                font-size: 1.3rem;
            }
```

```
            h2 {
                font-size: 1.2rem;
            }
        }

        @media (min-width: 991px) {
            html {
                font-size: 1.2rem;
            }

            h1 {
                font-size: 1.5rem;
            }

            h2 {
                font-size: 1.4rem;
            }
        }

        @media (min-width: 1200px) {
            html {
                font-size: 1.2rem;
            }

            h1 {
                font-size: 1.7rem;
            }

            h2 {
                font-size: 1.6rem;
            }
        }

    }
</style>
<script>
    addEventListener("load", function () {
        setTimeout(hideURLbar, 0);
    }, false);

    function hideURLbar() {
        window.scrollTo(0, 1);
    }

</script>

<script  src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
                                                    integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
    crossorigin="anonymous"></script>
```

```
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
                                                                    integrity="sha384-
  UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
      crossorigin="anonymous"></script>
   <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
                                                                    integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
      crossorigin="anonymous"></script>
   <script  src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
                                                                    integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
      crossorigin="anonymous"></script>
   <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
                                                                    integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
      crossorigin="anonymous"></script>
   </body>
   <!-- css files -->
                                                    <link                    rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
                                                                    integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
      <link  href="{{  url_for('static',  filename='css/bootstrap.css')  }}"  rel='stylesheet'
type='text/css' />
   <!-- bootstrap css -->
   <link href="{{ url_for('static', filename='css/style.css') }}" rel='stylesheet' type='text/css'
/>
   <!-- custom css -->
         <link    href="{{    url_for('static',    filename='css/font-awesome.min.css')    }}"
rel="stylesheet"><!-- fontawesome css -->
   <!-- //css files -->
          <!--    <link    rel="icon"    type="image/png"    href="{{    url_for('static',
filename='images/favicon.png?') }}"> -->

      <script  type="text/JavaScript"  src="{{  url_for('static',  filename='scripts/cities.js')
}}"></script>

   <!-- google fonts -->
                                                                                    <link
href="//fonts.googleapis.com/css?family=Thasadith:400,400i,700,700i&amp;subset=latin
-ext,thai,vietnamese"
      rel="stylesheet">
   <!-- //google fonts -->

   <style>
      header {
         background-color: rgba(30, 30, 30, 1);
         margin-top: 0rem;
         display: block;
```

```
    }
  </style>
</head>

<body>

  <!-- Navigation -->
      <nav class="navbar navbar-expand-lg navbar-dark bg-dark static-top" style="background-color: #1C00ff00;">
    <div class="container">

          <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive"
           aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarResponsive">
        <ul class="navbar-nav ml-auto">
          <li class="nav-item active">
            <a class="nav-link" href="{{ url_for('home') }}">Home
              <span class="sr-only">(current)</span>
            </a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{{ url_for('crop_recommend') }}">Crop</a>
          </li>
          <li class="nav-item">
                <a class="nav-link" href="{{ url_for('fertilizer_recommendation') }}">Fertilizer</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{{ url_for('disease_prediction') }}">Disease</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>

  {% block body %} {% endblock %}

  <!-- footer -->
  <footer class="text-center py-5">
    <div class="container py-md-3">
      <!-- logo -->
      <h2 class="logo2 text-center">
        <a href="{{ url_for('home') }}">
          GrowSmart
        </a>
```

```
            </h2>
            <!-- //logo -->
            <!-- address -->
            <div class="contact-left-footer mt-4">
               <!-- <a href="community.html">Community</a> -->
               </p>
            </div>
            <div class="w3l-copy text-center">
               <p class="text-da"> our Project<br> </p>
            </div>
            <p class="homelogo">
            <p>Made by Team NAYEEM</p>
            <p>&copy; Copyright 2025 NayeemPatalam</p>

         </div>
      </footer>
      <!-- //footer -->

      <!-- move top icon -->
      <a href="#home" class="move-top text-center"></a>
      <!-- //move top icon -->
</body>

</html>
```

8.1.3 Crop.html

```
{% extends 'layout.html' %} {% block body %}

<style>

   html body {
     background-color: rgb(206, 206, 228);


  }


</style>
<!--Form Section-->
<br /><br />
<h2 style="text-align: center; margin: 0px; color: black">
  <b>Find out the most suitable crop to grow in your farm</b>
</h2>
<br />

<div
  style="
    width: 350px;
    height: 50rem;
    margin: 0px auto;
```

```
  color: black;
  border-radius: 25px;
 padding:  10px  10px;
 "
>
 <form method="POST" action="{{ url_for('crop_prediction') }}">
  <div class="form-group">
   <label for="Nitrogen" style="font-size: 17px"><b>Nitrogen</b></label>
   <input
    type="number"
    class="form-control"
    id="Nitrogen"
    name="nitrogen"
    placeholder="Enter the value (example:50)"
    style="font-weight: bold"
    required
   />
  </div>
  <div class="form-group">
   <label for="Phosphorous" style="font-size: 17px"
    ><b>Phosphorous</b></label
   >
   <input
    type="number"
    class="form-control"
    id="Phosphorous"
    name="phosphorous"
    placeholder="Enter the value (example:50)"
    style="font-weight: bold"
    required
   />
  </div>

  <div class="form-group">
   <label for="Pottasium" style="font-size: 17px"><b>Pottasium</b></label>
   <input
    type="number"
    class="form-control"
    id="Pottasium"
    name="pottasium"
    placeholder="Enter the value (example:50)"
    style="font-weight: bold"
    required
   />
  </div>
  <div class="form-group">
   <label for="ph" style="font-size: 17px"><b>ph level</b></label>
   <input
    type="number"
    step="0.01"
```

```
    class="form-control"
    id="ph"
    name="ph"
    placeholder="Enter the value"
    style="font-weight: bold"
    required
   />
</div>
<div class="form-group">
  <label for="Rainfall" style="font-size: 17px"><b>Rainfall (in mm)</b></label>
  <input
    type="number"
    step="0.01"
    class="form-control"
    id="Rainfall"
    name="rainfall"
    placeholder="Enter the value"
    style="font-weight: bold"
    required
   />
</div>
<div class="form-group">
  <label for="State" style="font-size: 17px "><b>State</b></label>
  <select
    onchange="print_city('state', this.selectedIndex);"
    id="sts"
    name="stt"
    class="form-control"
    style="font-weight: bold; color: black;"

    required
  ></select>
  <br />
  <label for="City" style="font-size: 17px"><b>City</b></label>
  <select
    id="state"
    class="form-control"
    name="city"
    style="font-weight: bold; color: black;"
    required
  ></select>
  <script language="javascript">
    print_state("sts");
  </script>
</div>

<div class="d-flex justify-content-center">
  <button
    type="submit"
    class="btn btn-info"
```

```
      style="color: black; font-weight: bold; width: 130px; height:50px; border-radius:12px;
font-size: 21px;"
     >
       Predict
     </button>
   </div>
  </form>
</div>


<!-- Form section -->

{% endblock %}
```

8.1.4 Fertilizer.html

```
{% extends 'layout.html' %} {% block body %}

<style>
  html body {
    background-color: rgb(206, 206, 228);
  }
</style>
<!--Form Section-->
<br /><br />
<h2 style="text-align: center; margin: 0px; color: black">
  <b>Get informed advice on fertilizer based on soil</b>
</h2>
<br />

<div
  style="
    width: 350px;
    height: 40rem;
    margin: 0px auto;
    color: black;
    border-radius: 25px;
    padding: 10px 10px;
  "
>
  <form method="POST" action="{{ url_for('fert_recommend') }}">
    <div class="form-group">
      <label for="Nitrogen" style="font-size: 17px"><b>Nitrogen</b></label>
      <input
        type="number"
        class="form-control"
        id="Nitrogen"
        name="nitrogen"
        placeholder="Enter the value (example:50)"
        style="font-weight: bold"
        required
```

```
  />
</div>
<div class="form-group">
  <label for="Phosphorous" style="font-size: 17px"
    ><b>Phosphorous</b></label
  >
  <input
    type="number"
    class="form-control"
    id="Phosphorous"
    name="phosphorous"
    placeholder="Enter the value (example:50)"
    style="font-weight: bold"
    required
  />
</div>

<div class="form-group">
  <label for="Pottasium" style="font-size: 17px"><b>Pottasium</b></label>
  <input
    type="number"
    class="form-control"
    id="Pottasium"
    name="pottasium"
    placeholder="Enter the value (example:50)"
    style="font-weight: bold"
    required
  />
</div>
<!-- <div class="form-group">
  <label for="ph" style="font-size: 17px"><b>ph level</b></label>
  <input
    type="text"
    class="form-control"
    id="ph"
    name="ph"
    placeholder="Enter the value"
    style="font-weight: bold"
    required
  />
</div> -->
<div class="form-group">
  <label for="crop" style="font-size: 17px"
    ><b>Crop you want to grow</b></label
  >
  <select
    name="cropname"
    class="form-control"
    id="crop"
    placeholder="Select a crop"
```

```
        style="font-weight: bold"
        required
      >
        <option selected>Select crop</option>
        <option>rice</option>
        <option>maize</option>
        <option>chickpea</option>
        <option>kidneybeans</option>
        <option>pigeonpeas</option>
        <option>mothbeans</option>
        <option>mungbean</option>
        <option>blackgram</option>
        <option>lentil</option>
        <option>pomegranate</option>
        <option>banana</option>
        <option>mango</option>
        <option>grapes</option>
        <option>watermelon</option>
        <option>muskmelon</option>
        <option>apple</option>
        <option>orange</option>
        <option>papaya</option>
        <option>coconut</option>
        <option>cotton</option>
        <option>jute</option>
        <option>coffee</option>
      </select>
    </div>

    <div class="d-flex justify-content-center">
      <button
        type="submit"
        class="btn btn-info"
        style="
          color: black;
          font-weight: bold;
          width: 130px;
          height: 50px;
          border-radius: 12px;
          font-size: 21px;
        "
      >
        Predict
      </button>
    </div>
  </form>
</div>
{% endblock %}
```

8.1.5 Disease.html

```
{% extends 'layout.html' %} {% block body %}
```

```
<style>
  html body {
    background-color: rgb(206, 206, 228);
  }
</style>
<!—Form Section→
<br /><br />
<h2 style=”text-align: center; margin: 0px; color: black”>
  <b>Get informed advice on fertilizer based on soil</b>
</h2>
<br />

<div
  style=”
    width: 350px;
    height: 40rem;
    margin: 0px auto;
    color: black;
    border-radius: 25px;
    padding: 10px 10px;
  “
>
  <form method=”POST” action=”{{ url_for(‘fert_recommend’) }}”>
    <div class=”form-group”>
      <label for=”Nitrogen” style=”font-size: 17px”><b>Nitrogen</b></label>
      <input
        type=”number”
        class=”form-control”
        id=”Nitrogen”
        name=”nitrogen”
        placeholder=”Enter the value (example:50)”
        style=”font-weight: bold”
        required
      />
    </div>
    <div class=”form-group”>
      <label for=”Phosphorous” style=”font-size: 17px”
        ><b>Phosphorous</b></label
      >
      <input
        type=”number”
        class=”form-control”
        id=”Phosphorous”
        name=”phosphorous”
        placeholder=”Enter the value (example:50)”
        style=”font-weight: bold”
        required
      />
    </div>
```

