A

Major Project Report

On

Smart Farming Precision Agriculture Using Machine Learning

Submitted to ***CMREC, HYDERABAD***

In Partial Fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Submitted By

T. Veera Prasanna Lakshmi (218R1A6760)
M. Srishanth (218R1A6740)
G. Siddhartha (218R1A6725)
L. Manish (228R5A6706)

Under the Esteemed guidance of

Mrs. N. Madhavi

Assistant Professor, Department of CSE (Data Science)

Department of Computer Science & Engineering (Data Science)

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad)
Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401.

2024-25

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Accredited by NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU,Hyderabad)
Kandlakoya, Medchal Road, Hyderabad-501 401

Department of Computer Science & Engineering (Data Science)

CERTIFICATE

This is to certify that the project entitled **"Smart Farming – Precision Agriculture using Machine Learning"** is a Bonafide work carried out by in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING(DATASCIENCE)** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

T. Veera Prasanna Lakshmi (218R1A6760)
M. Srishanth (218R1A6740)
G. Siddhartha (218R1A6725)
L. Manish (228R5A6706)

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	**Project Coordinator**	**Head of the Department**	**External Examiner**
Mrs. N. Madhavi	**Mrs. G. Shruthi**	**Dr. M. Laxmaiah**	
Assistant Professor CSE (Data Science), CMREC	Assistant Professor CSE (Data Science), CMREC	Professor & HoD CSE (Data Science), CMREC	

DECLARATION

This is to certify that the work reported in the present Major project entitled "**Smart Farming – Precision Agriculture using Machine Learning"** is a record of Bonafide work done by us in the Department of Computer Science and Engineering (Data Science), CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

T. Veera Prasanna Lakshmi (218R1A6760)
M. Srishanth (218R1A6740)
G. Siddhartha (218R1A6725)
L. Manish (228R5A6706)

ACKNOWLEDGMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. M. Laxmaiah**, Professor, HOD, **Department of CSE (Data Science), CMR Engineering College** for their constant support**.**

We are extremely thankful to **Mrs. N. Madhavi,** Assistant Professor, Internal Guide, Department of CSE(DS), for her constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We thank **Mrs. G. Shruthi**, Assistant Professor, CSE (DS) Department**,** Project Coordinator for her constant support in carrying out the project activities and reviews.

We express my thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, We are very much thankful to our parents who guided me for every step.

T. Veera Prasanna Lakshmi (218R1A6760)
M. Srishanth (218R1A6740)
G. Siddhartha (218R1A6725)
L. Manish (228R5A6706)

ABSTRACT

The practice of cultivating the soil, producing crops, and keeping livestock is referred to as farming. Agriculture is critical to a country's economic development. Nearly 58 percent of a country's primary source of livelihood is farming. Farmers till date had adopted conventional farming techniques. These techniques were not precise thus reduced the productivity and consumed a lot of time. Precise farming helps to increase the productivity by precisely determining the steps that needs to be practiced at its due season. Predicting the weather conditions, analyzing the soil, recommending the crops for cultivation, determine the amount of fertilizers, pesticides that need to be used are some elements of precision farming.

Precise Farming uses advanced technologies such as IOT, Data Mining, Data Analytics, Machine Learning to collect the data, train the systems and predict the results. With the help of technologies Precise farming helps to reduce manual labor and increase productivity. Farmers have been facing various challenges in these recent times, this includes crop failure due to less rainfall, infertility of soil and so on. Due to the changes taking place in the environment the proposed work helps to identify how to manage crops and harvest in a smart way. It guides an individual for smart farming.

The aim of this work is to help an individual cultivate crops efficiently and hence achieve high productivity at low cost. It also helps to predict the total cost needed for cultivation. This would help an individual to pre-plan the activities before cultivation resulting in an integrated solution in farming.

CONTENTS

LIST OF FIGURES

LIST OF SCREENSHOTS

1. INTRODUCTION

1.1 OVERVIEW

Agriculture plays a vital role in sustaining the global economy and ensuring food security. However, traditional farming methods often struggle with inefficiencies, resource wastage, and unpredictable environmental factors, leading to reduced crop yields and financial losses. Factors such as soil quality, climate variations, pest infestations, and improper irrigation contribute to suboptimal agricultural productivity. The ability to monitor, analyze, and optimize farming conditions is crucial for enhancing crop yields, minimizing resource consumption, and ensuring sustainable agricultural practices. Traditional farming decision-making relies on manual observations and rule-based approaches, which often fail to leverage the vast amounts of data available in modern agricultural systems. These conventional methods lack the ability to process high-dimensional data, identify hidden patterns, and provide real-time insights, limiting their effectiveness in precision farming applications.

To address these limitations, this project introduces a Machine Learning (ML)-powered Precision Agriculture System that leverages advanced data analytics and predictive modeling for smart farming. The proposed system utilizes IoT sensors and satellite imagery to collect real-time data on soil moisture, weather conditions, crop health, and nutrient levels. Using Random Forest (RF) for feature selection and Convolutional Neural Networks (CNNs) for deep feature extraction and pattern recognition, the model provides accurate predictions on optimal irrigation schedules, pest detection, and disease prevention strategies. A decision support system (DSS) powered by machine learning algorithms classifies crop health status and provides recommendations for fertilizers, pesticides, and harvesting timelines.

The integration of ML techniques in precision agriculture enhances farm productivity by optimizing water usage, reducing chemical overuse, and mitigating risks associated with climate change. By employing real-time data processing and predictive analytics, the system ensures adaptive and data-driven farming decisions, making it highly suitable for modern smart farming applications. The insights generated by this model empower farmers, agricultural scientists, and policymakers to maximize yields, minimize losses, and contribute to sustainable agricultural practices.

1.2 RESEARCH MOTIVATION

The motivation behind this research stems from the urgent need to enhance agricultural productivity and sustainability while addressing the growing challenges faced by traditional farming practices. With the global population projected to reach 9.7 billion by 2050, the demand for food production is increasing exponentially. However, conventional farming methods often result in inefficient resource utilization, unpredictable crop yields, and significant environmental degradation due to excessive use of fertilizers, pesticides, and water. Climate change, soil degradation, and pest infestations further exacerbate the problem, making it essential to adopt data-driven and technology-assisted farming techniques to ensure food security and economic stability.

One of the major research gaps in precision agriculture is the lack of an intelligent, AI-powered decision support system that can process real-time, high-dimensional agricultural data and provide actionable insights. Traditional manual observation and rule-based decision-making methods are inefficient in handling the complexity of farming conditions, often leading to suboptimal decisions regarding irrigation, fertilization, and pest control. Machine Learning (ML)-based predictive models can help address these challenges by identifying hidden patterns in soil health, weather conditions, and crop growth, thus enabling more accurate and timely interventions to optimize farm productivity.

Additionally, the rapid expansion of smart farming and IoT-based agricultural systems presents an opportunity to integrate sensor-driven data collection and ML algorithms for real-time monitoring of soil nutrients, crop diseases, and water levels. Many governments and agricultural organizations are now advocating for the adoption of precision farming techniques to enhance food production while minimizing waste and environmental impact. By developing an ML-driven smart farming system, this research contributes to the advancement of sustainable agriculture, reduction of resource wastage, and maximization of crop yield. The proposed model can assist farmers, policymakers, and agronomists in making informed, data-driven decisions, thus improving overall agricultural efficiency and sustainability.

1.3 PROBLEM STATEMENT

The increasing complexity of agricultural operations and the unpredictability of climate conditions demand a more advanced and intelligent precision farming system. Traditional farming methods often rely on manual observations and fixed-schedule interventions, which are inefficient in optimizing crop yield, soil health, and resource management. These conventional techniques fail to adapt to real-time environmental changes, leading to inefficient water usage, overuse of fertilizers, and susceptibility to pest infestations. Additionally, imbalanced decision-making, where farmers lack access to real-time insights, results in reduced productivity and financial losses. The need for automated, data-driven, and adaptive farming models is greater than ever, especially as climate variability and food demand continue to rise worldwide.

The Smart Farming - Precision Agriculture using ML model is designed to address these challenges by integrating Machine Learning (ML) and Internet of Things (IoT) technologies. Unlike traditional rule-based models, which rely on static assumptions and limited feature sets, this system leverages sensor data, satellite imagery, and ML algorithms to make real-time predictions on factors like soil moisture, weather conditions, crop health, and disease outbreaks. This intelligent approach ensures more accurate, timely, and adaptive decision-making, enabling efficient resource allocation and higher agricultural productivity. Moreover, the model's continuous learning mechanism allows it to adapt to new data trends in farming conditions, ensuring long-term sustainability and efficiency.

One of the key innovations of this ML-based precision agriculture system is its scalability and computational efficiency, allowing deployment in both high-tech commercial farms and small-scale agricultural settings. Many existing models struggle with high computational costs and require specialized hardware, making them impractical for widespread adoption in rural and resource-constrained regions. By employing feature selection techniques, the model minimizes redundant data while preserving critical variables that influence crop yield and health. Additionally, its predictive analytics and automated decision support system help farmers optimize irrigation schedules, detect pest infestations early, and improve soil fertility management, reducing waste and maximizing productivity.

The proposed ML-based Smart Farming system also enables government agencies, agribusinesses, and policymakers to make data-driven decisions for sustainable agriculture and food security. By analyzing climate trends, soil conditions, and crop health patterns, authorities can implement precision farming policies, offer targeted subsidies, and promote sustainable

agricultural practices. Additionally, agricultural cooperatives can leverage AI-driven insights to forecast market trends, improve supply chain efficiency, and reduce post-harvest losses. The integration of AI-powered predictive analytics into modern farming ecosystems represents a significant step toward enhancing global food security, minimizing environmental impact, and ensuring resilient agricultural practices.

2. LITERATURE SURVEY

Crop growth is primarily influenced by the soil's macro nutrient and trace mineral content of the soil. Soil being the broad representation of several environmental factors including rainfall, humidity, sunlight, temperature and soil ph. The use of a support vector machine and decision tree algorithm to distinguish the type of crop based on micronutrients and meteorological characteristics has been presented as an efficient means of predicting the crop. Three crops where selected such as rice, wheat and sugarcane. Based on certain observations details about micro nutrients where been obtained. These details where feed into the classifier model that in turn predicted the crop based on the passed values. There are many Machine Learning algorithms that works in a different manner. Hence selecting only two models will not provide the required output. The accuracy score of SVM was greater than decision tree algorithm with a sore of 92% [14]. In this work best out of two algorithms is selected. But there are various algorithms dedicated for classification tasks.

There is a need for working on other models such as K Neighbors classifier, Logistic Regression, Ensemble classifiers. These algorithms are indeed applied in proposed research work. The [14] predicts only a crop based on the values entered into the SVM model. Data is most valuable. Hence more information can be obtained apart from using them for prediction. The proposed research work not only recommends the crops and also uses the data to obtain various information that would provide a detailed view about the predicted crops this includes specifying the Growing Degree Days such as heat units, amount of heat needed for the crop growth and the amount of nitrogen, phosphorous and potassium content need to be supplied for the growth per 200 lb. fertilizer. Machine Learning algorithms such as SVM and decision tree classifier was used [14] but in this work Machine Learning algorithms such as Decision Tree, K Nearest Neighbor, Linear Regression model, Neural Network, Naïve Bayes and Support Vector Machine was used for recommending a crop to the user. It has provided an exposure to other algorithms compared to [14].

Linear Regression model was used to predict the production value against the climatic parameters such as rainfall, temperature and humidity. The scores of all these algorithms were below 90% [15]. This work was just a model implementation using the dataset. We be inter face needs to be implemented so that even common people can use it efficiently. All the values need to be provided manually for the model to predict the crop. The proposed work helps in extracting temperature and humidity values using Web Scraping. Hence manually entering the values are

not needed. The proposed work provides an interactive web interface where the user specifies the average rainfall and soil Ph value. The temperature and humidity details are extracted automatically and feed into the best model that includes 10 algorithms with hyper parameter tuning. The proposed work tends to achieve an accuracy of 95.45% with hyper parameter tuning the algorithms which was not included in [14]. The predicted results along with certain information are displayed in the web interface which makes the user to understand the results more efficiently.

Base temperature of a given crop can be used to calculate the GDD Growing Degree Days. The main aim of this study is to come up with easy and mathematically acceptable formulas for calculating GDD's base temperature. Temperature data for snap beans, sweet corn, and cow pea are used to propose, prove, and test mathematical formulas. These new mathematical formulae, in comparison to earlier approaches, can produce the base temperature quickly and correctly. These formulas can be used to calculate the GDD base temperature for every crop at any developmental stage [16]. This work provides a formula to calculate the GDD for the crops. Hence the formula specified in [16] was applied to the predicted crop to estimate their GDD in the proposed work.

Weeds grown along with soy bean can be detected using K-means and CNN model. K-means were used for identifying the features of the images and convolutional neural network for was used for classifying the weeds and soy bean. It also suggests that accuracy can we improved by finetuning the CNN model. CNN model provides an efficient way to detect the weeds present among crops. When used along with K-means initially the images and its augmentations are clustered and on using CNN model helps to precisely identify the weed [17]. The proposed work uses the pretrained model such as Resnet152V2 hence it has important layers such as skip layer and identity layer. The main goal of these layer is to make sure that the output image is same as the input. This increases the accuracy and the predictions are correct. Not only predicting the image the proposed model also helps to provide details about the herbicides that can be used which is an additional information for the user.