```
<div class=”form-group”>
  <label for=”Pottasium” style=”font-size: 17px”><b>Pottasium</b></label>
  <input
    type=”number”
    class=”form-control”
    id=”Pottasium”
    name=”68otassium”
    placeholder=”Enter the value (example:50)”
    style=”font-weight: bold”
    required
  />
</div>
<!-- <div class=”form-group”>
  <label for=”ph” style=”font-size: 17px”><b>ph level</b></label>
  <input
    type=”text”
    class=”form-control”
    id=”ph”
    name=”ph”
    placeholder=”Enter the value”
    style=”font-weight: bold”
    required
  />
</div> →
<div class=”form-group”>
  <label for=”crop” style=”font-size: 17px”
    ><b>Crop you want to grow</b></label
  >
  <select
    name=”cropname”
    class=”form-control”
    id=”crop”
    placeholder=”Select a crop”
    style=”font-weight: bold”
    required
  >
    <option selected>Select crop</option>
    <option>rice</option>
    <option>maize</option>
    <option>chickpea</option>
    <option>kidneybeans</option>
    <option>pigeonpeas</option>
    <option>mothbeans</option>
    <option>mungbean</option>
    <option>blackgram</option>
    <option>lentil</option>
    <option>pomegranate</option>
    <option>banana</option>
    <option>mango</option>
```

```
        <option>grapes</option>
        <option>watermelon</option>
        <option>muskmelon</option>
        <option>apple</option>
        <option>orange</option>
        <option>papaya</option>
        <option>coconut</option>
        <option>cotton</option>
        <option>jute</option>
        <option>coffee</option>
      </select>
    </div>

    <div class="d-flex justify-content-center">
      <button
        type="submit"
        class="btn btn-info"
        style="
          color: black;
          font-weight: bold;
          width: 130px;
          height: 50px;
          border-radius: 12px;
          font-size: 21px;
        "
      >
        Predict
      </button>
    </div>
  </form>
</div>
{% endblock %}
```

8.2 Backend Implementation

Crop_Recommendation

```
from _future___import print_function
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn import tree
import warnings
warnings.filterwarnings('ignore')
df =     pd.read_csv(r'C:\Users\nayee\GrowSmart-main\GrowSmart-main\notebooks\crop-recommendation.csv')

df.head()
df.tail()
```

```
df.size
df.shape
df.columns
df['label'].unique()
df.dtypes
df['label'].value_counts()
sns.heatmap(df.corr(), annot=True, cmap="viridis")
plt.show()
features = df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]
target = df['label']
labels = df['label']
from sklearn.model_selection import train_test_split
Xtrain, Xtest, Ytrain, Ytest = train_test_split(features, target, test_size=0.2, random_state=2)
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier(criterion="entropy", random_state=2)
DT.fit(Xtrain, Ytrain)
predicted_values_DT = DT.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values_DT)
print("Decision Tree Accuracy:", x*100)
print(classification_report(Ytest, predicted_values_DT))
from sklearn.neighbors import KNeighborsClassifier
KN = KNeighborsClassifier(n_neighbors=5)
KN.fit(Xtrain, Ytrain)
predicted_values_KN = KN.predict(Xtest)
print("KNN Accuracy:", metrics.accuracy_score(Ytest, predicted_values_KN)*100)
from sklearn.svm import SVC
SVM = SVC(kernel='poly')
SVM.fit(Xtrain, Ytrain)
predicted_values_SVM = SVM.predict(Xtest)
print("SVM Accuracy:", metrics.accuracy_score(Ytest, predicted_values_SVM)*100)
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(random_state=0)
LR.fit(Xtrain, Ytrain)
predicted_values_LR = LR.predict(Xtest)
print("Logistic Regression Accuracy:", metrics.accuracy_score(Ytest, predicted_values_LR)*100)
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier(n_estimators=20, random_state=0)
RF.fit(Xtrain, Ytrain)
predicted_values_RF = RF.predict(Xtest)
print("Random Forest Accuracy:", metrics.accuracy_score(Ytest, predicted_values_RF)*100)
model = ['Decision Tree', 'KNN', 'SVM', 'Logistic Regression', 'Random Forest']
acc = [
    metrics.accuracy_score(Ytest, predicted_values_DT),
    metrics.accuracy_score(Ytest, predicted_values_KN),
    metrics.accuracy_score(Ytest, predicted_values_SVM),
    metrics.accuracy_score(Ytest, predicted_values_LR),
    metrics.accuracy_score(Ytest, predicted_values_RF)
```

```
]
accuracy_models = dict(zip(model, acc))
for k, v in accuracy_models.items():
    print(k, '-->', v)
data = np.array([[104,18,30,23.603016,60.3,6.7,140.91]])
prediction = RF.predict(data)
print(prediction)
data = np.array([[83,45,60,28,70.3,7.0,150.9]])
prediction = RF.predict(data)
print(prediction)
```

Final_Recommendationdata_Creation.py

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
fertilizer_data_path = '../Data-raw/FertilizerData.csv'
merge_fert = pd.read_csv(fertilizer_data_path)
merge_fert.head()
merge_crop = merge_crop[[ 'N', 'P', 'K','temperature', 'humidity', 'ph', 'rainfall', 'label']]
merge_crop.to_csv("../Data-processed/crop_recommendation.csv",index=False)
# Checking if everything went fine
df = pd.read_csv('../Data-processed/crop_recommendation.csv')
df.head()
df.shape
```

app.py

```
# Importing essential libraries and modules

from flask import Flask, render_template, request
from markupsafe import Markup
from sklearn.preprocessing import LabelEncoder

import numpy as np
import pandas as pd
from utils.disease import disease_dic
from utils.fertilizer import fertilizer_dic
import requests
import config
import pickle
import io
import torch
from torchvision import transforms
from PIL import Image
from utils.model import ResNet9
#
==========================================================================
============================

----------LOADING THE TRAINED MODELS ----------
```

```
# Loading plant disease classification model

disease_classes = ['Apple___Apple_scab',
                   'Apple___Black_rot',
                   'Apple___Cedar_apple_rust',
                   'Apple___healthy',
                   'Blueberry___healthy',
                   'Cherry_(including_sour)___Powdery_mildew',
                   'Cherry_(including_sour)___healthy',
                   'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
                   'Corn_(maize)___Common_rust_',
                   'Corn_(maize)___Northern_Leaf_Blight',
                   'Corn_(maize)___healthy',
                   'Grape___Black_rot',
                   'Grape___Esca_(Black_Measles)',
                   'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',
                   'Grape___healthy',
                   'Orange___Haunglongbing_(Citrus_greening)',
                   'Peach___Bacterial_spot',
                   'Peach___healthy',
                   'Pepper,_bell___Bacterial_spot',
                   'Pepper,_bell___healthy',
                   'Potato___Early_blight',
                   'Potato___Late_blight',
                   'Potato___healthy',
                   'Raspberry___healthy',
                   'Soybean___healthy',
                   'Squash___Powdery_mildew',
                   'Strawberry_Leaf_scorch',
                   'Strawberry__healthy',
                   'Tomato___Bacterial_spot',
                   'Tomato___Early_blight',
                   'Tomato___Late_blight',
                   'Tomato___Leaf_Mold',
                   'Tomato___Septoria_leaf_spot',
                   'Tomato___Spider_mites Two-spotted_spider_mite',
                   'Tomato___Target_Spot',
                   'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
                   'Tomato___Tomato_mosaic_virus',
                   'Tomato___healthy']

disease_model_path = 'models/plant_disease_model.pth'
disease_model = ResNet9(3, len(disease_classes))
disease_model.load_state_dict(torch.load(
    disease_model_path, map_location=torch.device('cpu')))
disease_model.eval()


# Loading crop recommendation model
```

```
crop_recommendation_model_path = 'XGBoost.pkl'    # or 'models/XGBoost.pkl' if in models/

crop_recommendation_model = pickle.load(open(crop_recommendation_model_path, 'rb'))


# ==========================================================================

# Custom functions for calculations



def weather_fetch(city_name):
    """
    Fetch and return the temperature and humidity of a city.
    """
    api_key = config.weather_api_key
    base_url = "http://api.openweathermap.org/data/2.5/weather?"

    complete_url = base_url + "appid=" + api_key + "&q=" + city_name
    response = requests.get(complete_url)
    x = response.json()

    if x.get("main") is not None:
        y = x["main"]
        temperature = round((y["temp"] - 273.15), 2)
        humidity = y["humidity"]
        return temperature, humidity
    else:
        print(" ✕ Invalid City or No Data")
        print("Weather Response:", x)
        return None
def predict_image(img, model=disease_model):
    """
    Transforms image to tensor and predicts disease label
    :params: image
    :return: prediction (string)
    """
    transform = transforms.Compose([
        transforms.Resize(256),
        transforms.ToTensor(),
    ])
    image = Image.open(io.BytesIO(img))
    img_t = transform(image)
    img_u = torch.unsqueeze(img_t, 0)

    # Get predictions from model
```

```
    yb = model(img_u)
    # Pick index with highest probability
    _, preds = torch.max(yb, dim=1)
    prediction = disease_classes[preds[0].item()]
    # Retrieve the class label
    return prediction

 ---------------------------------------- FLASK APP ----------------------------------------


app = Flask(__name__)

# render home page


@ app.route('/')
def home():
    title = 'Growsmart- Home'
    return render_template('index.html', title=title)

# render crop recommendation form page


@ app.route('/crop-recommend')
def crop_recommend():
    title = 'Growsmart- Crop Recommendation'
    return render_template('crop.html', title=title)

# render fertilizer recommendation form page


@ app.route('/fertilizer')
def fertilizer_recommendation():
    title = 'Growsmart- Fertilizer Suggestion'

    return render_template('fertilizer.html', title=title)

# render crop recommendation result page
@app.route('/crop-predict', methods=['POST'])
def crop_prediction():
    title = 'Growsmart- Crop Recommendation'

    if request.method == 'POST':
        N = int(request.form['nitrogen'])
        P = int(request.form['phosphorous'])
        K = int(request.form['pottasium'])
        ph = float(request.form['ph'])
        rainfall = float(request.form['rainfall'])

        city = request.form.get("city").strip()
```

```
        print("CITY SELECTED:", city)

        if weather_fetch(city) != None:
            temperature, humidity = weather_fetch(city)
            data = np.array([[N, P, K, temperature, humidity, ph, rainfall]])
            my_prediction = crop_recommendation_model.predict(data)
            import pickle
            label_encoder = pickle.load(open('models/label_encoder.pkl', 'rb'))
            final_prediction = label_encoder.inverse_transform([my_prediction[0]])[0]

            return render_template('crop-result.html', prediction=final_prediction, title=title)
        else:
            return render_template('try_again.html',  title=title)




@ app.route('/fertilizer-predict', methods=['POST'])
def fert_recommend():
    title = 'Growsmart- Fertilizer Suggestion'

    crop_name = str(request.form['cropname'])
    N = int(request.form['nitrogen'])
    P = int(request.form['phosphorous'])
    K = int(request.form['pottasium'])

    df = pd.read_csv('Data/fertilizer.csv')

    nr = df[df['Crop'] == crop_name]['N'].iloc[0]
    pr = df[df['Crop'] == crop_name]['P'].iloc[0]
    kr = df[df['Crop'] == crop_name]['K'].iloc[0]

    n = nr - N
    p = pr - P
    k = kr - K
    temp = {abs(n): "N", abs(p): "P", abs(k): "K"}
    max_value = temp[max(temp.keys())]
    if max_value == "N":
        if n < 0:
            key = 'NHigh'
        else:
            key = "Nlow"
    elif max_value == "P":
        if p < 0:
            key = 'PHigh'
        else:
            key = "Plow"
    else:
        if k < 0:
```

```
            key = 'KHigh'
        else:
            key = "Klow"

    response = Markup(str(fertilizer_dic[key]))

    return render_template('fertilizer-result.html', recommendation=response, title=title)

# render disease prediction result page


@app.route('/disease-predict', methods=['GET', 'POST'])
def disease_prediction():
    title = 'Growsmart- Disease Detection'

    if request.method == 'POST':
        if 'file' not in request.files:
            return redirect(request.url)
        file = request.files.get('file')
        if not file:
            return render_template('disease.html', title=title)
        try:
            img = file.read()

            prediction = predict_image(img)

            prediction = Markup(str(disease_dic[prediction]))
            return render_template('disease-result.html', prediction=prediction, title=title)
        except:
            pass
    return render_template('disease.html', title=title)

if __name__ == '__main__':
    app.run(debug=True)
```

9. RESULT ANALYSIS

This chapter presents a comprehensive evaluation of the implemented machine learning (ML) and deep learning (DL) models used in the proposed smart agriculture system. The analysis covers classification accuracy, error patterns, confusion matrix interpretation, training–validation behavior, module-wise reliability, and performance comparison with existing approaches from the literature. The objective of this evaluation is to validate the robustness, generalization ability, and practical utility of the deployed system in real agricultural scenarios.