Existing deep learning techniques are used for weed detection. This study provides information of various ML and Deep Learning algorithms that can be used for identifying weeds. It mainly emphasis on pre-trained models. It suggests that pre-trained models as lot of benefits and hence can be used to image classification. It also provides guidance of how to work on datasets and make the datasets efficient for building the models. Many public datasets are available on various platforms that can be used for this purpose. It specifies Image Resizing, data

augmentation, image segmentation some of the techniques would bring about accurate classifications and tendency of increasing the accuracy is also more in pre-trained models [18]. Since this study provides directions to perform deep learning techniques the proposed model has opted certain techniques preprocessing steps such as Image Resizing, data augmentation is opted before building the actual deep learning model to predict the weeds.

Another algorithm that can be used for identifying weeds in vegetable plantation is the CenterNet. CenterNet is used for weed identification. It includes two stages. In first stage the Bok choy images were collected and detected. In the second stage, color-index based segmentation were performed on the images collected to identify the weeds present in the dataset. The images were collected from Nanjing, China. The images were augmented to increase the dataset size and images were annotated. CenterNet algorithm was used for both training and testing the images. It is a ground-based weed identification technique. More optimization would lead to better results was suggested [19]. CenterNet algorithm is simple yet there is a need an algorithm that strives to get correct prediction. The proposed work uses Resnet152V2 algorithm that strives to achieve more accuracy since it has special layers such as skip layer and identity layer that tries to get input image as output itself. Hence predictions would be absolutely correct. Hence Resnet152V2 algorithm is selected to obtain accurate prediction and based on the prediction obtain the list of herbicides.

Farmers face a challenging task in identifying crop insects since pest infestation destroys a substantial portion of the crop and affects its quality. The use of highly skilled taxonomists to correctly identify insects based on their physical traits is a shortcoming of traditional insect identification. Experiments were conducted using image characteristics and ml algorithms such as neural networks, support vector machine, k-nearest neighbors, naive bayes, and convolutional neural network model to identify twenty-four insects from the Wang and Xie dataset. To increase the performance of the classification models,9-foldcross-validation was used. The CNN model had the greatest classification rates of 91.5 percent and 90 percent, respectively.

The results revealed a considerable improvement in classification accuracy and computational time when compared to state-of-the-art classification algorithms [20]. This work [20] has used basic CNN model for classification as well as the same dataset used by various researchers. Hence the proposed model has used a different dataset called the Pest’s dataset from Kaggle website. This dataset consists of 9 classes of insects. Each image is taken from different locations. This dataset was selected for the proposed model since the model is trained of images about various locations that gives more knowledge for the model to understand the image and

distinguish them. The proposed model uses Resnet152V2 model for classification. The Resnet152V2 model is the basic model and top of which Global Average Pooling 2D, Drop outs and more hidden layers are been implemented. This refers to finetuning the base pre-trained model. This helps in extracting more information and helps in efficient classification.

The association between the degree of difficulty in identifying insects and the identification key was investigated in this article. For a collection of 134 insects, the SPIPOLL database was utilized to generate 193 characteristic value pathways. Based on the average IES of all the insects with that of characteristic value was formulated. The CV's derived IES was then used to generate an estimated IES for each bug, resulting in a ranked list of insects. Finally, the anticipated bug ranking list was compared to the actual bug ranking list. The results showed a significant correlation between the estimated and actual truth IES, indicating that the CV can be used to estimate the IES of SPIPOLL insects [21]. This work has specified of how to consider the features of an image with respect to insects' dataset. Its main goal is to identify a key that helps in distinguishing the classes. This proposed work contributes in specifying that a key is important for distinguishing the insect classes.

Hence the proposed work uses Resnet152V2 algorithm for this very reason. Resnet152V2 is a pre-trained model and it automatically picks the important features rather than manually defining them. The Resnet152V2 base model on addition with Dropouts helps in removing unnecessary hidden layers and selecting the relevant ones is an advantage. Identification of insects does not solve the problem completely. Suggesting Pesticides provides a complete solution. The proposed model helps to identify the insects as well as suggest Pesticides for the same.

Various elements must be considered when estimating the cost of a crop. It divides agricultural costs into five categories and provides calculations for each. It also gives examples of how to figure out how much a crop cost. It is a theoretical article that always guide the implementation of estimating the cost of cultivation [22]. This theoretical study was used in the proposed model to calculate the cost of cultivation. It was very helpful as it provided elementary description to calculate the costs for cultivation. The formulas proposed in this study was used in the proposed system to estimate the costs till the year 2028

3. EXISTING SYSTEM

3.1 EXISTING METHODS

Several traditional and modern approaches have been used in smart farming to improve crop yield, resource management, and automation. However, most existing systems face challenges in scalability, accuracy, and real-time decision-making. Below are some widely used methods and their characteristics:

1. **Rule-Based Agricultural Systems:** Rule-based systems use predefined thresholds for soil moisture, temperature, and humidity to automate irrigation and fertilization. These systems work well for small-scale farms but struggle with dynamic environmental conditions and variations in soil types across different locations.

2. **Traditional Farming Methods:** Conventional farming relies on manual observations, historical knowledge, and generic farming practices. Farmers decide on irrigation, fertilization, and pest control based on past experiences rather than real-time data. While effective in some cases, these methods lack precision, waste resources, and are highly labor-intensive.

3. **Remote Sensing and GIS-Based Systems:** Geographic Information Systems (GIS) and remote sensing technologies use satellite images and aerial data to monitor crop health, detect drought conditions, and optimize land use. While effective for large-scale farming, these methods require high infrastructure costs, expert knowledge, and periodic updates.

4. **IoT-Based Smart Farming Solutions:** Internet of Things (IoT) technology enables real-time monitoring of soil moisture, temperature, and humidity using sensors. These systems allow automated irrigation and fertilizer application. However, challenges include high initial costs, connectivity issues in rural areas, and sensor maintenance.

5. **Machine Learning-Based Crop Prediction Models:** Machine learning algorithms such as Random Forest, Support Vector Machines (SVM), and Gradient Boosting Machines (GBM) analyze historical weather data, soil properties, and crop health to predict yields and optimize farming practices. However, they require large datasets and suffer from model bias if data is insufficient.

3.2 CHALLENGES

Current smart farming systems face several challenges, including high implementation costs, data integration issues, connectivity limitations, and model accuracy constraints. Traditional IoT-

based and AI-driven agricultural systems require significant financial investment, making them inaccessible to small-scale farmers. Data collection and integration from multiple sources, such as soil sensors, weather stations, and satellite images, present interoperability challenges, limiting the efficiency of predictive models.

While machine learning models like Random Forest (RF) and Support Vector Machines (SVM) analyze agricultural data, they struggle with imbalanced datasets, leading to inaccurate predictions for rare but critical events like droughts and pest outbreaks. Similarly, deep learning models such as CNNs and LSTMs, although effective in detecting complex patterns, require large datasets and high computational resources, making real-time decision-making difficult for farms with limited infrastructure.

Another key limitation is poor network connectivity in rural areas, restricting real-time monitoring and data-driven automation. Privacy concerns and cybersecurity threats also hinder smart farming adoption, as farm data, including soil quality and yield forecasts, is vulnerable to cyberattacks. Additionally, farmers' resistance to technology and lack of technical expertise further slowdown adoption, emphasizing the need for user-friendly, cost-effective, and scalable smart farming solutions.

Hybrid AI approaches, integrating IoT-driven real-time monitoring, machine learning-based predictive analytics, and blockchain-secured data storage, offer a promising solution. These advancements can enhance precision agriculture, optimize resource allocation, and improve food security, addressing the critical challenges in modern farming.

4. PROPOSED METHODOLOGY

4.1 OVERVIEW

Ensemble models have been extensively utilized in machine learning to enhance classification accuracy and overall efficiency. The fundamental principle behind ensemble learning is that combining multiple models often results in better performance than using a single classifier. In smart farming, where diverse factors such as soil health, weather conditions, and pest infestations influence agricultural outcomes, ensemble models can improve predictive reliability and decision-making.

To optimize precision agriculture, this study introduces a multi-model ensemble approach that integrates machine learning, deep learning, and IoT-driven analytics. The proposed system consists of several key components, each leveraging different technologies for enhanced accuracy and adaptability. The first ensemble model integrates multiple machine learning algorithms, improving structured data analysis. The second ensemble model combines a machine learning classifier with deep learning, harnessing both structured data processing and advanced feature extraction capabilities.

The proposed system, termed RF-CNN Farming Model, utilizes a soft voting mechanism to enhance predictive accuracy. Soft voting aggregates probability scores from multiple classifiers and determines the final decision based on the weighted average of these probabilities. This method ensures that the model considers confidence levels rather than merely selecting the most frequently predicted outcome. As a result, the system balances multiple predictions, reducing errors and improving reliability in smart farming applications.

By integrating Random Forest (RF) for structured data analysis and Convolutional Neural Networks (CNN) for deep feature extraction, the RF-CNN model effectively captures soil health indicators, plant disease patterns, and environmental variables. RF identifies key farming parameters, while CNN processes drone and sensor images, detecting crop diseases, nutrient deficiencies, and pest infestations. This hybrid approach provides a comprehensive and scalable precision agriculture framework.

The smart farming system consists of:

1. Smart IoT-Based Data Collection – Embedded sensors monitor temperature, humidity, soil moisture, and nutrients, while drones and satellites provide real-time aerial crop health assessments.

2. Machine Learning-Based Predictive Analytics – ML algorithms analyze historical data to predict optimal irrigation schedules, early disease signs, and precise fertilizer use. Deep learning

models classify crop health and detect abnormalities through image analysis.

3. Automated Decision Support System (DSS) – A cloud-based AI platform provides real-time insights and recommendations via mobile or web applications, optimizing resource management and yield forecasting.

4. Autonomous Farming Operations – AI-driven robots and autonomous tractors perform precision tasks like targeted spraying, weeding, and harvesting, reducing labor costs and environmental impact.

5. Supply Chain and Market Integration – The system includes a smart marketplace connecting farmers with buyers, optimizing storage, transportation, and pricing strategies through demand forecasting. Blockchain technology ensures secure and transparent transactions.

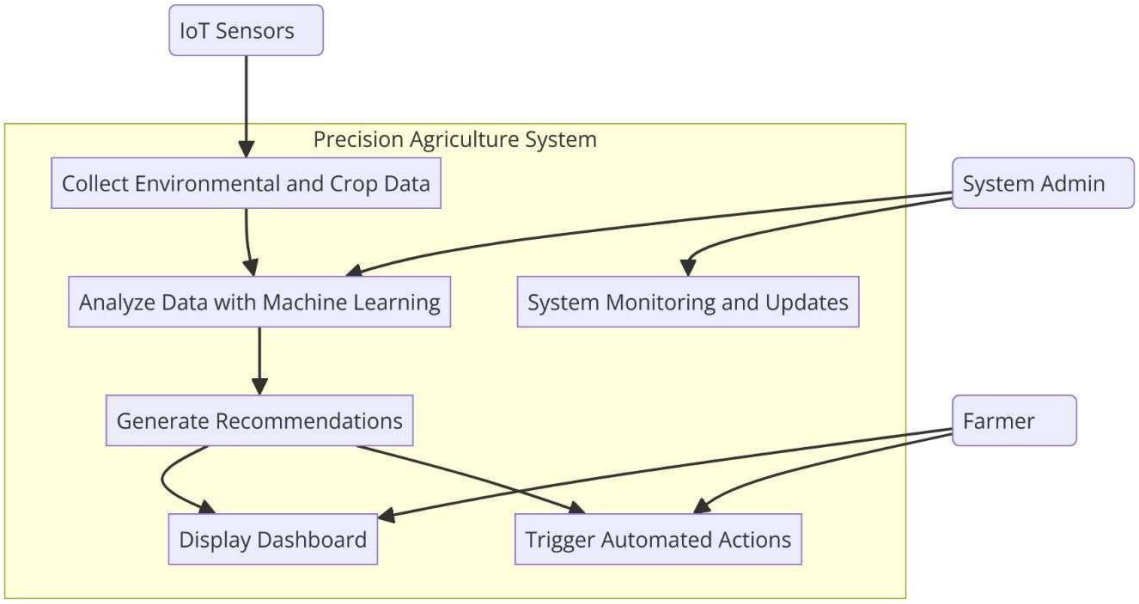


Fig 4.1.1 Block Diagram of Precision Agriculture

4.2 ADVANTAGES

- **High Prediction Accuracy**: Uses ML and DL techniques for accurate crop health monitoring and yield prediction.
- **Effective Feature Selection:** Random Forest ensures only important parameters (soil moisture, temperature, etc.) are used, improving efficiency.
- **Deep Feature Extraction:** CNN helps detect crop diseases and pest infestations from satellite and drone images.
- **Soft Voting Mechanism:** Uses multiple classifiers to enhance decision stability.
- **Improved Handling of Noisy Data:** Reduces the impact of inconsistent sensor data, making predictions more reliable
- **Better Generalization:** Works well across different climates, soil types, and farming

regions.