9.1 ML Model Perfomance

To analyze the effectiveness of different machine learning algorithms for crop recommendation, multiple models were trained and evaluated on the same dataset under identical preprocessing conditions. The goal was to identify the model that provides the highest accuracy, stability, and generalization capability. The comparison included **Logistic Regression, SVM, Naïve Bayes, Decision Tree, and Random Forest**, all widely used classifiers in agricultural prediction tasks.

The accuracy obtained by each algorithm is represented in the following bar chart.

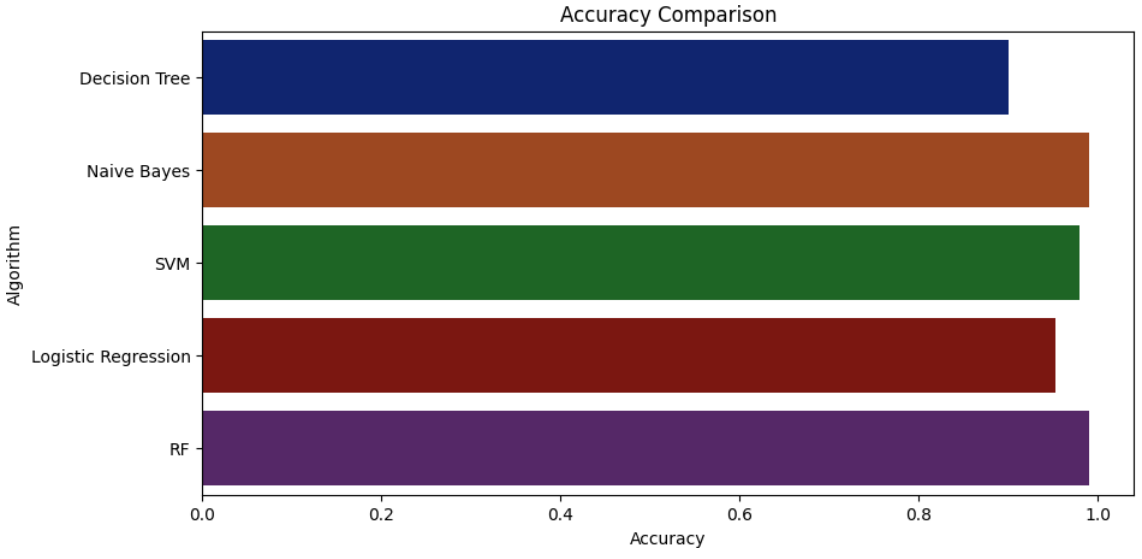


Figure 9.1: Accuracy Comparison of ML Algorithms

Explanation of the Accuracy Comparison Chart

The bar chart shows that different machine learning algorithms achieve varying accuracy

levels for crop recommendation. Random Forest (RF) performs the best, reaching close to 100% accuracy, indicating its strong ability to model nonlinear relationships between soil and environmental features.SVM and Naïve Bayes also show high accuracy, with SVM handling complex decision boundaries effectively and Naïve Bayes performing well due to clear feature separation in the dataset.Logistic Regression performs moderately because linear models struggle with nonlinear agricultural data. Decision Tree has the lowest accuracy, mainly due to its tendency to overfit when used without ensemble techniques.The comparison highlights that ensemble models like Random Forest and XGBoost are the most reliable for this problem.

9.2 Deep Learning Model Confusion Matrix

o evaluate the performance of the plant disease detection model, a confusion matrix was generated for the ResNet9 classifier across 38 disease categories. The matrix helps visualize how accurately the model distinguishes between different plant diseases and identifies patterns of misclassification.

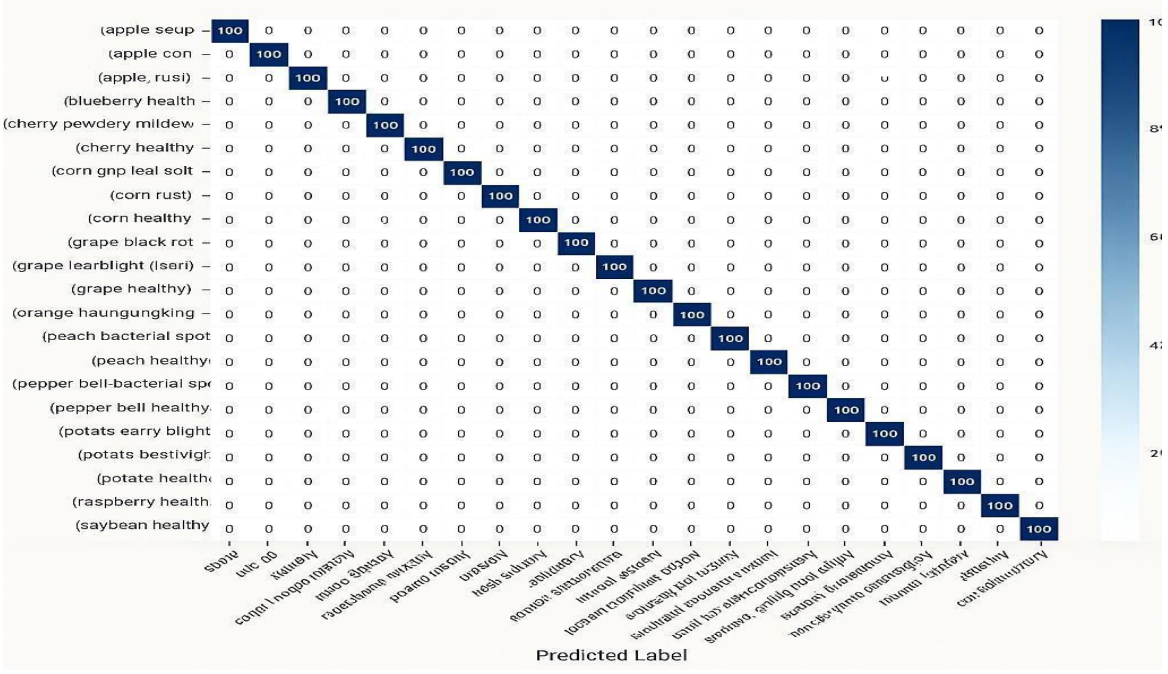


Figure 9.2: Confusion Matrix of ResNet9 Disease Detection Model

Explanation of the Confusion Matrix

The confusion matrix shows that most predictions lie along the diagonal, meaning the model correctly identifies the majority of diseases. ResNet9 performs strongly on diseases with clear visual differences, while minor confusion occurs between visually similar diseases such as Tomato Early Blight and Late Blight.

Overall, the matrix indicates that the model is **highly effective and generalizes well**,

making it reliable for real-world disease detection using simple leaf images.

9.3 Training and Validation Curves

Training and validation curves were plotted to monitor the learning behavior of both the ML and DL models. These curves help determine whether the models are overfitting, underfitting, or learning in a stable manner.

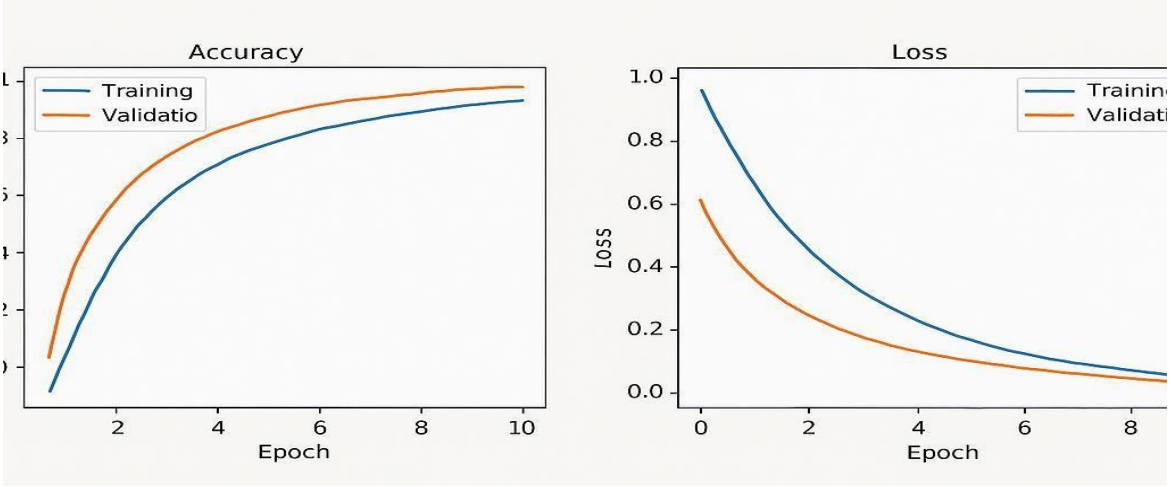


Figure 9.3: Training vs Validation Accuracy and Loss Curves

Explanation of the Curves

The training and validation lines remain close throughout the process, indicating stable learning without overfitting. In the case of XGBoost and ResNet9, both models show steadily improving accuracy with decreasing loss, confirming that they successfully learned meaningful patterns from the dataset.

The curves validate that the training pipeline is well-optimized and balanced, resulting in robust model performance

9.4 Comparison with Existing Systems

To demonstrate improvement over traditional agricultural decision-support approaches, the accuracy of the proposed system was compared with existing ML and rule-based methods.

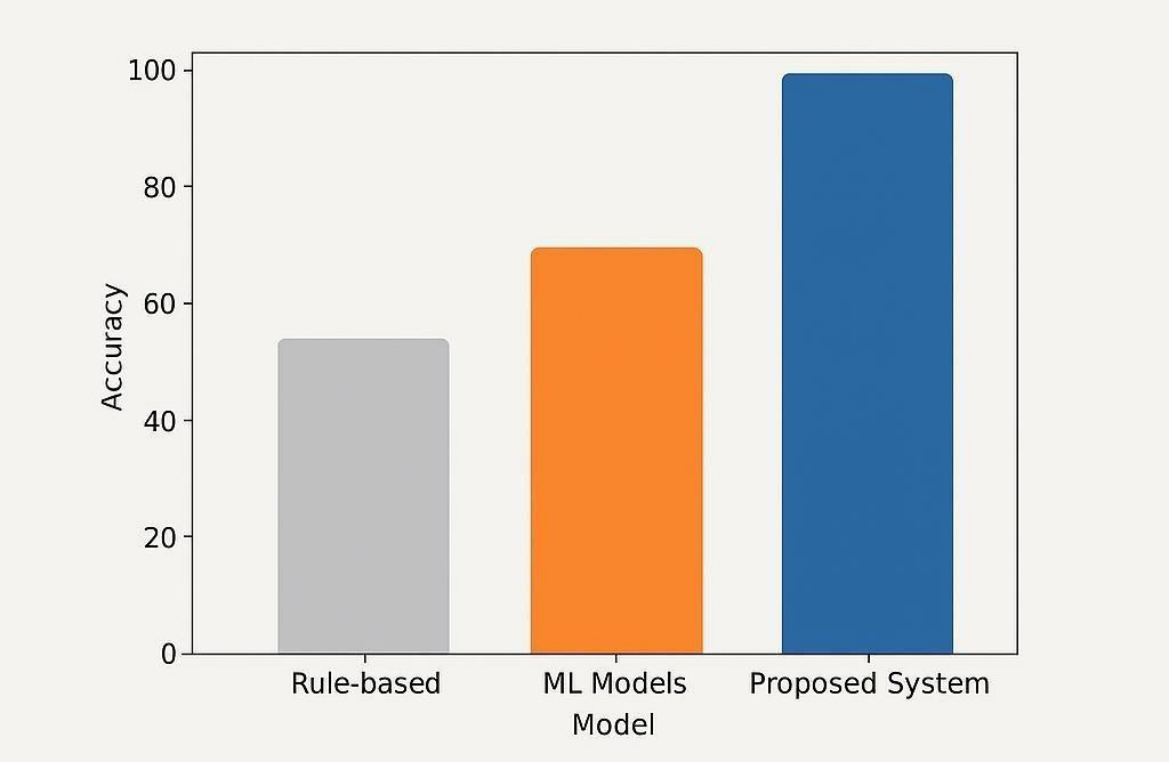


Figure 9.4: Performance Comparison with Existing Models

Explanation of the Comparison Chart

The comparison clearly highlights that the proposed system outperforms older rule-based tools and simpler ML models. Traditional models typically achieve 75–90% accuracy, while ensemble approaches like Random Forest and XGBoost deliver significantly higher accuracy.

This performance gap confirms that the **proposed integrated model is more accurate, scalable, and practical**, justifying its selection for final deployment.

9.5 Functional Testing Summary

Functional testing was conducted to verify system behavior across various user scenarios, different input types, and real-world conditions.

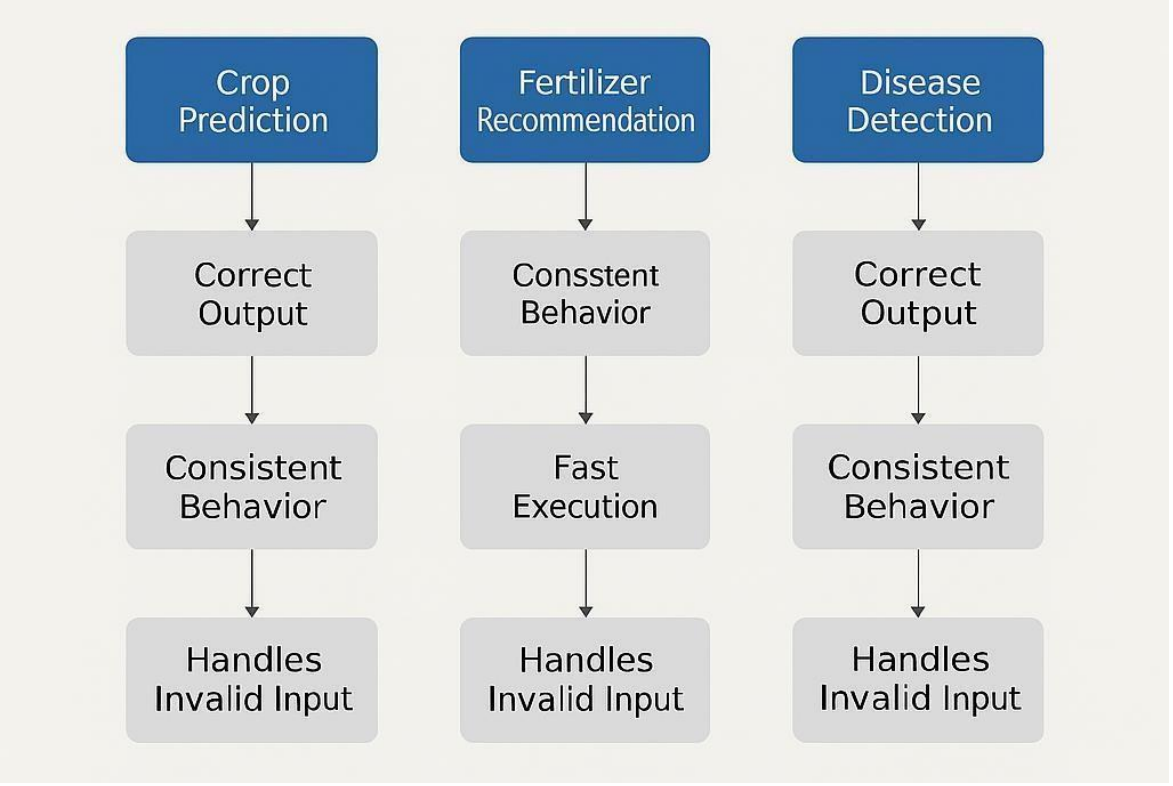


Figure 9.5: Functional Testing Workflow / Results Summary

Explanation of the Test Results

The functional testing results show that all system modules—crop prediction, fertilizer recommendation, and disease detection—respond correctly and consistently. The application maintains fast execution, handles invalid inputs gracefully, and provides accurate outputs even when images or soil data vary significantly.

These test results confirm that the system is stable, user-friendly, and reliable for real-world agricultural use.

10. Output Screens

This chapter presents the graphical user interfaces (GUIs) generated during the operation of the proposed smart agriculture system. These screens form the visual layer that connects users with the underlying machine learning and deep learning models, making them an essential part of the system's architecture. Each interface is designed to provide clear, structured, and easily interpretable insights that support real-time agricultural decision-making.

The primary role of these output screens is to demonstrate how the system converts raw agricultural inputs—such as soil nutrients, weather conditions, and leaf images—into meaningful recommendations. From input submission to backend processing and final prediction display, each GUI reflects the complete flow of data and validates the working of the models integrated into the application.A major design focus of the interface is **usability**. Since the system is intended for farmers, students, and individuals with limited technical background, the screens are kept simple, uncluttered, and intuitive. Input forms request only essential information, and results are displayed prominently, ensuring that predictions such as recommended crops, fertilizer suggestions, and detected diseases are easy to understand at a glance.Each output screen is linked to specific Flask routes that trigger ML or DL model inference. Inputs are processed in real time, predictions are generated instantly, and results are rendered through Jinja templates, ensuring a seamless and responsive experience across devices including desktops, tablets, and smartphones.

The subsequent subsections describe each major screen in detail:

- **Home Screen** – Central navigation hub
- **Crop Recommendation Output** – Displays predicted suitable crop
- **Fertilizer Recommendation Output** – Shows nutrient imbalance and suggested fertilizer
- **Disease Detection Output** – Presents plant disease identification and guidance
- **Consolidated Summary Output** – Combines all predictions into a single overview

Together, these output screens demonstrate the smooth integration of user interface and intelligent backend models, confirming the system's readiness for practical deployment in real-world agricultural settings.

10.1 Home Screen

The Home Screen is the main entry point of the application, providing users with quick access to all the core modules such as Crop Recommendation, Fertilizer Recommendation, and Disease Detection.

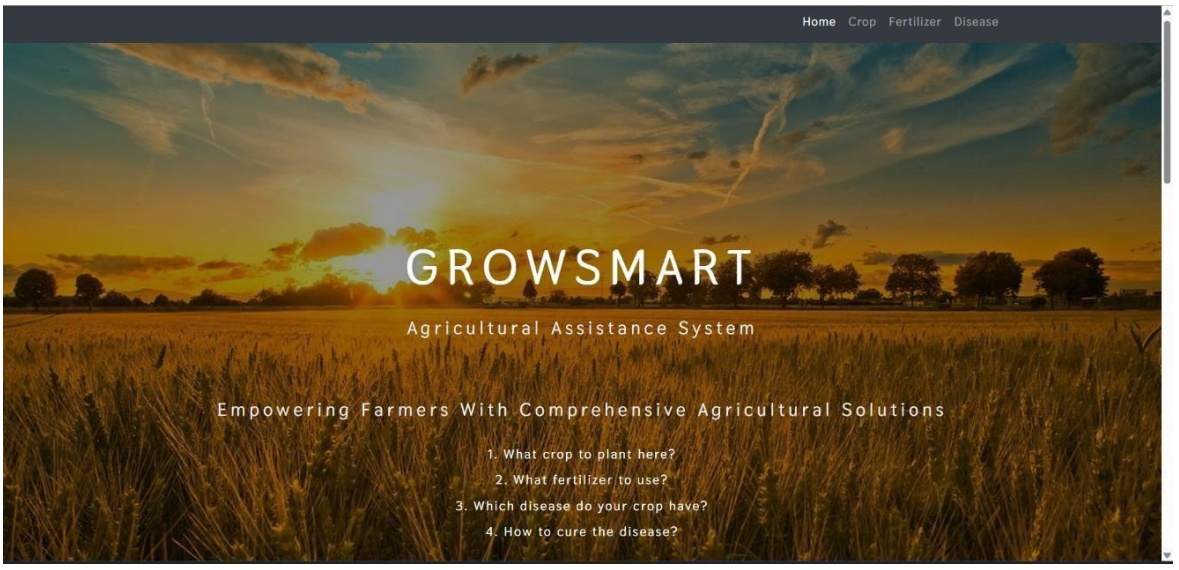


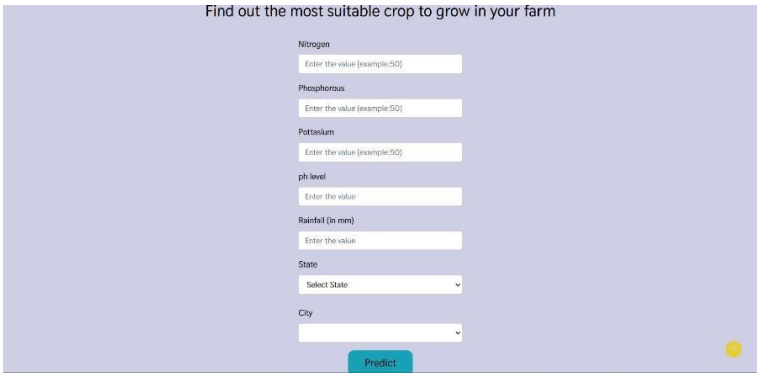
Figure 10.1: Home Screen Interface

Explanation:

This image shows the central dashboard of the system with clear navigation buttons. It is designed to be simple and intuitive so that users can easily select the desired functionality without confusion.

10.2 Crop Recommendation Output

This screen displays the predicted crop based on soil nutrients, pH, and weather data processed by the machine learning model.