- **Real-Time Monitoring & Prediction:** Processes live sensor data for immediate corrective actions
- **Enhanced Crop Disease Detection:** Identifies early signs of infections to reduce crop loss.
- **Optimized Resource Utilization:** Provides smart irrigation and fertilization recommendations to minimize waste.
- **Scalability:** Can be adapted to different crops, soil conditions, and environmental factors.
- **Reduced Computational Complexity:** Eliminates redundant data, improving efficiency.
- **Improved Yield Forecasting:** Helps farmers make better harvesting and supply chain decisions.
- **Integration with IoT:** Works with smart farming devices for real-time monitoring and automation
- **Supports Sustainable Farming:** Promotes eco-friendly agricultural practices.
- **Facilitates Precision Agriculture:** Enables location-based insights for better resource use.
- **Automated Weed and Pest Control:** Can be used with robotic farming solutions for pesticide application.

5.SYSTEM DESIGN

5.1 ARCHITECTURE DESIGN

The RF model integrates machine learning and deep learning to enhance Smart Farming severity prediction. The architecture is structured in multiple stages, including data preprocessing, feature selection, model fusion, and prediction output.

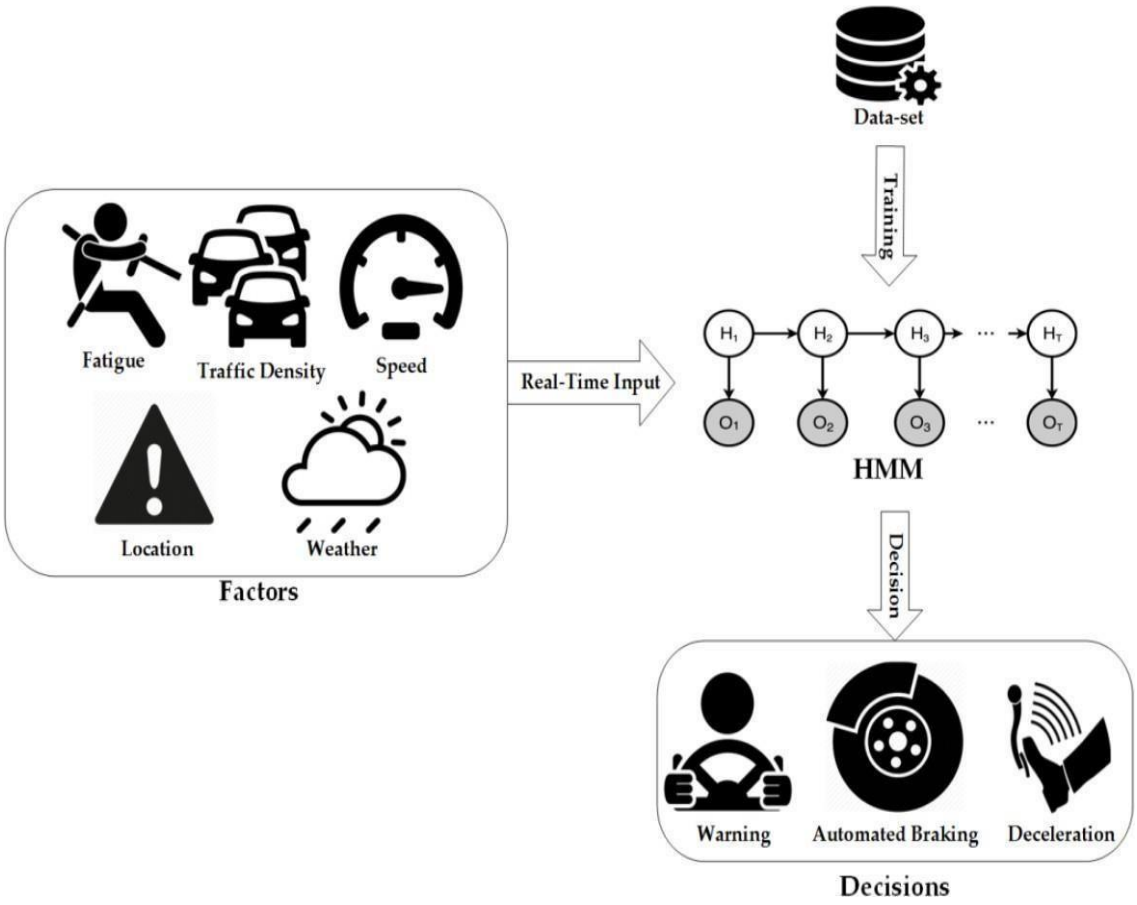


Fig 5.1.1 System Architecture of RF Model

5.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and annotation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization,

Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects

5.2.1 Class diagram

A class diagram is a UML diagram that visually represents the structure of a system by showing its classes, attributes, methods, and relationships.

The UML class diagram represents a user interacting with a dataset in an Smart Farming severity prediction system. The user begins by uploading a US road Smart Farming dataset, which is then processed to extract full or selected features. The dataset is split into training and testing sets, ensuring that the model can learn from one portion and be evaluated on another. Classifiers are run on both full and selected features to analyze their effectiveness in predicting Smart Farming severity. To evaluate performance, the system generates a comparison graph and a table, visually and numerically comparing different classifier results. Finally, the trained model predicts Smart Farming severity based on the test data, classifying Smart Farmings into different severity levels. This structured workflow enhances the efficiency of Smart Farming analysis and prediction using machine learning techniques.

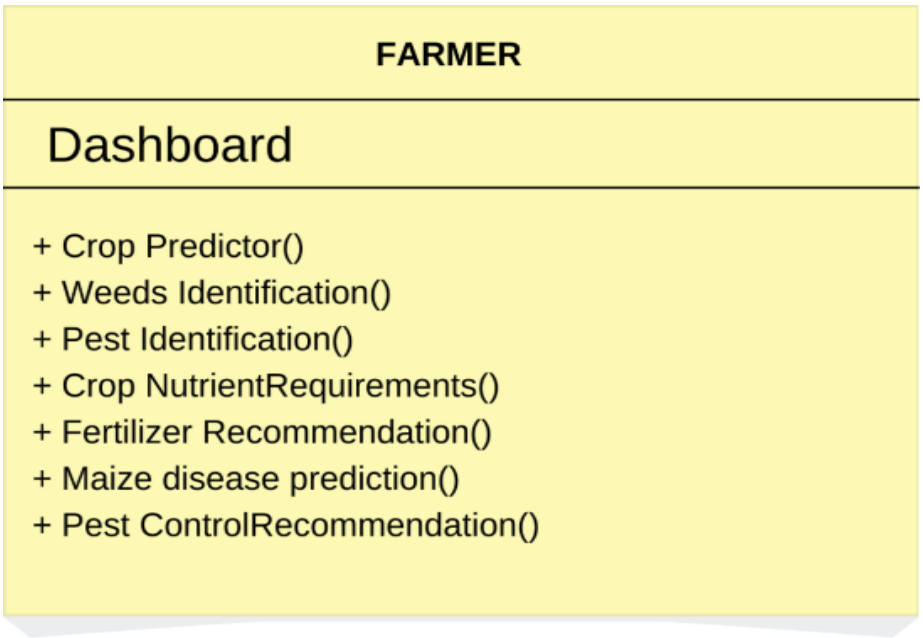


Fig 5.2.1 Class Diagram Smart Farming

5.2.2 Use case Diagram

The UML use case diagram illustrates the interaction between a user and a dataset in an Smart Farming severity prediction system. The user performs multiple actions, starting with uploading a US road dataset, which is then processed to extract full or selected features. The dataset is subsequently split into training and testing sets to ensure proper model evaluation. Various classifiers are applied to both the complete and selected features, allowing comparison of their performance. The system provides both a comparison graph and a comparison table, visually and numerically presenting the results of different classifiers. Finally, the model is used to predict Smart Farming severity from test data, classifying Smart Farmings based on severity levels. This structured workflow ensures efficient Smart Farming severity prediction using machine learning techniques.

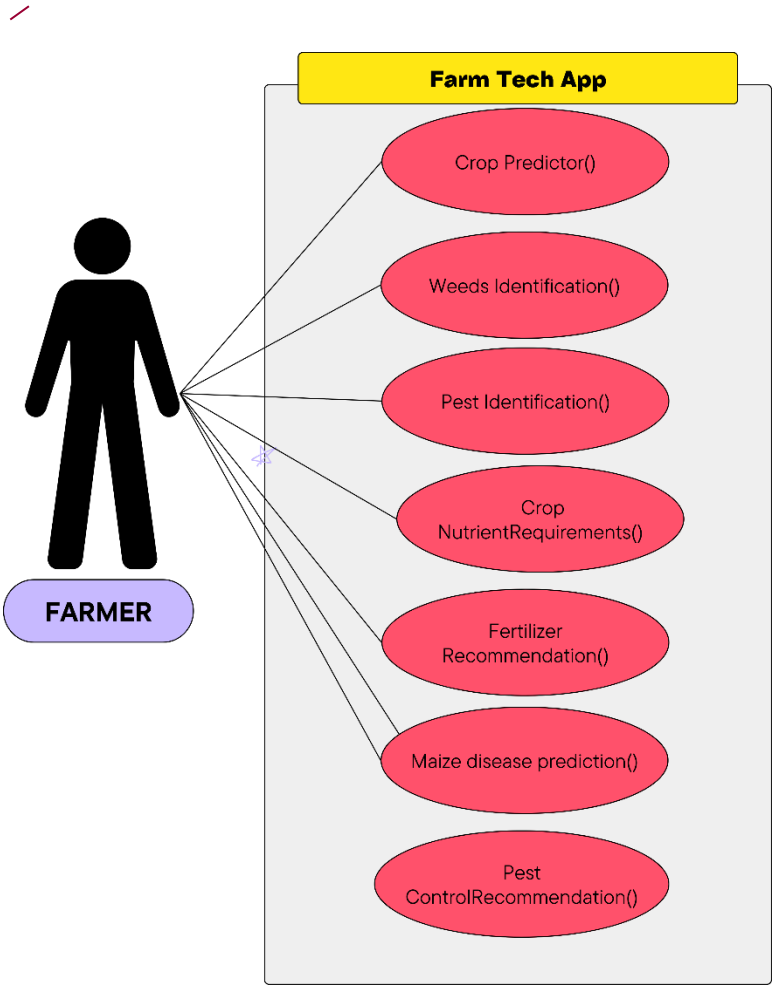


Fig 5.2.2 Use case Diagram for Smart Farming

5.2.3 Collaboration Diagram

A collaboration diagram (or communication diagram) is a UML diagram that shows object interactions and their relationships, focusing on the structural organization and message flow.

The Collaboration diagram represents the interaction between a user and a dataset in an Smart Farming severity prediction system. The process starts with the user uploading a US road dataset, followed by extracting full and selected features for analysis. The dataset is then split into training and testing sets to train machine learning models effectively. The user runs classifiers on both full and selected features, allowing a comparative analysis of their performance.

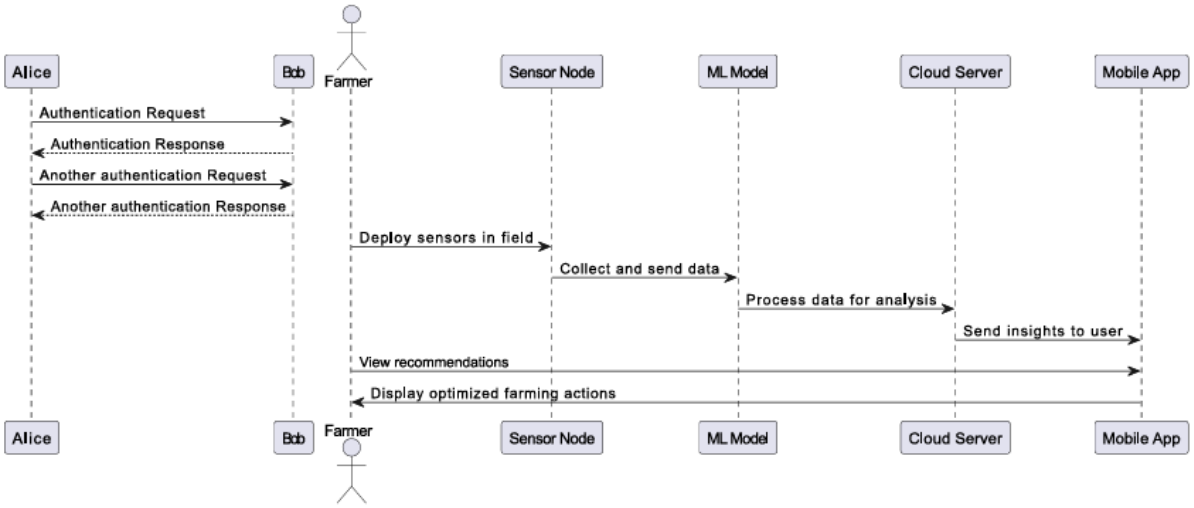


Fig 5.2.3 Collaboration Diagram of Smart Farming

5.2.4 Sequence Diagram

The sequence diagram illustrates the interaction between a user and the dataset in an Smart Farming severity prediction system. The process starts with the user uploading a US road dataset, followed by extracting full and selected features for further analysis. The dataset is then split into training and testing sets to facilitate model evaluation. The user runs classifiers on full features as well as on selected features, enabling a comparative analysis of their effectiveness. To visualize performance, the system generates a comparison graph and a comparison table, providing insights into the classifier results. Finally, the model is utilized to predict Smart Farming severity from the test data, classifying Smart Farmings into different severity levels. This structured workflow ensures a systematic and data-driven approach to Smart Farming severity prediction using machine learning techniques.

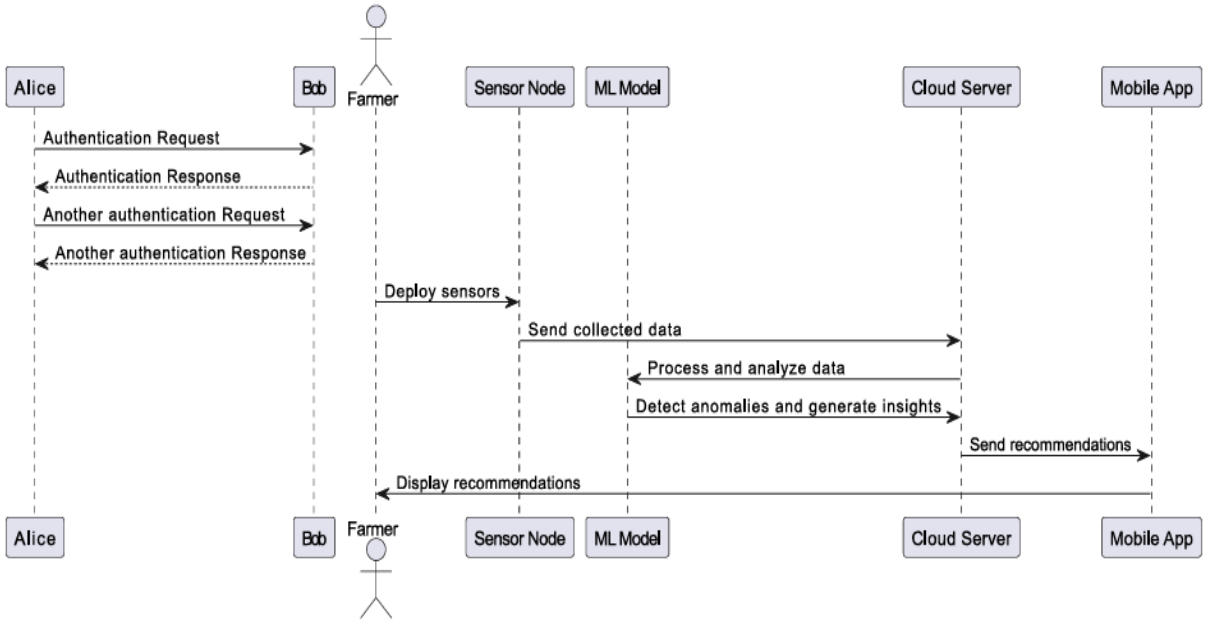


Fig 5.2.4 Sequence Diagram of Smart Farming

6.REQUIREMENT SPECIFICATIONS

6.1 REQUIREMENT ANALYSIS

Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

Operating system : Minimum Windows 7 or above

Coding Language : python 3.0 or above

Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

System : intel processor i3 or above.

Ram : Minimum 8 GB.

Hard Disk : Minimum 40 GB

6.2 SPECIFICATION PRINCIPLES

PYTHON

Python is currently the most widely used multi-purpose, high-level programming language. It supports both Object-Oriented and Procedural paradigms, making it versatile and easy to use. Python programs are generally smaller than those written in other languages like Java, requiring less typing and enforcing indentation for improved readability. Due to its simplicity and efficiency, Python is widely adopted by major tech companies such as Google, Amazon, Facebook, Instagram, Dropbox, and Uber. One of Python's biggest strengths is its extensive collection of standard libraries, which support various applications, including Machine Learning, GUI development (Kivy, Tkinter, PyQt), web frameworks like Django (used by YouTube, Instagram, and Dropbox), image processing (OpenCV, Pillow), web scraping (Scrapy, BeautifulSoup, Selenium), test frameworks, and multimedia applications.

ADVANTAGESOF PYTHON

Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++.

IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting toother more verbose languages like Java.

Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code.

Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

DISADVANTAGESOF PYTHON

So far, we've seen why Python is a great choice for your project. But if you choose it, you

should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

1. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client- side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

2. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

3. Under developed Database Access Layers

Compared to more widely used technologies like JDBC (Java Database Connectivity) and ODBC (Open Data Base Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

History of P ython

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late1980s. Guidovan Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voorWiskundeen Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my

experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types.

Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt. sources in February 1991. This release included already exception handling, functions, and the core datatypes of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossumn ever liked. Six and a half years later in October 2000, Python2.0was introduced. This release included list comprehensions, a full garbage collector and it was supporting Unicode

Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers— even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules

1. TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is asymbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

2. NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python

3. Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

4. Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc., via an object-oriented interface or via a set of functions familiar to MATLAB users.

5. Scikit– learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossumand first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python supports multiple programming paradigms, including object-oriented, imperative, functional, and procedural programming, making it versatile for different application needs. The object-oriented paradigm enables code reusability through classes and objects, while the functional programming paradigm supports higher-order functions and lambda expressions. The imperative and procedural styles allow developers to write clear and structured

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high- level programming language. Its style philosophy emphasizes code readability with its notable use of

great whitespace.

The object- oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e., operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So, the steps below are to install pythonversion3.7.4onWindows7 device or to install Python 3. Download the Python Cheat sheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4parts to help understand better.

Download the Correct version into the system

Step1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

Fig 6.2.1 Python Official Site

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Fig 6.2.2 Download page of python

Step3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Step4: Scroll down the page until you find the Files option.

Step5: Here you see a different version of python along with the operating system.

To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move a head with the second part in installing python i.e., Installation

Note: To know the changes or updates that a remade in the version you can click on the Release Note Option.

Installation of Python

Step1: Goto Download and Open the downloaded python version to carry out the installation process.

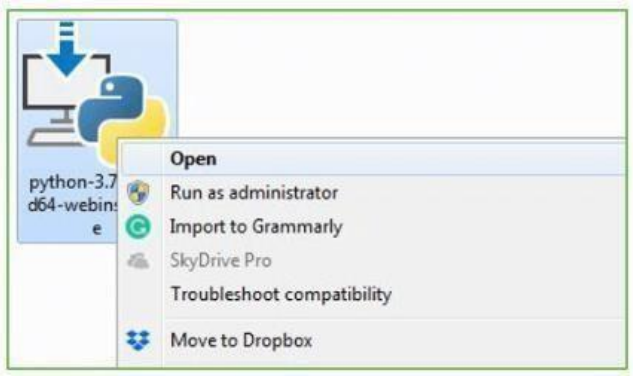


Fig 6.2.3 Python Installation

Step2: Before you click on Install Now, make sure to put a tick on Add Python3.7 to PATH.

Fig 6.2.4 Python Installation

Fig 6.2.5 Python Set Up Completed

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes. Verify the Python Installation

Step1: Click on Start

Step2: In the Windows Run Command, type “cmd”.

Step3: Click on Install NOW After the installation is successful. Click on Close.

Step3: Open the Command prompt option.

Step4: Let us test whether the python is correctly installed. Type python–V and press Enter.

Step5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

7.IMPLEMENTATION

7.1 PROJECT MODULES

The implementation of the Smart Farming Precision Agriculture (SFPA) model involves data preprocessing, feature extraction, model training, evaluation, and real-time crop yield prediction. The system integrates machine learning (Random Forest) and deep learning (CNN) to enhance accuracy in crop yield prediction and resource management. Various classifiers are compared using performance metrics like accuracy, precision, recall, and F1-score, ensuring reliable and efficient agricultural decision-making.

- **Upload Agricultural Dataset**

  Import data related to soil health, weather conditions, crop types, and sensor data.

- **Extract Full Set of Features**

  Perform feature engineering to extract relevant parameters such as moisture levels, nutrient content, and temperature data.

- **Split Data into Training and Testing Sets**

  Divide the dataset into training and testing subsets to validate model performance.

- **Run Classifiers on Full Features**

  Apply machine learning models (e.g., Random Forest, SVM) using all extracted features to assess initial performance.

- **Run Classifiers on Selected Features**

  Optimize model accuracy by selecting the most significant features through techniques like feature importance and dimensionality reduction.

- **Comparison Graph**

  Visualize the performance of different classifiers using graphs that show accuracy, precision, recall, and F1-score.

- **Comparison Table**

Tabulate the performance metrics of each classifier for easy comparison.

- **Predict Crop Yield from Table Data**

Use the trained model to predict crop yields based on new agricultural data inputs

7.2 MODULE DESCRIPTION

7.2.1 Data preprocessing

Data Cleaning

Duplicate Removal: Smart Farming data is often sourced from multiple platforms, leading to redundant entries. Eliminating duplicates ensures that each Smart Farming is counted only once, preventing bias in model predictions.

Filtering Irrelevant Features: Certain attributes, such as Pesticide ID order descriptive location details, may not contribute to severity prediction. Removing such non-informative features, identified through exploratory data analysis, improves dataset efficiency.

Handling Missing Data: Smart Farming frequently contain incomplete records, such as missing weather or vehicle details. Selecting appropriate imputation methods, such as replacing missing values with regional averages, or removing records with excessive gaps.

7.2.2 Classification Models Used in the Proposed System

Random Forest (RF): Feature Selection & Prediction Random Forest is an ensemble bagging technique that constructs multiple decision trees and predicts outcomes based on majority voting. RF is highly effective for Smart Farming severity prediction due to high accuracy in structured data classification is achieved by leveraging advanced machine learning techniques that effectively analyze patterns within organized datasets. The ability to identify significant features influencing Smart Farming severity enhances model interpretability, allowing for better decision- making and risk assessment. Additionally, the reduction in overfitting is ensured by averaging multiple tree predictions, which helps in improving generalization and ensuring that the model performs well on unseen data.

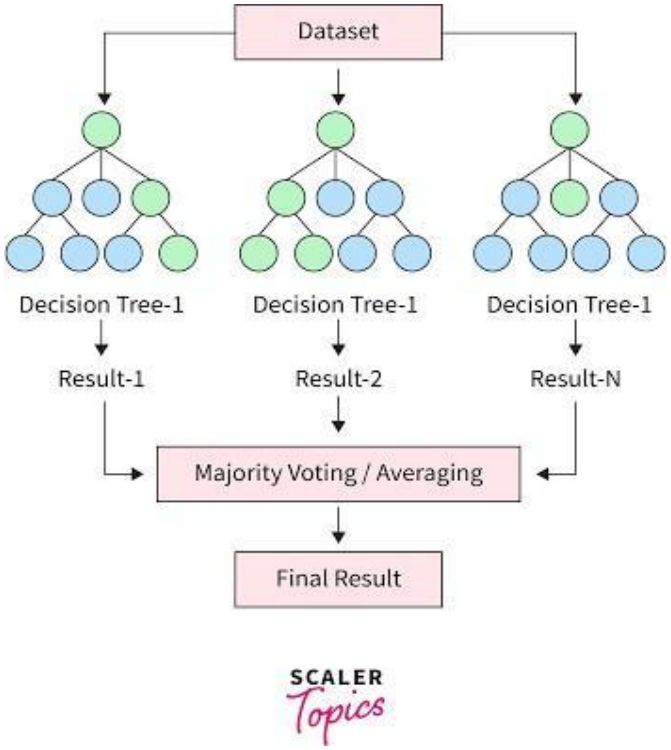


Fig 7.2.2.1 Random Forest Algorithm Diagram

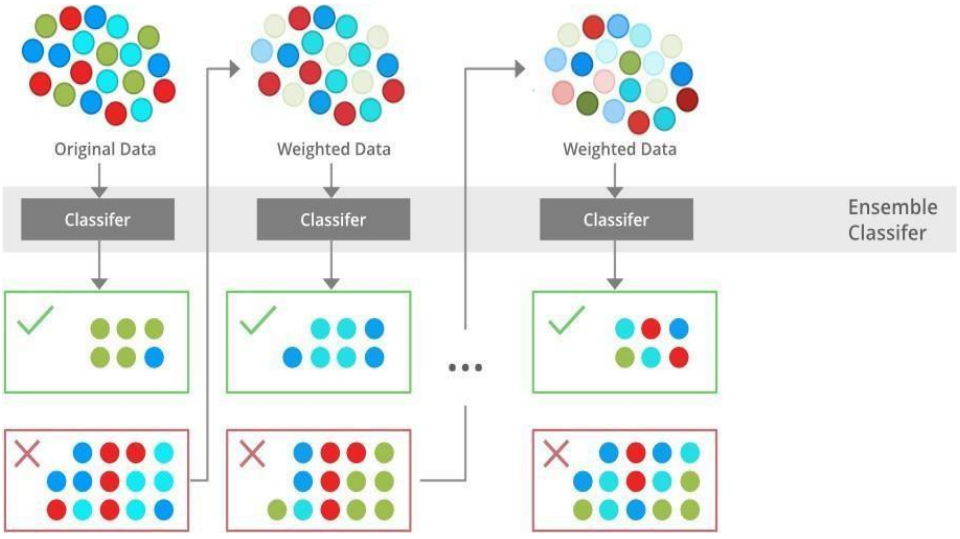


Fig 7.2.2.2 Ada Boost Classifier Diagram

AdaBoost Classifier (AC): Boosting-Based Classification, particularly Adaptive Boosting (AdaBoost),is an ensemble learning technique that enhances the performance of weak classifiers by training them sequentially. Each new classifier in the sequence focuses more on the

misclassified instances from previous iterations, improving overall accuracy. The technique dynamically adjusts weights, giving more importance to difficult samples, ensuring that challenging cases receive more attention. AdaBoost is especially effective for imbalanced datasets, such as Smart Farming severity prediction, where severe Smart Farmings occur less frequently, thereby improving model robustness and reliability.