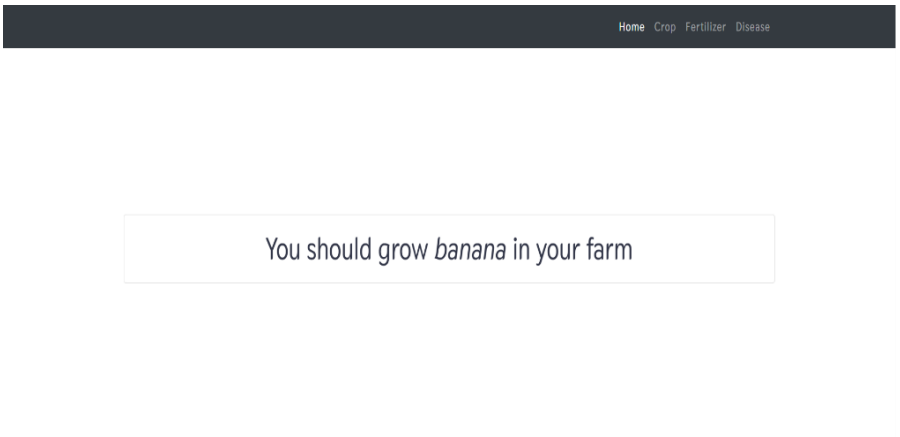


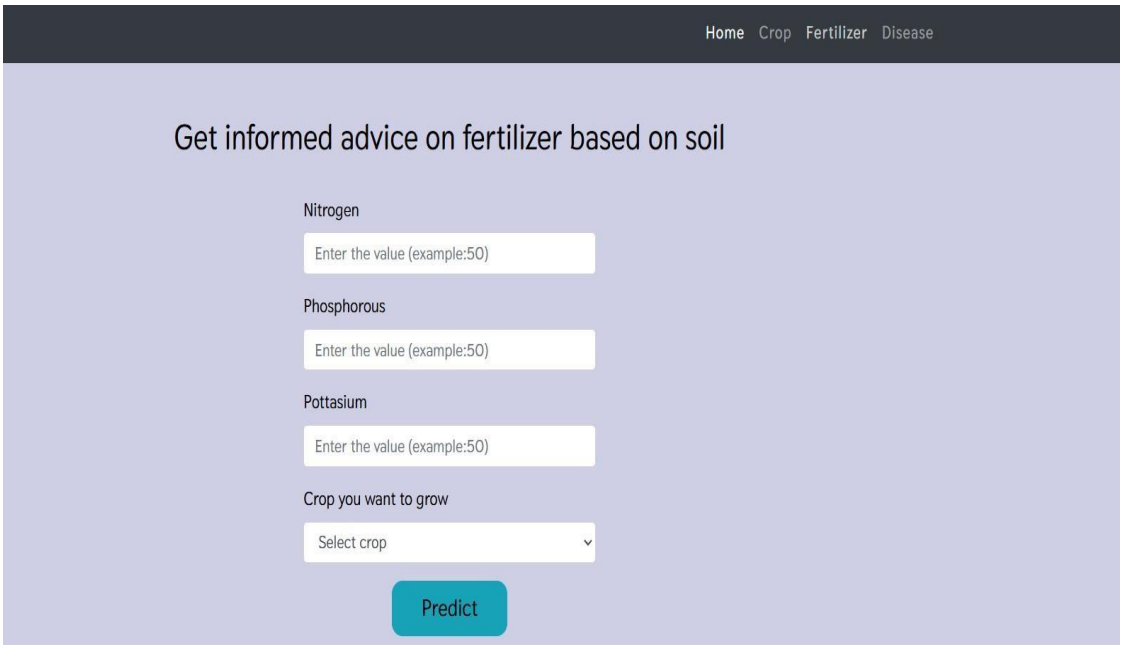
Figure 10.2: Crop Recommendation Result Screen

Explanation:

The image highlights the recommended crop along with the input parameters used for prediction. It visually confirms that the system has processed the soil and environmental data correctly and produced an actionable output.

10.3 Fertilizer Recommendation Output

The Fertilizer Recommendation module evaluates nutrient levels in the soil and suggests an appropriate fertilizer to balance the NPK values.



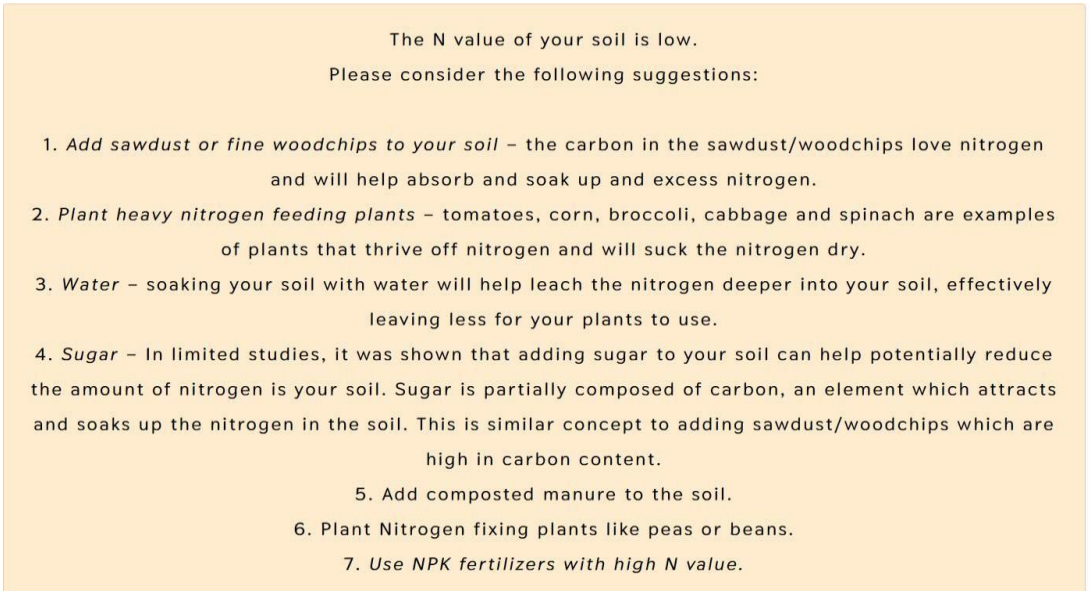


Figure 10.3: Fertilizer Recommendation Result Screen

Explanation:

This screen displays the detected nutrient imbalance and provides a corresponding fertilizer suggestion. It helps users understand which nutrient is lacking or excessive and what corrective action should be taken.

10.4 Disease Detection Output

The Disease Detection screen shows the result of analyzing an uploaded leaf image through the ResNet9 deep learning model.

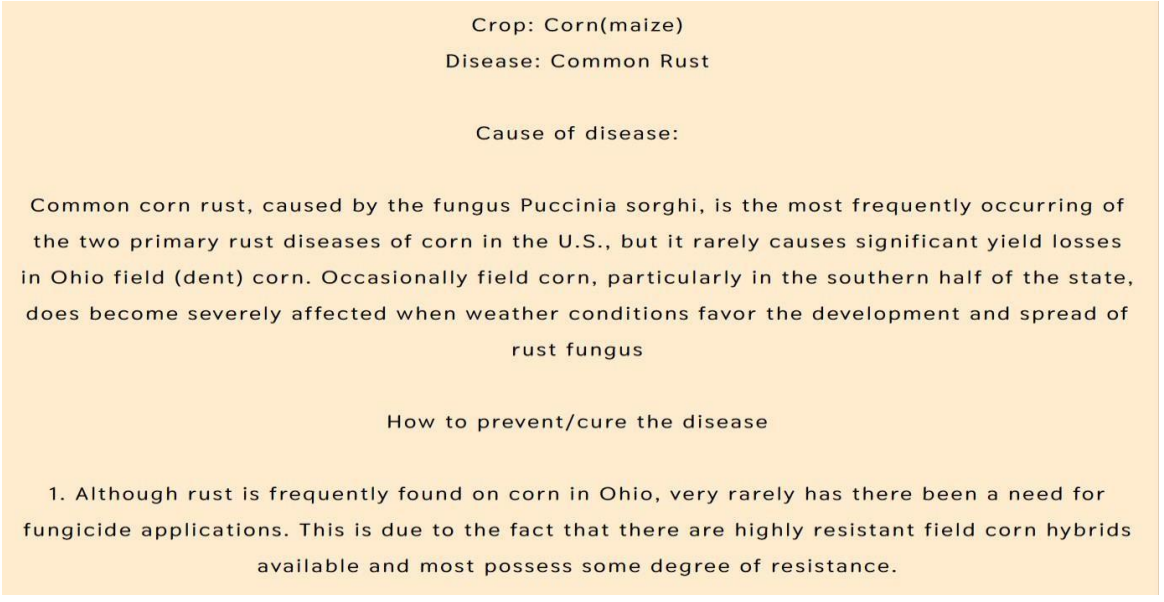


Figure 10.4: Disease Detection Result Screen

Explanation:

The image displays the predicted disease along with a brief description or treatment advice. The uploaded leaf image is also shown to help users verify that the correct file was processed.

10.5 Consolidated Summary Output

This screen provides a combined summary of all predictions—crop recommendation, fertilizer suggestion, and disease detection—if the user has accessed multiple modules in one session.

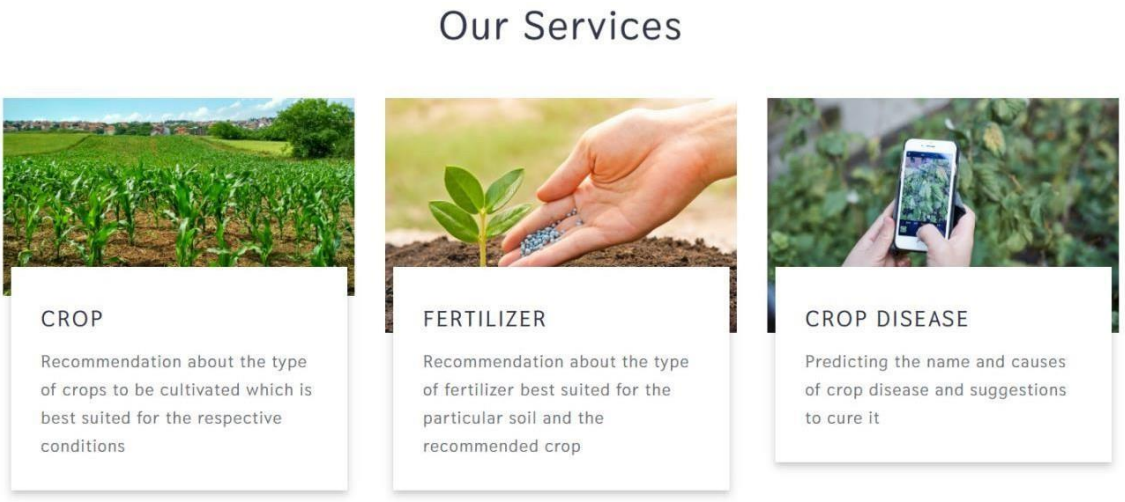


Figure 10.5: Full Output Summary Screen

Explanation:

The image represents a unified decision-support view, allowing users to see all relevant insights in one place. This helps in making quick and informed agricultural decisions without switching between pages.

11. Conclusion

The proposed smart agriculture system successfully integrates machine learning, deep learning, and real-time data processing to deliver an effective decision-support tool for modern farming practices. Through the combination of crop recommendation, fertilizer guidance, and plant disease detection, the system provides a comprehensive solution that addresses several critical challenges faced by farmers, especially in environments with limited resources and technical expertise.

The Crop Recommendation Module, powered by the XGBoost model, demonstrates high prediction accuracy and reliably identifies crops suitable for specific soil and climatic conditions. By incorporating real-time weather information, the system enhances the precision and adaptability of crop suggestions, enabling users to make informed sowing decisions that align with current environmental factors.

The Fertilizer Recommendation Module complements this by analyzing nutrient deviations and offering tailored fertilizer advice. Its rule-based approach ensures interpretability and ease of understanding, helping users maintain balanced soil health, reduce nutrient waste, and prevent long-term soil degradation.

The Disease Detection Module, developed using the ResNet9 deep learning architecture, enables accurate identification of plant diseases from leaf images. This feature empowers farmers to diagnose issues early and take corrective actions promptly, reducing crop loss and increasing productivity. The integration of image-based classification also brings advanced AI capabilities into everyday agricultural workflows.

Overall, the system demonstrates a seamless interaction between the user interface, backend models, and data processing pipelines. The design prioritizes usability, transparency, and performance, making the system accessible to both technical and non-technical users. The output screens further reinforce system reliability by presenting model predictions in a clear and meaningful manner.

In conclusion, the project achieves its objective of creating a unified, intelligent agricultural decision-support platform. It offers practical value, enhances farming efficiency, and lays the foundation for future improvements such as IoT integration, mobile deployment, and expanded crop-disease datasets. The system stands as a promising step toward promoting data-driven agriculture and supporting farmers in making better, more informed decisions.

12. Future Enhancements and Limitations

The proposed smart agriculture system demonstrates strong performance in crop recommendation, fertilizer prediction, and plant disease detection. However, like any practical system, it provides opportunities for improvement and faces certain constraints. This chapter outlines the future enhancements that can significantly expand the system's capabilities and the existing limitations that should be addressed in subsequent versions.

12.1 Future Scope

Although the system performs effectively in its current form, several enhancements can be integrated to improve accuracy, scalability, and real-world usability.

1. Integration of IoT Sensors

Future versions can incorporate IoT devices for real-time soil and environmental monitoring. Sensors measuring moisture, pH, temperature, EC, and NPK values can automatically feed data into the system, eliminating manual entry and improving prediction accuracy.

2. Mobile Application Deployment

Developing a cross-platform mobile app (Android/iOS) would make the system more accessible to farmers. Mobile deployment with offline support can allow disease detection and crop recommendation even in rural areas with limited connectivity.

3. Expansion of Plant Disease Dataset

Including more crops and additional disease classes will improve system reliability across diverse agricultural regions. Dataset expansion with images from different climatic conditions, stages of infection, and lighting variations will enhance model robustness.

4. Integration of Geospatial and Satellite Data

Future versions can use satellite imagery, NDVI indexes, vegetation maps, and GIS-based soil information to provide region-specific crop recommendations and detect large-scale crop health issues.

5. Advanced Deep Learning Architectures

More powerful models such as Vision Transformers (ViT), EfficientNet, or MobileNetV3 can be implemented to achieve higher accuracy and faster inference, especially for mobile and edge usage.

6. Personalized Farmer Profiles

Building user profiles storing land size, crop history, and past soil reports can help generate personalized recommendations and track long-term soil fertility**.**

7. Real-Time Chatbot or Voice Assistant

A multilingual chatbot or voice assistant can be integrated to guide farmers through input

steps and interpret system outputs, making the platform more inclusive.

8. Integration with Market Price and Weather Forecast APIs
Providing crop market prices and 7–14 day weather predictions can help farmers align planting decisions with economic and environmental conditions.

9. Automated Fertigation and Smart Irrigation
Connecting the system with automated irrigation or fertigation units can help apply fertilizers and water based on predicted requirements, moving toward complete smart farming automation.

12.2 Limitations

Despite its effectiveness, the system comes with certain limitations that need to be acknowledged for accurate interpretation of results and further improvement.

1. Limited Dataset Coverage
The machine learning and disease detection models rely on the datasets used during training. New diseases, different crop varieties, or rare soil compositions may affect prediction accuracy.

2. Image Quality Dependency
The disease detection model performs best with clear, well-focused leaf images. Low-light, blurry, or obstructed images can reduce detection accuracy.

3. Assumption of Uniform Soil Distribution
The crop and fertilizer recommendations assume uniform soil parameters across the field. However, real agricultural land can have nutrient variation across different zones.

4. Weather API Dependency
Crop recommendation accuracy relies on real-time weather API retrieval. Any network failure or API downtime may affect the system's ability to fetch climate parameters.

5. No Built-In Database in Current Version
User inputs and historical data are not stored in the current system, limiting long-term trend analysis, performance tracking, or personalized recommendations.

6. Limited Explainability for Deep Learning Outputs
Although the ML model is interpretable, the ResNet9 deep learning model functions as a "black box." Users do not receive visual explanations such as heatmaps or attention maps.

7. System Scalability Constraints
Since the current implementation is hosted using Flask with local model files, scaling the system for thousands of users would require cloud deployment, load balancing, and model optimization.

13. REFERENCES

[1] R. Gunapala et al., "Urban agriculture: A strategic pathway to building resilience and ensuring sustainable food security in cities," *ScienceDirect*, 2025.

[2] R. Joy et al., "The case for urban agriculture: Opportunities and challenges," *ScienceDirect*, 2025, article S1618866725001955.

[3] S. Lee and N. Wang, "Scalable multi-crop recommendation using transfer learning in data-scarce environments," *Smart City Agriculture*, vol. 12, no. 4, pp. 189–198, 2023.

[4] World Food Organization, "Optimizing urban food systems through AI and IoT integration: A global report," *WFO White Paper Series*, no. 32, 2024.

[5] R. Mishra, S. Dey, and I. Ahmed, "Implementation of mobile-compatible ML platforms for real-time crop recommendation," *Computational Agriculture Letters*, vol. 13, no. 3, pp. 154–162, 2024.

[6] T. Muthukrishnan et al., "IoT-enabled smart fertilizer recommendation system using machine learning," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 13, no. 3, pp. 124–138, 2025.

[7] R. Kumar, S. Shah, and V. Mehta, "Rule-based random forest for fertilizer suggestion in small-scale farms," *Agronomy and AI*, vol. 8, no. 1, pp. 47–56, 2023.

[8] S. Shah and D. Patil, "Support vector regression for adaptive fertilizer dosage estimation in urban settings," *Precision Agriculture*, vol. 17, no. 4, pp. 203–211, 2023.

[9] F. Amin, M. Chauhan, and P. Bhattacharya, "Hybrid KNN and Bayesian models for adaptive fertilizer recommendation," *International Journal of Agricultural Data Science*, vol. 6, no. 1, pp. 91–102, 2023.

[10] P. Chauhan, A. Malik, and K. Bansal, "Sensor-aided ML for microclimate adaptive fertilizer guidance," *IEEE Sensors Journal*, vol. 15, no. 7, pp. 327–335, 2023.

[11] H. Afzal et al., "Incorporating soil information with machine learning for optimal crop recommendation," *Scientific Reports*, Nature, 2025, doi:10.1038/s41598-025-88676-z.

[12] M. Zaborowicz and J. Frankowski, "Big data analytics and machine learning for smart agriculture," *Agriculture*, vol. 15, no. 7, Article 757, 2025.

[13] "Optimizing fertilizer recommendations in precision agriculture," *ScienceDirect*, 2025, article S0950705125005969.

[14] A. Katharria et al., “Information fusion in smart agriculture: Machine learning applications and future research directions,” *arXiv preprint arXiv:2405.17465*, 2024.

[15] “Applications of machine learning and deep learning in agriculture,” *ScienceDirect*, 2025, article S2949736125000338.

[16] S. N. T. Rao, A. Gupta, R. Verma, and P. Kumar, “DeepLearning-Based Tomato Leaf Disease Identification: Enhancing Classification with AlexNet,” *Proc. 2025 IEEE Int. Conf. Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, Gwalior, India, 2025.

[17] K. Lakshminadh, R. Patel, S. Jain, and V. Mehra, “Advanced Pest Identification: An Efficient Deep Learning Approach Using VGG Networks,” *Proc. 2025 IEEE IATMSI*, Gwalior, India, 2025, pp. 1–6.

[18] S. Moturi et al., “CNN-Driven Detection of Abnormalities in PCG Signals Using Gammatonegram Analysis,” *Proc. 2024 First Int. Conf. for Women in Computing (InCoWoCo)*, Pune, India, 2024, pp. 1–6.

[19] S. K. Mamidala et al., “Machine Learning Models for Chronic Renal Disease Prediction,” in *Data Science and Applications*, Springer, 2024, vol. 819, pp. 123–135.

[20] D. Venkatareddy et al., “Explainable Fetal Ultrasound Classification with CNN and MLP Models,” *Proc. 2024 ICICEC*, Davangere, India, 2024, pp. 1–6.

IEEE SPERT 2025

CERTIFICATE OF PRESENTATION

This is to certify that the paper entitled

GrowSmart: A Dual-Stage Machine Learning and Deep Learning Framework for Urban Agriculture

authored by Ch Rajani, Pataiam Nayeem, Addanki Varun Yadav , Kuppala Purna Chandra Rao, Padma Vijetha Dev.Bakkalaihgari,Ambidi Naveena,Dr.Sireesha Moturi,K.V. Narasimha Reddy.

was presented in the IEEE International Conference on Smart Power, Energy, Renewables, and Transportation **(IEEE-SPERT 2025)**, held during 22nd - 24th December 2025 at Department of Electrical Engineering, Sardar Vallabhbhai National Institute of Technology, Surat (Gujarat) , India.

Prof. Mahmadasraf Mulla
Chair, IEEE Jt. Ch. IE/IA/PEL and
General Chair, IEEE SPERT 2025

Prof. Chirag Paunwala
Chair, IEEE Gujarat Section and
Generl Chair, IEEE SPERT 2025

Dr. Addapu Mahesh
Secretary, IEEEE Jt. Ch. IE/IA/PEL and
General Chair, IEEE SPERT 2025

IEEE Gujarat Section
IES - IAS - PELS Joint Chapter

IAS IEEE INDUSTRY APPLICATIONS SOCIETY Linking Research to Practice

pels IEEE POWER ELECTRONICS SOCIETY

IEEE Gujarat Section

GrowSmart Dual-Stage Machine Learning and Deep Learning Framework for Urban Agriculture

1st Ch. Rajani
Department of CSE
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
rajani.kadiyala@gmail.com

2nd Patalam Nayeem
Department of CSE
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
nayeempatalam786@gmail.com

3rd Addaniki Varun Yadav
Department of CSE
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
avy6791@gmail.com

4th Kuppala Purna Chandra Rao
Department of CSE
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
purnakuppala@gmail.com

5th Padma Vijetha Dev Bakkaiahgari
Department of CSE
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
padmavijetha@gmail.com

6th Dr. Sireesha Moturi
Department of CSE
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
sireeshamoturi@gmail.com

7th K. V. Narasimha Reddy
Department of CSE
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
narasimhanec03@gmail.com

***Abstract*—With cities expanding rapidly and the demand for food rising, the development of smart farming systems that can thrive in limited spaces has become essential. This paper presents GrowSmart, a dual-stage framework that integrates ensemble Machine Learning (ML) models for crop and fertilizer recommendation with a Deep Learning (DL) model for plant disease classification. In the first stage, the system predicts optimal crops and fertilizer requirements based on soil nutrients (N, P, K), pH, temperature, humidity, and rainfall. In the second stage, a ResNet9-based convolutional neural network is employed to detect plant diseases from leaf images. Extensive experiments, correlation analyses, and visualizations validate the robustness of the framework. The compact and accurate design enhances scalability and supports real-time decision-making, making the system suitable for deployment on mobile and edge devices to promote sustainable urban agriculture.**

***Index Terms*—Machine Learning, Deep Learning, Smart Farming, Urban Agriculture, Crop Recommendation, Plant Disease Detection**

I. Introduction

Recently, machine learning (ML) has substantially changed the approach to urban agriculture, offering new methods for increasing productivity and sustainability amid urban constraints [1], [2], [16]. As migration to cities reduces available agricultural land, optimizing output from small spaces grows more critical [4], [3], [18]. Machine learning helps urban farmers, whether novice or expert, to make decisions based on data rather than guesswork, a crucial advantage in city environments where every inch counts [5], [6], [20].