Extra Trees Classifier (ETC): Extra Trees Classifier (ETC) enhances performance by introducing randomized node splitting while constructing decision trees. Unlike Random Forest (RF), ETC selects split points randomly, leading to faster training and reducing computational cost. This randomness helps in lowering variance while maintaining high accuracy, making it a more efficient alternative. Additionally, ETC achieves better generalization compared to standard RF classifiers, making it suitable for large datasets where both speed and accuracy are crucial.

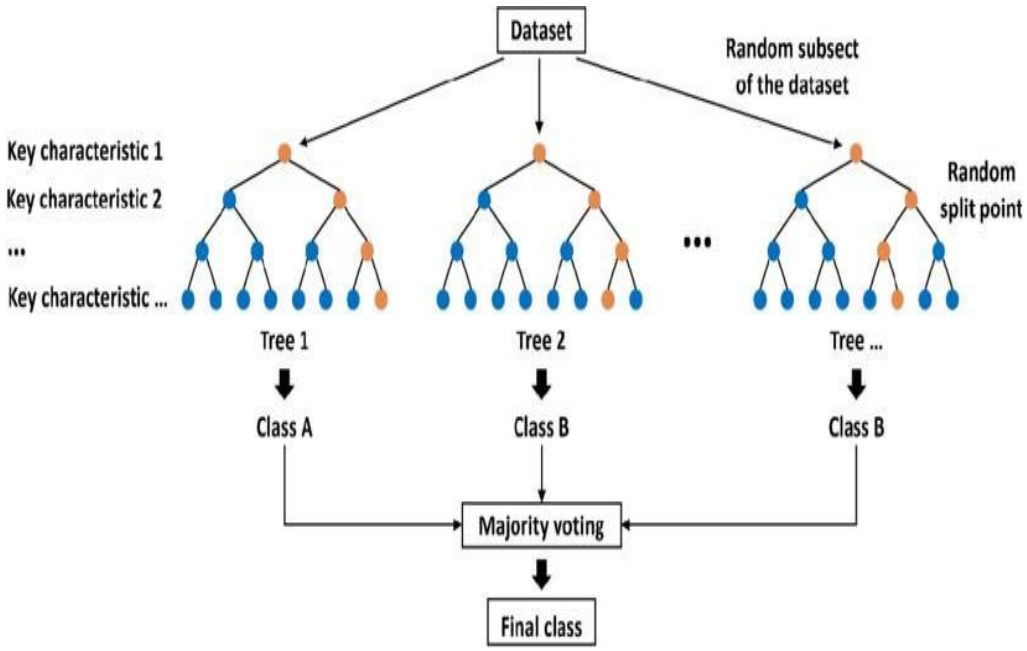


Fig 7.2.2.3 Extra Trees Classifier

Gradient Boosting Machine (GBM): Gradient Boosting Machine (GBM) is a sequential learning model that builds decision trees iteratively, where each tree corrects the errors made by the previous ones. It employs gradient descent to minimize classification errors, making it highly effective in structured Smart Farming data analysis. GBM efficiently optimizes performance by capturing complex non-linear relationships, enhancing Smart Farming severity prediction accuracy. Its ability to learn from previous mistakes makes it a powerful technique for predictive modeling in agricultural Smart Farming analysis.

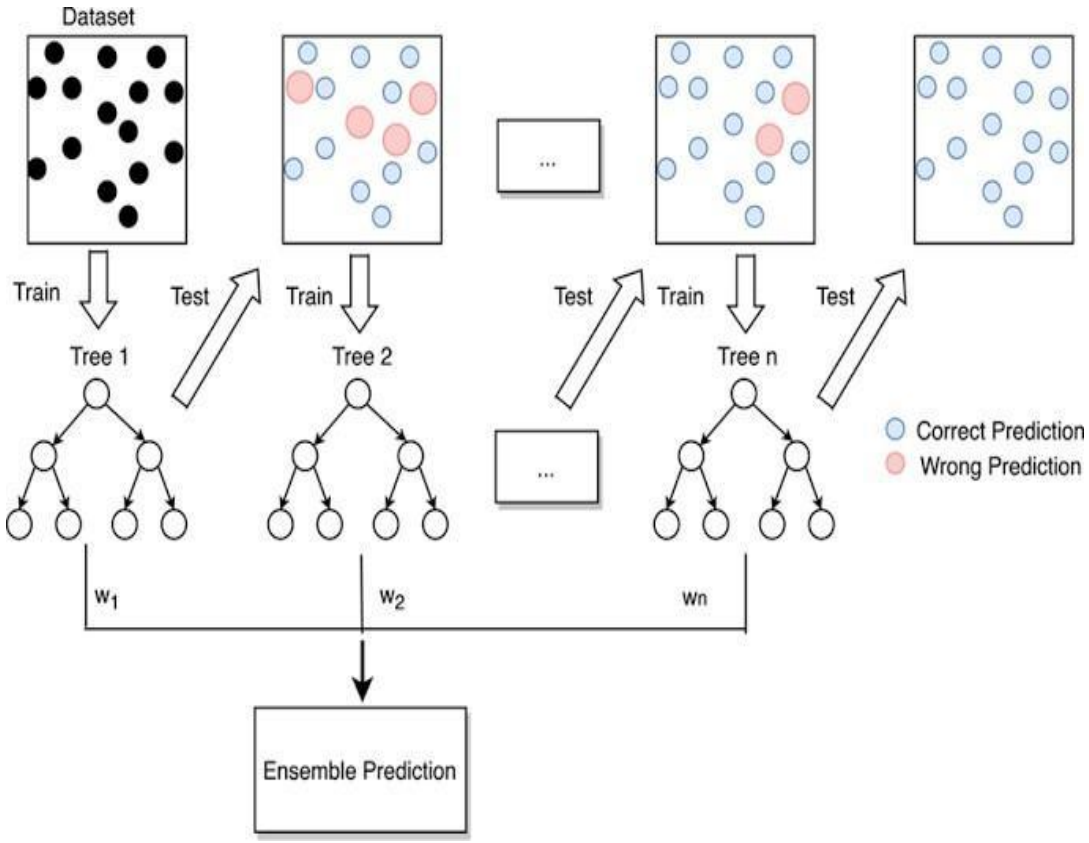


Fig 7.2.2.4 Gradient Boosting Machine (GBM)

Convolutional Neural Network (CNN): Convolutional Neural Networks (CNN) are deep learning models designed for high-dimensional feature extraction, making them highly effective in Smart Farming severity prediction. The CNN architecture used in this system includes convolutional layers that identify spatial patterns in Smart Farming data, pooling layers that reduce dimensionality while retaining essential information, dropout layers that prevent overfitting by randomly deactivating neurons, and flatten layers that prepare extracted features for classification. The ReLU activation function and a 0.2 dropout rate are applied to enhance performance and generalization. CNN effectively captures intricate relationships between Smart Farming severity, weather conditions, and road attributes, improving predictive accuracy.

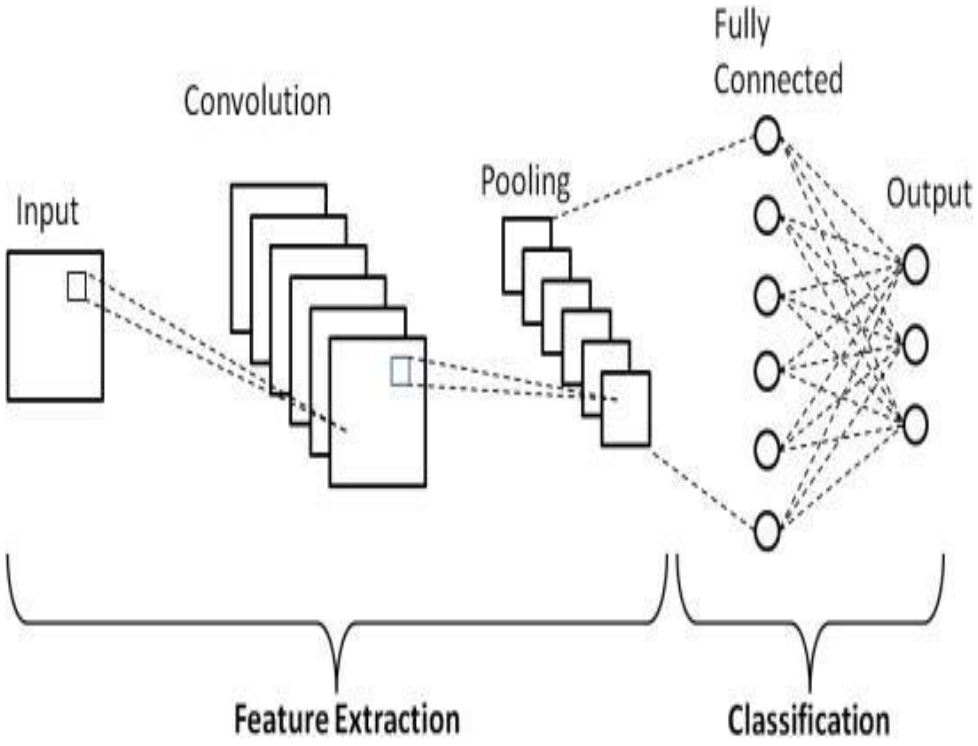


Fig 7.2.2.5 Convolutional Neural Network (CNN)

Voting Classifier (LR+SGD): The Hybrid Model for structured data utilizes a Voting Classifier that integrates Logistic Regression (LR) and Stochastic Gradient Descent (SGD) to enhance classification accuracy. Logistic Regression employs logit functions to estimate Smart Farming severity probabilities, ensuring reliable probabilistic predictions. Meanwhile, Stochastic Gradient Descent optimizes the model by iteratively updating weights, leading to improved learning efficiency and reduced computational cost. By combining these approaches, the model achieves better generalization and robustness in Smart Farming severity prediction.

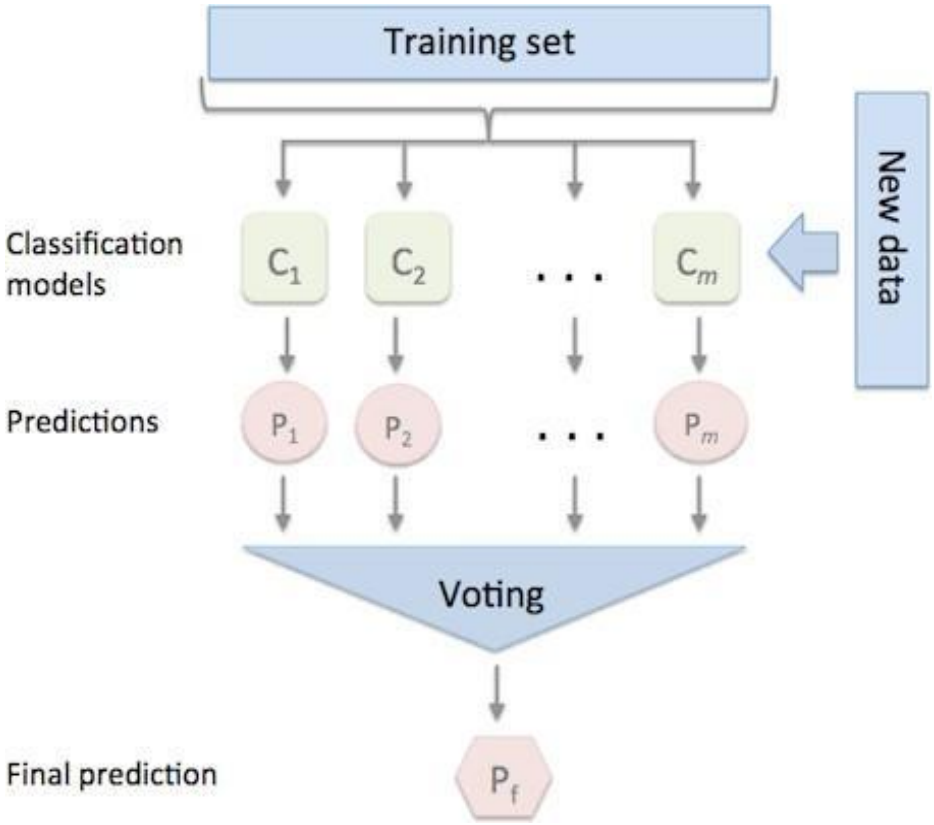


Fig 7.2.2.6 Voting Classifier for RF Model

Soft Voting Mechanism for Decision Making

The proposed system employs a soft voting mechanism, which averages prediction probabilities from different models and selects the class with the highest probability score as the final output. This method enhances classification accuracy by ensuring that predictions from multiple models are combined to improve reliability.

The formula for soft voting is:

$$p = argmax(\sum i = 1nRFi + \sum i = 1nCNNi)p = \text{argmax}\left(\sum_{i=1}^{n} RFi + \sum_{i=1}^{n} CNNi\right)p = argmax(i = 1\sum nRFi + i = 1\sum nCNNi)\begin{pmatrix} a \\ b \end{pmatrix}$$

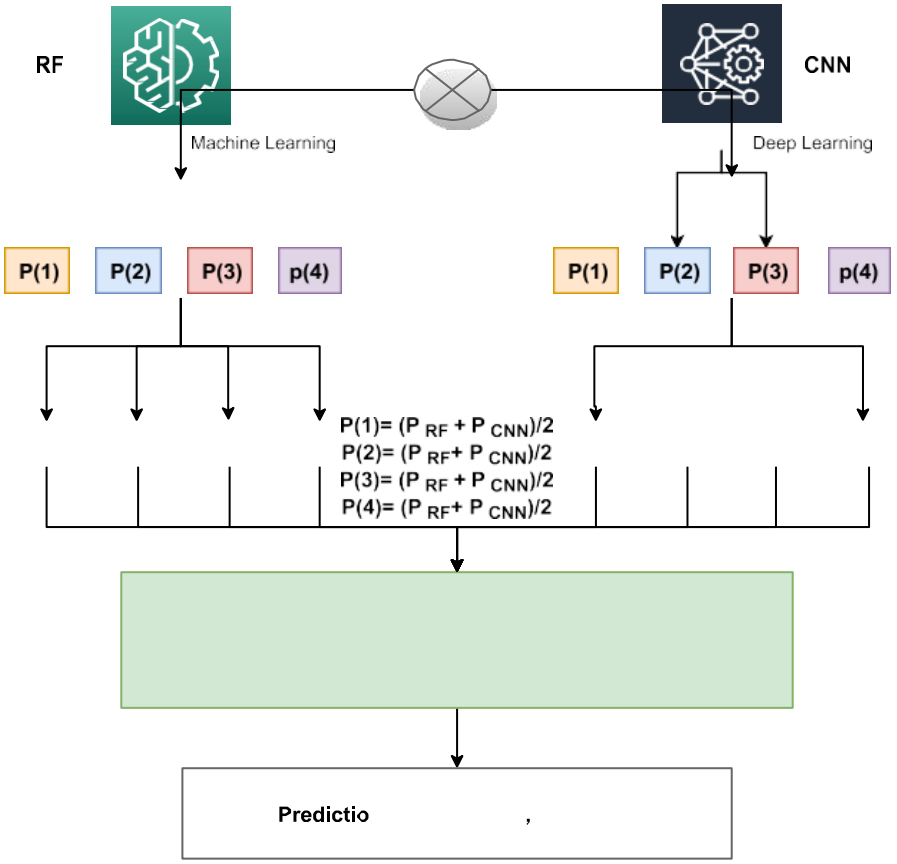


Fig 7.2.2.7 Working of the RF Model

7.2.3 Performance Evaluation Metrics

To assess the effectiveness of the Ensemble Fusion Classifier (EFC) model in predict in Smart Farming severity, several performance evaluation metrics are used. These metrics help determine how well the model classifies Smart Farming severity levels and compare its performance with other models

1. Precision

Precision (also called Positive Predictive Value) evaluates how many of the predicted severe Smart Farmings were actually severe. It is given by:

$$Precision = TP/TP + FP$$

A high precision score means that false positives (mis classifications of non-severe Smart Farmings as severe) are minimized, making the model more reliable for decision-making.

2. Recall(Sensitivity)

Recall(also known as True Positive Rate)measures how well the model identifies actual severe Smart Farmings. It is calculated as:

Ahighrecall value ensures that the model correctly captures most severe Smart Farming cases, reducing therisk of underestimating dangerous situations.

3. F1-Score

F1-score is the harmonic mean of precision and recall,balancing both metrics when the dataset is imbalanced. It is computed as:

$$F1 - Score = 2 * Precision * Recall/Precision + Recall$$

7.3 SOURCE CODE

```
from fastapi import FastAPI, File, UploadFile
from pydantic import BaseModel
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from sklearn.preprocessing import LabelEncoder
import io
import torch
import torchvision.transforms as transforms
from torchvision import models
import joblib  # Import for loading the crop recommendation model
from PIL import Image
app = FastAPI()
# Load the trained models
modelWeed = load_model("weed_detection_model.h5")
modelPest = load_model("pestIdentification.h5")
#modelCrop = joblib.load("CropRecommendetion_RF_Model.pkl")  # Using
joblib to load the .pkl file
modelCrop = joblib.load("random_forest_model1.pkl")
modelCropNuts = joblib.load("random_forest_model_soil_Nutrients.pkl")
modelFerti = joblib.load("bagging_model_Fertilizer.pkl")
# Device Configuration
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")
# Define transformations (same as training)
transform = transforms.Compose([
transforms.Resize((224, 224)),
transforms.ToTensor(),
transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225])  # ResNet normalization
])
```

```
# Load the trained ResNet model for maize disease detection
num_classes_maize = 7  # Update this based on the number of classes
in your dataset
modelMaize = models.resnet18(pretrained=False)  # Use the correct
torchvision model
modelMaize.fc = torch.nn.Linear(modelMaize.fc.in_features,
num_classes_maize)  # Adjust output layer
modelMaize.load_state_dict(torch.load("maize_classifier1.pth",
map_location=device))  # Load trained weights
modelMaize.to(device)
modelMaize.eval()  # Set model to evaluation mode
# Load the trained ResNet model for Tomato disease detection
num_classes_tomato = 5  # Update this based on the number of classes
in your dataset
modelTomato = models.resnet18(pretrained=False)  # Use the correct
torchvision model
modelTomato.fc = torch.nn.Linear(modelTomato.fc.in_features,
num_classes_tomato)  # Adjust output layer
modelTomato.load_state_dict(torch.load("Tomato_model_25.pth",
map_location=device))  # Load trained weights
modelTomato.to(device)
modelTomato.eval()  # Set model to evaluation mode
# Define class names (ensure this matches your training dataset)
class_names_maize = [
    "fall armyworm", "grasshopper", "healthy", "leaf beetle",
    "leaf blight", "leaf spot", "streak virus"
]
# Define class names (ensure this matches your training dataset)
class_names_tomato = [
    "verticulium wilt", "healthy", "leaf blight",
    "leaf curl", "septoria leaf spot"
]
# Function to preprocess image
```

```
def preprocess_image_torch(img_bytes):
img = Image.open(io.BytesIO(img_bytes)).convert("RGB")  # Ensure RGB
format
img = transform(img).unsqueeze(0)  # Apply transformations and add
batch dimension
    return img.to(device)
#class DummyModel:
#    def predict(self, X):
#        return [0]  # Always returns 0 for testing (index of "Urea")

#modelFerti = DummyModel()

# Define class names
class_names_weed = [
    "Black-grass", "Charlock", "Cleavers", "Common Chickweed",
"Common wheat",
    "Fat Hen", "Loose Silky-bent", "Maize", "Scentless Mayweed",
    "Shepherd’s Purse", "Small-flowered Cranesbill", "Sugar beet"
]

class_names_pest = [
    "aphids", "armyworm", "beetle", "bollworm", "grasshopper",
    "mites", "mosquito", "sawfly", "stem_borer"
]

crop_mapping = {
    'rice': 1, 'maize': 2, 'chickpea': 3, 'kidneybeans': 4,
'pigeonpeas': 5,
    'mothbeans': 6, 'mungbean': 7, 'blackgram': 8, 'lentil': 9,
'pomegranate': 10,
    'banana': 11, 'mango': 12, 'grapes': 13, 'watermelon': 14,
'muskmelon': 15,
    'apple': 16, 'orange': 17, 'papaya': 18, 'coconut': 19, 'cotton':
20,
```

```
    'jute': 21, 'coffee': 22
}

crop_Nuts_mapping = {
    'pomegranate': 1, 'mango': 2, 'grapes': 3,
    'mulberry': 4, 'ragi': 5, 'potato': 6
}

crop_classes = [
    'Maize', 'Sugarcane', 'Cotton', 'Tobacco', 'Paddy', 'Barley',
'Wheat',
    'Millets', 'Oil seeds', 'Pulses', 'Ground Nuts'
]

soil_classes = ['Sandy', 'Loamy', 'Black', 'Red', 'Clayey']

fertilizer_classes = ['Urea', 'DAP', '14-35-14', '28-28', '17-17-17',
'20-20', '10-26-26']



# Create label encoders for crop and soil types
crop_encoder = LabelEncoder()
soil_encoder = LabelEncoder()

# Fit the encoders with predefined classes
crop_encoder.fit(crop_classes)
soil_encoder.fit(soil_classes)

# Reverse the crop mapping to get names from numerical predictions
crop_labels = {v: k for k, v in crop_mapping.items()}

# Reverse the crop mapping for nutrients to get names from numerical
predictions
crop_Nuts_labels = {v: k for k, v in crop_Nuts_mapping.items()}
```

```
# Reverse mapping for fertilizer names if needed
fertilizer_labels = {0: "Urea", 1: "DAP", 2: "14-35-14", 3: "28-28-
28", 4: "Urea"}  # Adjust based on training data

# Define input schema using Pydantic
class CropInput(BaseModel):
    n: float
    p: float
    k: float
    temp: float
    humidity: float
ph: float
    rainfall: float
soilType: int

class CropNutsInput(BaseModel):
    n: float
    p: float
    k: float
ph: float
ec: float
    s: float
    cu: float
fe: float
mn: float
zn: float
    b: float

class FertilizerInput(BaseModel):
    temperature: float
    humidity: float
    moisture: float
soil_type: str
```

```
crop_type: str
    nitrogen: float
    potassium: float
    phosphorous: float


def preprocess_image(img_bytes):
    """Convert image bytes to a preprocessed model input."""
img = Image.open(io.BytesIO(img_bytes)).convert("RGB")  # Convert to
RGB
img = img.resize((224, 224))  # Resize to match model input size
img_array = np.array(img) / 255.0  # Normalize
img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
    return img_array


@app.post("/predict-weed")
async def predict_weed(file: UploadFile = File(...)):
    try:
img_bytes = await file.read()
processed_img = preprocess_image(img_bytes)

        # Get prediction
        prediction = modelWeed.predict(processed_img)
predicted_class = np.argmax(prediction)
        confidence = float(np.max(prediction))

        return {"predicted_class": class_names_weed[predicted_class],
"confidence": confidence}
    except Exception as e:
        return {"error": str(e)}


@app.post("/predict-pest")
```

```
async def predict_pest(file: UploadFile = File(...)):  # Changed
function name
    try:
img_bytes = await file.read()
processed_img = preprocess_image(img_bytes)

        # Get prediction
        prediction = modelPest.predict(processed_img)
predicted_class = np.argmax(prediction)
        confidence = float(np.max(prediction))

        return {"predicted_class": class_names_pest[predicted_class],
"confidence": confidence}
    except Exception as e:
        return {"error": str(e)}

@app.post("/predict-maize")
async def predict_maize(file: UploadFile = File(...)):
    try:
img_bytes = await file.read()
processed_img = preprocess_image_torch(img_bytes)

        # Perform inference
        with torch.no_grad():
            prediction = modelMaize(processed_img)

predicted_class = torch.argmax(prediction, dim=1).item()
        confidence = torch.softmax(prediction,
dim=1)[0][predicted_class].item()

        return {"predicted_class":
class_names_maize[predicted_class], "confidence": round(confidence,
4)}
```

```
    except Exception as e:
        return {"error": str(e)}

@app.post("/predict-tomato")
async def predict_tomato(file: UploadFile = File(...)):
    try:
img_bytes = await file.read()
processed_img = preprocess_image_torch(img_bytes)

        # Perform inference
        with torch.no_grad():
            prediction = modelTomato(processed_img)

predicted_class = torch.argmax(prediction, dim=1).item()
        confidence = torch.softmax(prediction,
dim=1)[0][predicted_class].item()

        return {"predicted_class":
class_names_tomato[predicted_class], "confidence": round(confidence,
4)}

    except Exception as e:
        return {"error": str(e)}

@app.post("/predict")
def predict_best_crops(input_data: CropInput):
    try:
        # Convert input to numpy array
input_features = np.array([
input_data.n, input_data.p, input_data.k,
input_data.temp, input_data.humidity,
            input_data.ph, input_data.rainfall, input_data.soilType
        ]).reshape(1, -1)
```

```
        # Predict probabilities for all crops
        probabilities = modelCrop.predict_proba(input_features)[0]

        # Get top 5 crop indices with highest probabilities
        top_5_indices = np.argsort(probabilities)[::-1][:5]

        # Filter crops with probability > 0%
        top_5_crops = [{"crop": crop_labels[i+1], "probability":
round(probabilities[i], 4)}
                       for i in top_5_indices if probabilities[i] > 0]

        return {"recommended_crops": top_5_crops}
    except Exception as e:
        return {"error": str(e)}



@app.post("/predictCropNuts")
def predict_best_crops(input_data: CropNutsInput):
    try:
        # Convert input to numpy array
input_features = np.array([
input_data.n, input_data.p, input_data.k,
            input_data.ph, input_data.ec, input_data.fe,
input_data.s, input_data.cu, input_data.mn, input_data.zn,
input_data.b,
        ]).reshape(1, -1)

        # Predict probabilities for all crops
        probabilities =
modelCropNuts.predict_proba(input_features)[0]

        # Get top 5 crop indices with highest probabilities
        top_5_indices = np.argsort(probabilities)[::-1][:5]
```

```
        # Filter crops with probability > 0%
        top_5_crops = [{"crop": crop_Nuts_labels[i+1], "probability":
round(probabilities[i], 4)}
                       for i in top_5_indices if probabilities[i] > 0]

        return {"recommended_crops": top_5_crops}
    except Exception as e:
        return {"error": str(e)}


@app.post("/predict-fertilizer")
def predict_fertilizer(input_data: FertilizerInput):
    try:
        # Validate Crop Type
        if input_data.crop_type not in crop_classes:
            return {"error": f"Invalid Crop Type:
{input_data.crop_type}. Must be one of {crop_classes}"}

        # Validate Soil Type
        if input_data.soil_type not in soil_classes:
            return {"error": f"Invalid Soil Type:
{input_data.soil_type}. Must be one of {soil_classes}"}

        # Encode categorical variables
encoded_crop = crop_classes.index(input_data.crop_type)
encoded_soil = soil_classes.index(input_data.soil_type)

        # Prepare input feature array
input_features = np.array([
input_data.temperature, input_data.humidity, input_data.moisture,
input_data.nitrogen, input_data.potassium,
input_data.phosphorous, encoded_soil, encoded_crop
        ]).reshape(1, -1)
```

```
        # Predict fertilizer
recommended_fertilizer = modelFerti.predict(input_features)[0]


        #predicted_fertilizer = fertilizer_classes[prediction_index]
        return {"recommended_fertilizer": recommended_fertilizer}


    except Exception as e:
        return {"error": str(e)}
```

8. SYSTEMTEST

SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner.