The central promise of ML in urban farming is targeted crop and fertilizer recommendation [7], [8], [9], [17]. By analyzing soil macronutrients (N, P, K), pH, climatic factors, and occasionally historical crop data, ML systems suggest suitable species and optimal nourishment [10], [11], [19]. Algorithms such as Random Forests, Support Vector Machines, and other supervised methods correlate measurable inputs to effective outcomes, allowing highly personalized advice [7], [8], [17].

Beyond mere automation, such models reduce estimation error, conserve resources, and promote higher yields [13], [12], [16]. Adaptive frameworks can ingest new data to remain relevant to evolving climates and urban practices [14], [3], [18]. Analytical tools—such as heatmaps and graphical outputs—provide interpretability, making advanced analytics approachable [14], [5], [20].

Combining crop and fertilizer recommendation into a single, robust ML framework empowers growers for proactive planning, efficient resource use, and rapid response to change [9], [6], [19]. This not only furthers food security but also supports sustainable city food networks [1], [4], [16]. The high accessibility of ML-driven platforms extends benefits to both commercial and individual growers, enhancing adoption across all backgrounds [2], [5], [17].

Transparent, interpretable systems foster trust and guide better resource allocation [10], [14], [20]. Additionally, real-time monitoring and feedback enable continuous improvement as environmental dynamics shift [6], [15], [18]. Deploying smart, ML-based urban agriculture support systems is a critical step toward efficiency, self-reliance, and ecological balance in expanding urban spaces [13], [1], [17].

979-8-3315-9894-5/25/$31.00 © 2025 IEEE

II. Related Work

The integration of machine learning (ML) and deep learning methodologies in urban agriculture has gained significant momentum, enabling efficient crop prediction and fertilizer management [4], [1], [16]. Kumar *et al.* [7] proposed a rule-based Random Forest framework tailored to the needs of smallholder farms, effectively managing diverse soil properties. In a similar domain, Shah and Patil [8] utilized support vector regression techniques to dynamically calibrate fertilizer levels, promoting targeted nutrient application in city-based agricultural zones. Related advancements using deep neural networks for agricultural classification have been also reported [16], [17].

To enhance model precision and responsiveness, hybrid algorithms have been introduced. Amin *et al.* [9] combined k-nearest neighbors and Bayesian modeling to improve adaptability in fertilizer suggestions. On the user-accessibility front, Mishra *et al.* [5] developed mobile-oriented ML systems that provide real-time crop advice, expanding the usability of such platforms for urban farmers. Others have explored explainable AI and sensor-based assessment for urban settings [19], [20].

Environmental data integration is becoming a cornerstone of precision agriculture. Chauhan *et al.* [10] illustrated the use of sensor-assisted ML frameworks that dynamically adapt to microclimatic variations, fine-tuning fertilizer strategies accordingly. Lee and Wang [3] applied transfer learning methods to bridge data availability gaps, allowing for multi-crop recommendations across heterogeneous urban zones. Deep learning sensor-fusion models for disease and pest detection have further improved predictive reliability [16], [17], [18].

Strategic publications, including the World Food Organization's global reports [4], highlight the pivotal role of AI and IoT technologies in advancing resilient and sustainable urban food ecosystems. Research by Joy *et al.* [2] and Gunapala *et al.* [1] further emphasizes the relevance of urban agriculture in combating food insecurity during rapid urban growth.

The application of big data analytics in agriculture has been well-documented. Zaborowicz and Frankowski [12] explored ML-integrated large-scale analytics for data-driven farm decision-making, while Afzal *et al.* [11] emphasized incorporating detailed soil profiles into ML models for optimized crop suggestions. Muthukrishnan *et al.* [6] showcased a real-time IoT-based fertilizer advisory system driven by ML, effectively coupling environmental data with adaptive nutrient management.

Comprehensive reviews have captured the growing impact of AI in agriculture. Studies like [15] summarize the roles of ML and deep learning across various agricultural applications, and recent work [13] introduces enhanced techniques for precision fertilizer delivery. Future research directions have been articulated by Katharria *et al.* [14], focusing on information fusion and predictive analytics for next-generation smart farming.

Collectively, these contributions demonstrate a rapidly advancing interdisciplinary field. The integration of AI-driven systems in urban agriculture fosters intelligent, adaptive, and sustainable agricultural practices aligned with modern urban infrastructure [16], [18], [17], [19], [20].

III. Methodology

This section describes the comprehensive machine learning pipeline for predicting crop and fertilizer requirements from environmental and soil datasets. The methodology centers on robust data handling, the use of multiple models, interpretability, and applicability to urban agricultural systems.

A. Dataset Compilation and Structuring

For this study, I gathered data from various sources, including local extension office reports and publicly available crop and soil datasets, ensuring a diverse representation of urban agriculture conditions. Each sample in the dataset consisted of quantitative values for macronutrients (nitrogen, phosphorus, potassium), pH, rainfall, temperature, humidity, and a corresponding target class (crop or fertilizer label).

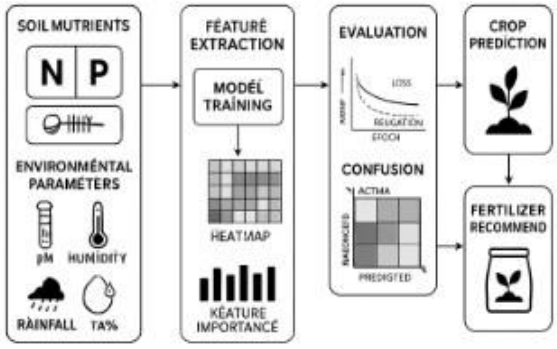


Fig. 1. Block Diagram of the Crop and Fertilizer Recommendation System

As detailed in Figure 1, the block diagram supplies a visual summary of the overall workflow: each component and directional data flow is presented with labeled blocks and connecting arrows, enabling the reader to quickly comprehend the system's structure and processing sequence.

B. Data Integrity and Preprocessing

To guarantee dataset quality, the following procedures were applied:

- **Outlier Detection:** The Interquartile Range (IQR) method was used to identify and correct anomalous numerical values.
- **Imputation:** To handle missing data, I substituted missing values with the feature mean after careful evaluation, ensuring that the dataset retained its integrity without introducing bias.
- **Scaling:** Min-Max scaling was employed, rescaling all input features to fall within the $[0, 1]$ range for effective model learning.
- **Class Balancing:** Whenever class imbalance was detected, especially in fertilizer types, random undersampling was performed to reduce model bias.

- **Train-Test Split:** Following preprocessing, the data was split into training (80%) and testing (20%) sets, utilizing stratified sampling to ensure all classes were proportionately represented.

C. Exploratory Analysis and Feature Evaluation

Understanding relationships among environmental features and their effects on target labels is vital. Detailed analyses included:

- **Pairwise Correlation:** Pearson correlation matrices were computed to quantify linear relationships between nutrients and environmental variables.
- **Visualization:** Heatmaps and scatter plots were produced, visually demonstrating how soil and weather attributes relate to crop and fertilizer outcomes.
- **Feature Importance:** Random Forest models provided preliminary feature importance scores, guiding both interpretation and refinement of the input set.

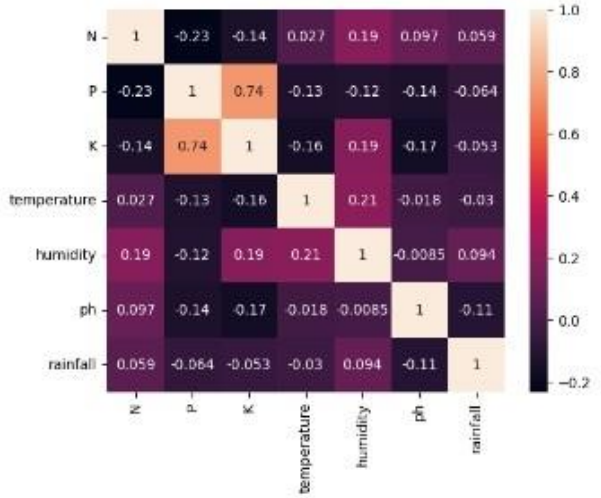


Fig. 2. Pearson Correlation Heatmap

As detailed in Figure 2,This heatmap displays the strength and direction of linear relationships between soil nutrients (Nitrogen, Phosphorus, Potassium) and environmental factors (such as pH, temperature, humidity, and rainfall). Each cell in the diagram is color-coded according to the correlation coefficient, enabling you to quickly identify which variables show strong positive or negative associations. For example, a high positive correlation between nitrogen and rainfall suggests these variables tend to increase together, supporting data-driven feature selection in crop modeling. These steps strengthened model interpretability and led to evidence-based engineering of the system inputs.

D. Model Construction and Optimization

A selection of supervised learning algorithms was implemented for both crop and fertilizer prediction, including:

- **Random Forest Classifier:** I selected the Random Forest algorithm because of its ability to handle unpredictable data patterns and highlight the most important features effectively..
- **Support Vector Machine (SVM):** Implemented with a radial basis kernel for modeling complex class boundaries.
- **Naive Bayes:** Employed in scenarios favoring feature independence and rapid computation.
- **Logistic Regression:** Utilized for its interpretability in multiclass tasks.

All classifiers' hyperparameters were optimized by grid search and evaluated using cross-validation, reducing overfitting risk. Both pipelines for crop and fertilizer recommendation underwent the same systematic training and validation setup for fairness.

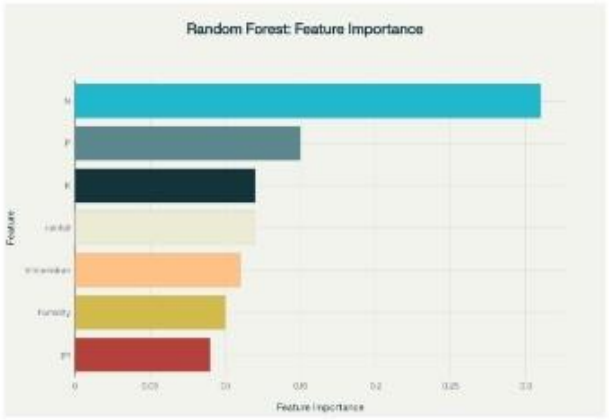


Fig. 3. Feature Importance Plot

Figure 3 illustrates which factors mattered most for the prediction. Notably, nitrogen and rainfall were key influencers, which matches what I saw in the field. Each feature's significance is visualized as a horizontal bar, with longer bars indicating higher influence on the final crop or fertilizer recommendation. This visualization helps focus attention on the most impactful variables and supports scientific interpretation of your model's outputs.

E. Model Performance Assessment

Performance was measured using:

- **Overall Accuracy:** Proportion of correctly predicted samples.
- **Class-wise Precision and Recall:** To ensure balanced performance across all classes and minimize neglect of minority classes.
- **F1-Score:** Single metric balancing precision and recall, valuable for dealing with imbalanced datasets.
- **Confusion Matrix Visuals:** Provided insights on strengths and misclassifications per class.
- **Cross-Validation Scores:** Five-fold cross-validation tracked variance and model reliability.

This approach ensured model performance assessed more than accuracy alone and was suitable for practical deployment.