8.1 Unit Testing

Unit testing involves testing individual components of the software to ensure they function correctly. It verifies that internal logic works as expected, with valid inputs producing valid outputs. In RF: Agricultural Smart Farming Severity Prediction, unit tests check the correctness of feature extraction, data preprocessing, and individual classifier implementations before integration.

8.2 Integration Testing

Integration testing validates that multiple components of the software interact correctly when combined. It ensures that integrated modules work as expected. In RF, this testing ensures that machine learning models, data processing pipelines, and UI components integrate seamlessly without conflicts.

8.3 Functional Testing

Functional testing ensures that the software meets business and technical requirements. It validates the correctness of input handling, expected outputs, and feature execution. In RF, this involves testing functions such as dataset loading, feature extraction, classifier execution, and result visualization to ensure expected behavior.

8.4 System Testing

System testing verifies that the complete, integrated system meets all functional and non-

functional requirements. It ensures correct system configuration and end-to-end workflows. In RF, this involves checking whether the full model pipeline—from data input to severity prediction—operates without issues and produces accurate results.

8.5 White Box Testing

White box testing examines the internal logic and structure of the code. It helps in identifying logical errors and optimizing the implementation. In RF, white box testing focuses on the implementation of classifiers, data processing logic, and feature selection methods to verify proper execution.

8.6 Black Box Testing

Black box testing validates the software without knowledge of its internal workings. It focuses on user interactions and expected results. In RF, this includes testing the user interface, verifying input data formats, and ensuring correct predictions without inspecting the internal model structures.

9.RESULTS AND DISCUSSION

The proposed work is a web interface through which the user can access the models efficiently. Represents the login page through which the user needs to login with their credentials to access the models. shows the user dashboard after logging in, thus enabling users to access the models. The results of each model are discussed in each section.

Crop recommendation

Ten Algorithms were used for crop recommendation. The accuracy score above 90% was selected as shown in figure. The accuracy drifts are been clearly observed using For these selected algorithms Hyperparameter Tuning was applied from which best model with highest accuracy was obtained as shown in figure.

According to the Random Forest classifier hyper tuned with Randomized CV is opted as best model since its accuracy is 95.45% and stored as a pickle file for further analysis.

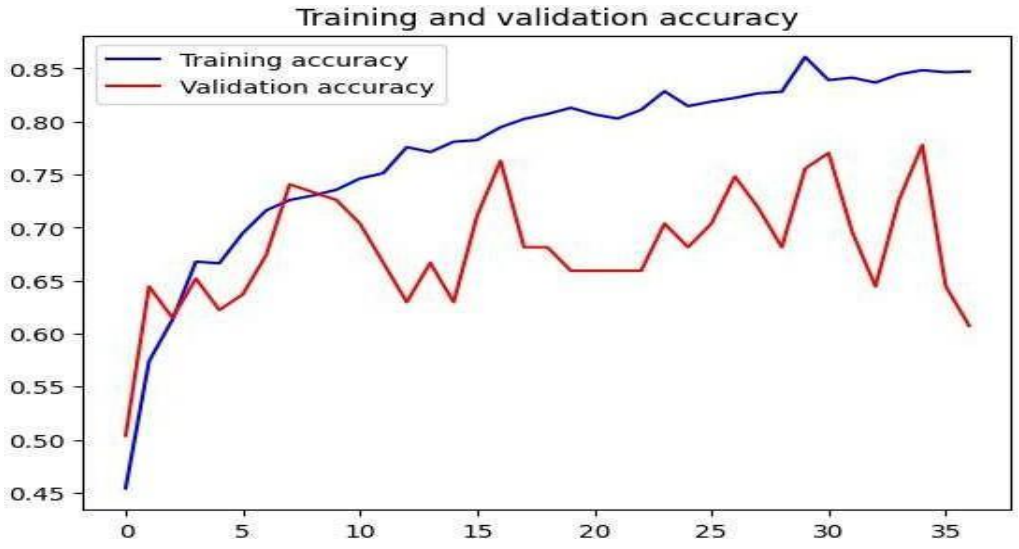


Fig 9.1 Training and validation accuracy

Training and Validation Accuracy

This section presents the training and validation accuracy trends observed during model training. The figure below illustrates how the accuracy evolved over the epochs for both training and validation data.

Observations

- The training accuracy (blue line) shows a steady increase over the epochs, indicating that the model is learning from the training data effectively.
- The validation accuracy (red line), however, fluctuates significantly, which suggests possible overfitting. While it initially improves, the variations indicate that the model may not generalize well to unseen data.
- Around the middle epochs, the validation accuracy does not show a consistent upward trend, which could be a sign of model complexity or insufficient regularization.

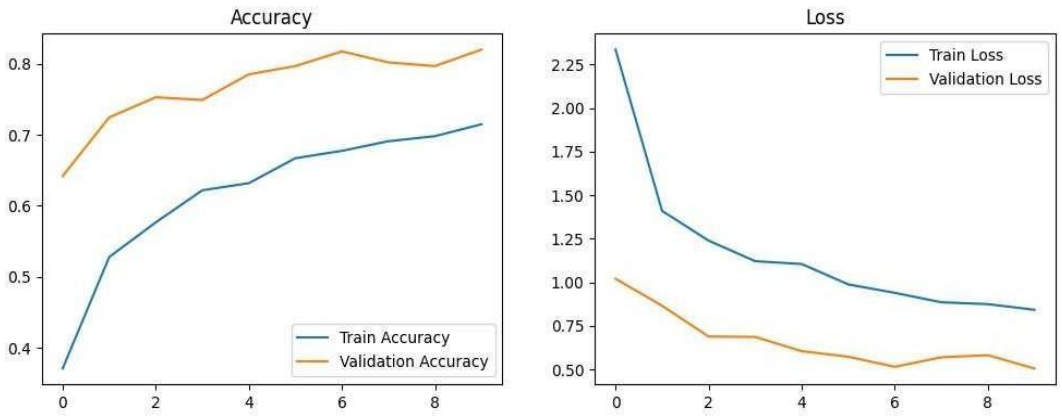


Fig 9.2 Accuracy and validation loss

Accuracy Plot (Left Graph)

- The blue line represents training accuracy, and the orange line represents validation accuracy.
- Training accuracy shows a steady increase, indicating that the model is learning from the training data.

- Validation accuracy starts higher than training accuracy and continues to increase, but it seems to stabilize toward the later epochs.

- The fact that validation accuracy is consistently higher than training accuracy may indicate the use of data augmentation or regularization techniques that improve generalization.

Loss Plot (Right Graph)

- The blue line represents training loss, and the orange line represents validation loss. o Both training and validation loss decrease over time, which is a good sign that the model is learning effectively.

- The gap between training and validation loss is small, suggesting that the model is not heavily over fitting.

- There is a slight fluctuation in validation loss towards the later epochs, which may indicate some variability in generalization but not necessarily overfitting.

Key Takeaways:

- The model appears to be well-trained with a good balance between training and validation accuracy.

- There is no major sign of over fitting, as validation performance remains strong and loss continues to decrease.

- If further optimization is needed, consider fine-tuning hyperparameters like learning rate, batch size, or adding early stopping to prevent unnecessary training beyond the optimal point.

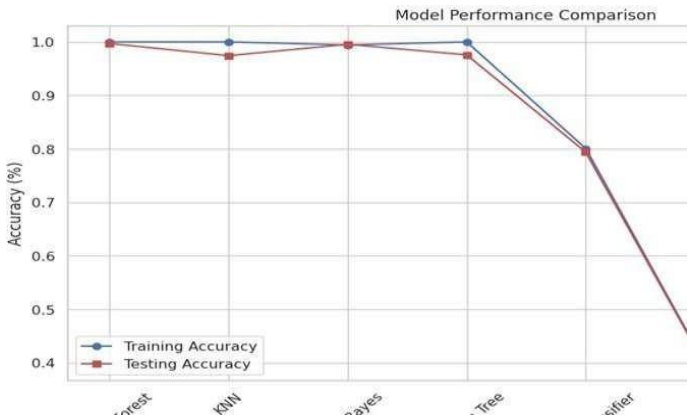


Fig 9.3 Model Performance Comparison

The image presents a Model Performance Comparison chart, displaying the training and testing accuracy of various machine learning models. Here's a breakdown of the insights:

Key Observations:

Models with High Performance:

- Random Forest, KNN, Naïve Bayes, Decision Tree, Gradient Boosting, and XGBoost exhibit high accuracy in both training and testing, indicating good generalization.
- Their training and testing accuracy values are close to 1.0 (or 100%), meaning they perform well without significant overfitting.

Overfitting & Generalization Issues:

- Bagging Classifier shows a slight drop in testing accuracy compared to training accuracy, but the difference is not substantial.

Poor Performance of AdaBoost:

- AdaBoost shows a significant drop in accuracy, with testing accuracy around 0.4 (40%).

- This suggests that AdaBoost may not be performing well on this dataset, potentially due to underfitting or sensitivity to noisy data.
- Further investigation, such as tuning hyperparameters or feature selection, might be needed to improve its performance.

Key Takeaways:

- Most models show high training and testing accuracy, meaning they generalize well.
- AdaBoost performs poorly, indicating potential underfitting or dataset incompatibility.
- If overfitting is a concern in some models (e.g., Decision Tree or Random Forest),
- regularization techniques like pruning or hyper parameter tuning might help.

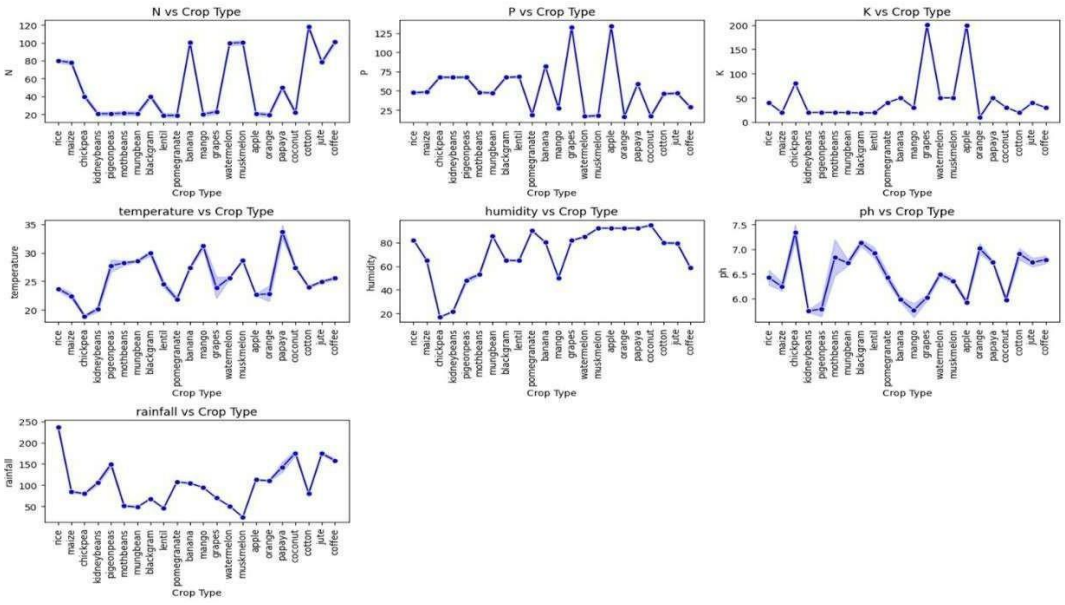


Fig 9.4 Six factors of various crop types

The image contains **six-line plots** comparing different environmental and soil-related factors across various crop types. Here's a detailed explanation of each plot:

Key Observations from the Graphs

N vs Crop Type (Top Left)

- The nitrogen (N) content varies significantly across different crops.
- Some crops, like rice and maize, have higher nitrogen requirements, while others, like coffee, have lower nitrogen levels.

P vs Crop Type (Top Middle)

- Phosphorus (P) levels also fluctuate depending on the crop.
- Some crops like papaya and banana appear to need higher phosphorus, while others require relatively low amounts.

K vs Crop Type (Top Right)

- Potassium (K) levels show significant variation, with some crops requiring very high amounts (e.g., coconut) while others require minimal potassium.
- This suggests that potassium plays a crucial role in specific crop growth.