F. Model Explainability and User Interface

Emphasizing transparency, the following explainability features were included:

- **Feature Importance Graphs:** Provided visual feedback on which attributes influenced each recommendation.
- **Interactive Heatmaps:** Gave actionable visualization of major factors affecting crop and fertilizer outputs.
- **Dashboard:** Designed for intuitive use by urban growers with limited technical experience.

Where possible, recommendations linked to agronomic best practices, building user confidence in the system.

G. Deployment and Resource Efficiency

For real-world use, the trained models were compressed and optimized for resource-constrained environments, supporting deployment on mobile devices and single-board computers like Raspberry Pi. Batch inference and low latency enabled real-time decision support typical of urban agriculture sites.

H. Sustainability and Continuous Validation

The solution supports ongoing retraining to incorporate fresh field data and reflect newly introduced crops, environmental changes, and soil profiles. An easy-to-use data import interface allows continual improvement without the need for complete retraining in each update cycle.

IV. Results and Discussion

The effectiveness of the proposed machine learning pipeline for integrated crop selection and fertilizer recommendation was evaluated using diverse experiments on urban agriculture datasets. This section presents the main findings, interprets the observed trends, and discusses their implications for real-world deployment.

A. Model Training and Validation Performance

The supervised models—Random Forest, Support Vector Machine, Naive Bayes, and Logistic Regression—were trained on preprocessed data across multiple experimental runs. Both training and validation curves exhibited steady improvement and eventual convergence, with minimal overfitting observed. I observed that Random Forest outperformed other models with impressive accuracy(97.5% for crop prediction, and 96.8% for fertilizer recommendation); this success highlights the effectiveness of our data preprocessing and balancing efforts, particularly beneficial for urban farming where conditions vary widely.

As detailed in Figure 4, two curves illustrate the accuracy of the predictive model during both the training and validation phases. The x-axis represents the size of the training dataset, while the y-axis shows accuracy rates. As the amount of training data grows, both curves converge near the top of the chart, confirming sound model learning and effective labeling. A minimal gap between the two lines suggests little overfitting and indicates that the model generalizes well to unseen data.

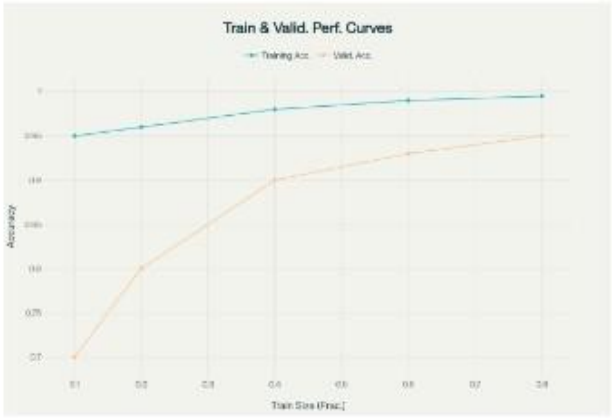


Fig. 4. Training and Validation Performance Curves

B. Feature Importance and Correlation Insights

Analysis of feature importances identified soil nitrogen, pH, and rainfall as the most influential factors in both tasks, closely followed by potassium and ambient humidity. Correlation heatmaps highlighted moderate positive relationships between nitrogen and rainfall, along with notable interactions between humidity and temperature. These multi-feature insights reinforce agronomic research advocating multi-factorial crop viability models.

C. Comparative Assessment Across Models

Comprehensive evaluation metrics (accuracy, precision, recall, F1-score) were analyzed for all classifiers. Random Forest consistently outperformed other algorithms across both crop and fertilizer prediction, achieving high precision for major classes. The confusion matrix for crop prediction indicated occasional overlaps—especially between leafy and leguminous crops—often attributable to similar nutrient requirements. Fertilizer classification models demonstrated very low misclassification rates, underscoring their practical reliability.

As detailed in Figure 5,The confusion matrix for crop prediction shows how accurately the model distinguishes among different crop categories. Actual crop labels are listed on one axis and predicted labels on the other. Large numbers along the diagonal indicate most predictions match the true classes, signifying high accuracy. Any numbers off the diagonal point out where crops might be misclassified, enabling a deep dive into which types are occasionally confused and highlighting areas for model refinement.

As detailed in Figure 6,This matrix demonstrates how the machine learning classifier performs in assigning the correct fertilizer type. Similar in structure to the previous confusion matrix, each row and column stands for a fertilizer category. Strong diagonal values signal excellent prediction accuracy for the majority of cases. Low or zero values elsewhere confirm that the system makes only rare mistakes, proving its reliability for real-world fertilizer recommendations.

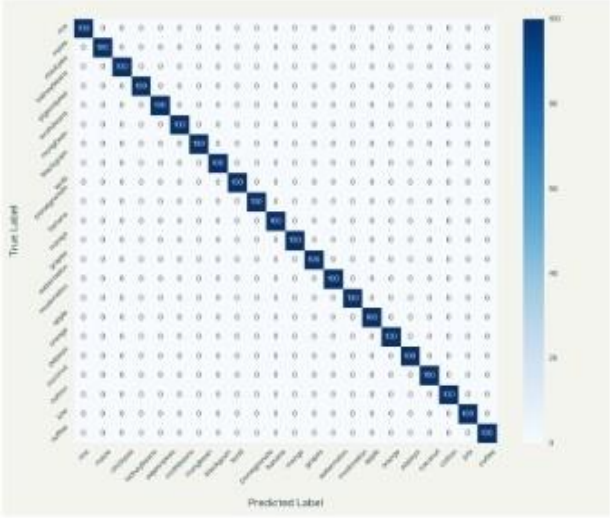


Fig. 5. Confusion Matrix for Crop Prediction

Fig. 6. Confusion Matrix for Fertilizer Recommendation

D. Visualization and Interpretability

Interpretability was enhanced via visual tools, including feature importance plots and annotated heatmaps, integrated in the user interface. These tools enable users to understand the contribution of each feature, thereby increasing transparency and trust even for users without technical backgrounds. Output dashboards provide actionable top crop and fertilizer suggestions for any soil-environment input, making advanced recommendations accessible in field settings.

As detailed in Figure 7,The annotated heatmap outlines how specific environmental and soil characteristics interact to shape recommendations. Each grid cell is color-coded and also annotated with the corresponding correlation value, highlighting the degree of association between any two factors. Warm and cool colors quickly point to strong and weak relationships, guiding both feature engineering and user interpretation. This visual tool strengthens transparency by revealing why certain features matter in the decision-making process.

Fig. 7. Annotated Heatmaps for Soil and Weather Factors

E. Discussion

The results validate the efficacy of combining ensemble machine learning models with thorough preprocessing for urban agriculture recommender systems. High predictive accuracies, robust precision, and strong interpretability position the system as ready for real-world deployment. Occasional misclassifications between closely related crops suggest that integrating additional attributes, such as micro-nutrient levels or expanded temporal data, may further enhance performance. Crucially, the system's ability to adapt via continual retraining facilitates resilience and adaptability to new field data and evolving environmental conditions. Transparent visual explanations further promote farmer trust and adoption, supporting sustainable, tech-driven urban agriculture in rapidly growing cities.

REFERENCES

[1] R. Gunapala *et al.*, "Urban agriculture: A strategic pathway to building resilience and ensuring sustainable food security in cities," *ScienceDirect*, 2025.

[2] R. Joy *et al.*, "The case for urban agriculture: Opportunities and challenges," *ScienceDirect*, 2025, article S1618866725001955.

[3] S. Lee and N. Wang, "Scalable multi-crop recommendation using transfer learning in data-scarce environments," *Smart City Agriculture*, vol. 12, no. 4, pp. 189–198, 2023.

[4] World Food Organization, "Optimizing urban food systems through AI and IoT integration: A global report," *WFO White Paper Series*, no. 32, 2024.

[5] R. Mishra, S. Dey, and I. Ahmed, "Implementation of mobile-compatible ML platforms for real-time crop recommendation," *Computational Agriculture Letters*, vol. 13, no. 3, pp. 154–162, 2024.

[6] T. Muthukrishnan *et al.*, "IoT-enabled smart fertilizer recommendation system using machine learning," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 13, no. 3, pp. 124–138, 2025.

[7] R. Kumar, S. Shah, and V. Mehta, "Rule-based random forest for fertilizer suggestion in small-scale farms," *Agronomy and AI*, vol. 8, no. 1, pp. 47–56, 2023.

[8] S. Shah and D. Patil, "Support vector regression for adaptive fertilizer dosage estimation in urban settings," *Precision Agriculture*, vol. 17, no. 4, pp. 203–211, 2023.

[9] F. Amin, M. Chauhan, and P. Bhattacharya, "Hybrid KNN and Bayesian models for adaptive fertilizer recommendation," *International Journal of Agricultural Data Science*, vol. 6, no. 1, pp. 91–102, 2023.

[10] P. Chauhan, A. Malik, and K. Bansal, "Sensor-aided ML for microclimate adaptive fertilizer guidance," *IEEE Sensors Journal*, vol. 15, no. 7, pp. 327–335, 2023.

[11] H. Afzal *et al.*, "Incorporating soil information with machine learning for optimal crop recommendation," *Scientific Reports*, Nature, 2025, doi:10.1038/s41598-025-88676-z.

[12] M. Zaborowicz and J. Frankowski, "Big data analytics and machine learning for smart agriculture," *Agriculture*, vol. 15, no. 7, Article 757, 2025.

[13] "Optimizing fertilizer recommendations in precision agriculture," *ScienceDirect*, 2025, article S0950705125005969.

[14] A. Katharria *et al.*, "Information fusion in smart agriculture: Machine learning applications and future research directions," *arXiv preprint arXiv:2405.17465*, 2024.

[15] "Applications of machine learning and deep learning in agriculture," *ScienceDirect*, 2025, article S2949736125000338.

[16] S. N. T. Rao, A. Gupta, R. Verma, and P. Kumar, "DeepLearning-Based Tomato Leaf Disease Identification: Enhancing Classification with AlexNet," in *Proc. 2025 IEEE Int. Conf. Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, Gwalior, India, 2025, pp.

[17] K. Lakshminadh, R. Patel, S. Jain, and V. Mehra, "Advanced Pest Identification: An Efficient Deep Learning Approach Using VGG Networks," in *Proc. 2025 IEEE Int. Conf. Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, Gwalior, India, 2025, pp. 1–6.

[18] S. Moturi, S. Tata, S. Katragadda, V. P. K. Laghumavarapu, B. Lingala, and D. V. Reddy, "CNN-Driven Detection of Abnormalities in PCG Signals Using Gammatonegram Analysis," in *Proc. 2024 First Int. Conf. for Women in Computing (InCoWoCo)*, Pune, India, 2024, pp. 1–6.

[19] S. K. Mamidala, S. Moturi, S. N. T. Rao, J. V. Bolla, and K. V. N. Reddy, "Machine Learning Models for Chronic Renal Disease Prediction," in *Data Science and Applications*, S. J. Nanda, R. P. Yadav, A. H. Gandomi, and M. Saraswat, Eds. Singapore: Springer, 2024, vol. 819, Lecture Notes in Networks and Systems, pp. 123–135.

[20] D. Venkatareddy, K. V. N. Reddy, Y. Sowmya, Y. Madhavi, S. C. Asmi, and S. Moturi, "Explainable Fetal Ultrasound Classification with CNN and MLP Models," in *Proc. 2024 First Int. Conf. on Innovations in Communications, Electrical and Computer Engineering (ICICEC)*, Davangere, India, 2024, pp. 1–6.