Temperature vs Crop Type (Middle Left)

- Temperature preferences vary widely among crops.
- Some crops, such as pomegranate and banana, prefer warmer conditions, while others, like coffee, thrive at lower temperatures.

Humidity vs Crop Type (Middle Right)

- Humidity levels show clear distinctions among different crops.
- Certain crops require high humidity (e.g., tropical crops like banana), while others can grow in lower humidity conditions.

pH vs Crop Type (Bottom Middle)

- The soil pH levels vary, but most crops appear to thrive in a range between 6.0 and 7.5, indicating slightly acidic to neutral soil conditions.

Rain fall vs Crop Type (Bottom Left)

- Rain fall requirements are diverse, with some crops needing significant water supply (e.g., rice and banana) and others thriving in relatively dry conditions (e.g., mustard).

Key Takeaways:

- Different crops have unique environmental and nutrient requirements, emphasizing the importance of precision agriculture and soil testing before planting.
- Understanding these variations can optimize fertilizer application, irrigation, and land management for better yield.
- If a crop has high nutrient or water requirements, farmers must ensure sufficient supply through
- fertilization and irrigation techniques.

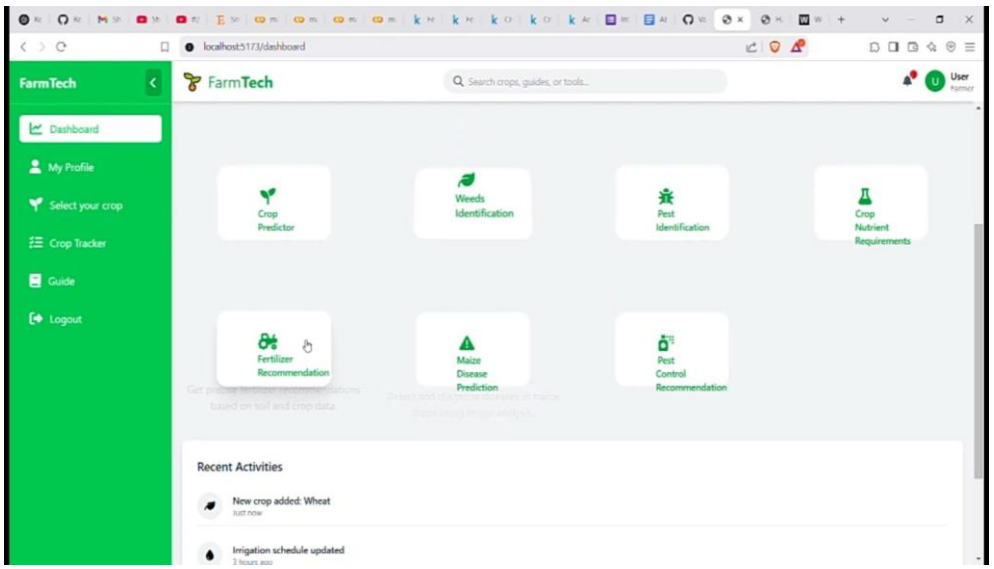
Fram Tech APP:

The FarmTech is an AI-driven application designed to enhance farming efficiency and crop management using machine learning. The FarmTech application designed to help farmers make informed decisions using machine learning. The system integrates advanced ML models to analyze real-time data, predict agricultural outcomes, and provide actionable recommendations. The system collects sensor data and stores it in a cloud database (Firebase/MySQL). APIs facilitate real-time data transfer over Wi-Fi using HTTP or MQTT. The central module aggregates sensor data and calculates optimized watering schedules. If internet connectivity is available, cloud-based ML models analyze data for precise recommendations. In offline mode, the system uses pre-defined logic for automated irrigation. Alerts (via SMS, email, or app notifications) notify farmers of potential risks. Built with a React frontend, a Spring Boot backend, and an H2 database, the platform leverages Python- based ML models for predictive analytics and decision-making. The system integrates powerful ML models such as Random Forest, ResNet152V2, MobileNet, and Bagging Classifier, utilizing TensorFlow and PyTorch to deliver accurate insights. The app consists of seven key modules, each addressing a critical aspect of agricultural management:

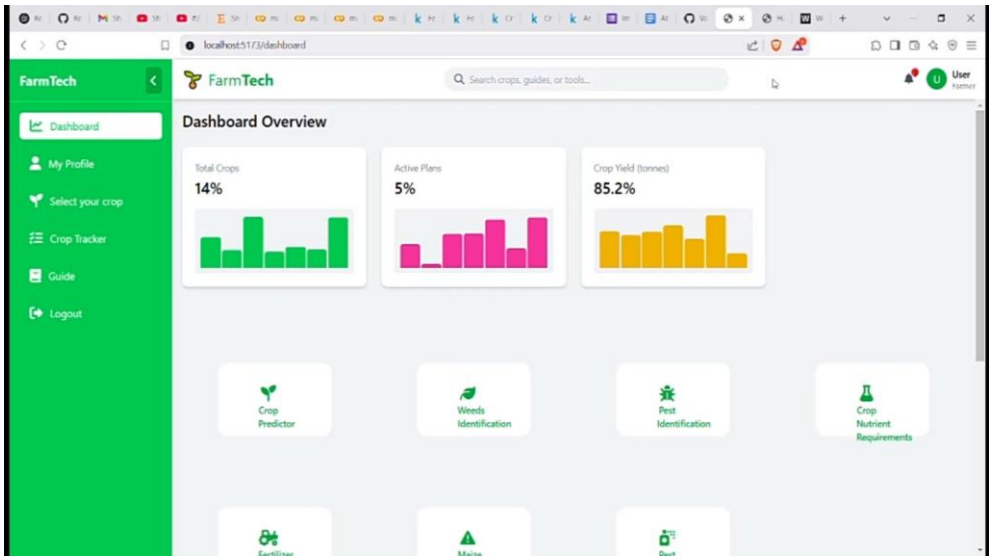
1) Crop Predictor,
2) Weeds Identification,
3) Pest Identification,
4) Crop Nutrient Requirements,
5) Fertilizer Recommendation
6) Maize disease prediction,
7) Pest Control Recommendation.

Dashboard:

This smart farming application empowers farmers with data driven insights, improving agricultural productivity, reducing losses, and optimizing resource usage through ai-powered recommendation.



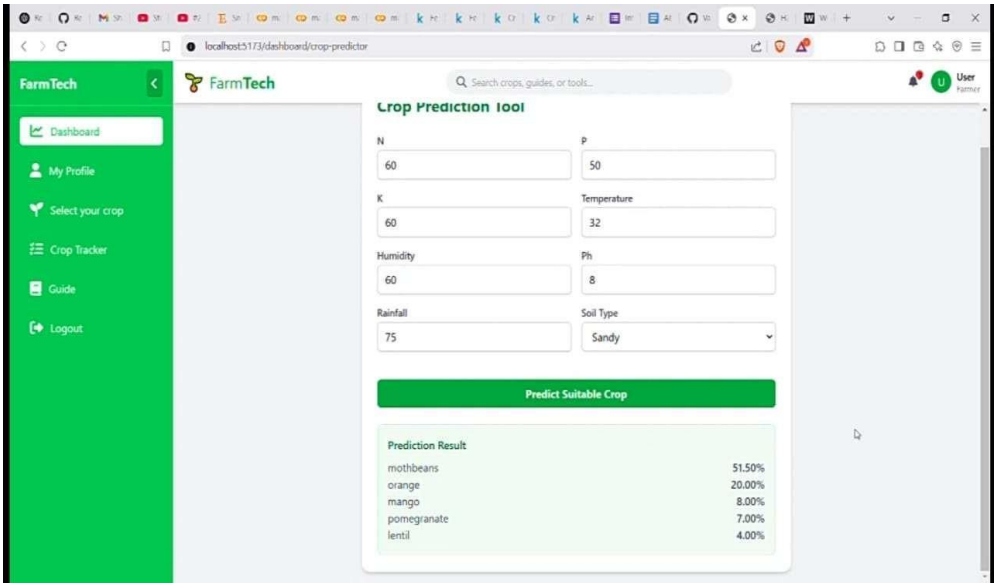
Screenshot 9.1 Dashboard



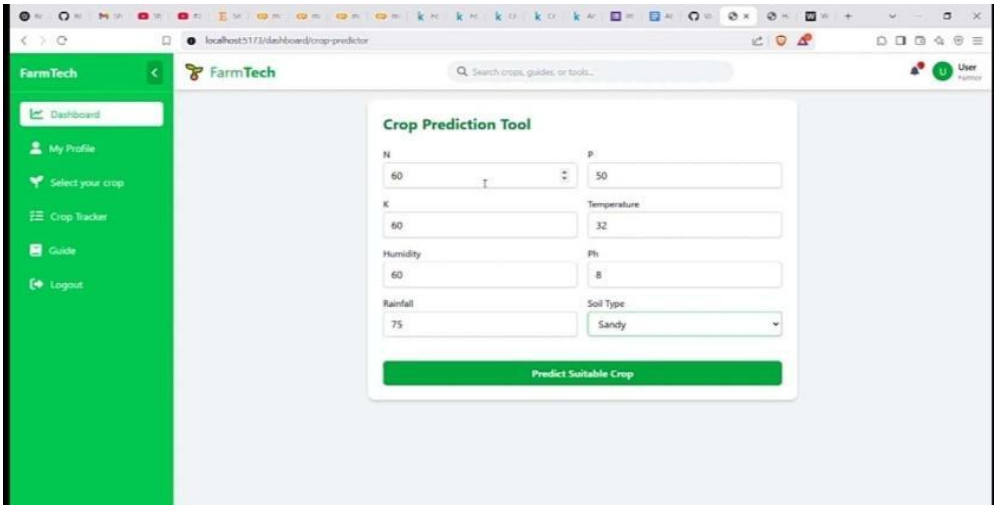
Screenshot 9.2 Dashboard

1) Crop Predictor:

Predicts suitable crops based on environmental factors. Predicts the best crops to grow based on environmental conditions.



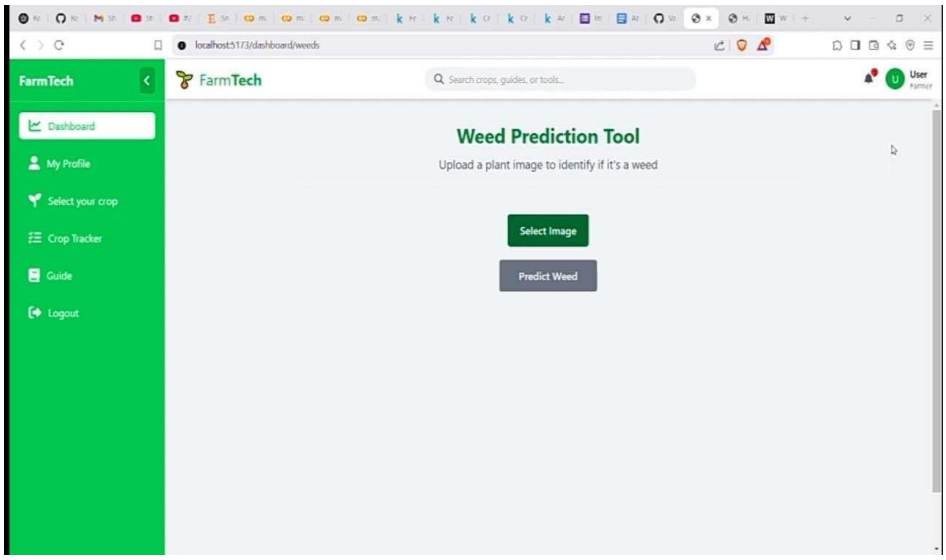
Screenshot 9.3 Crop Predictor



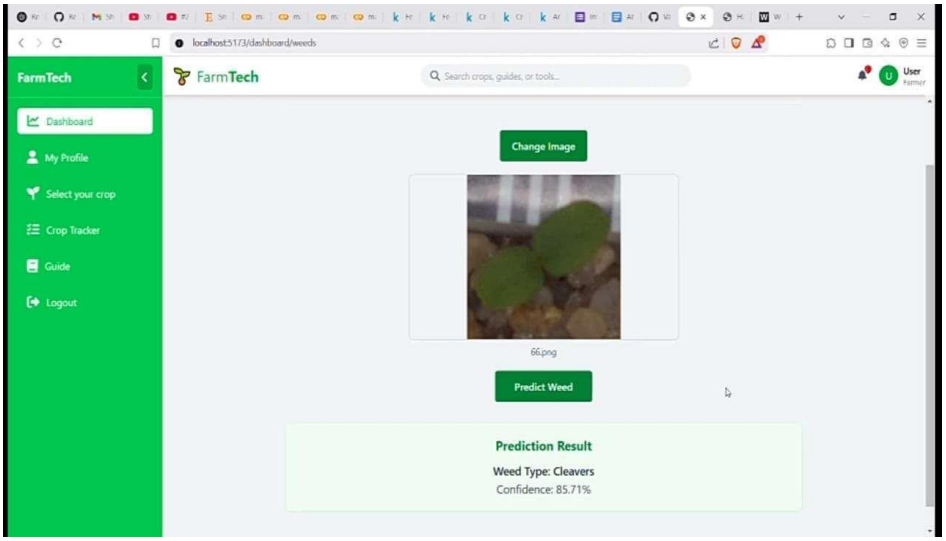
Screenshot 9.4 Crop Predictor

2) Weeds Identification:

Detects and classifies weeds for better weed management.



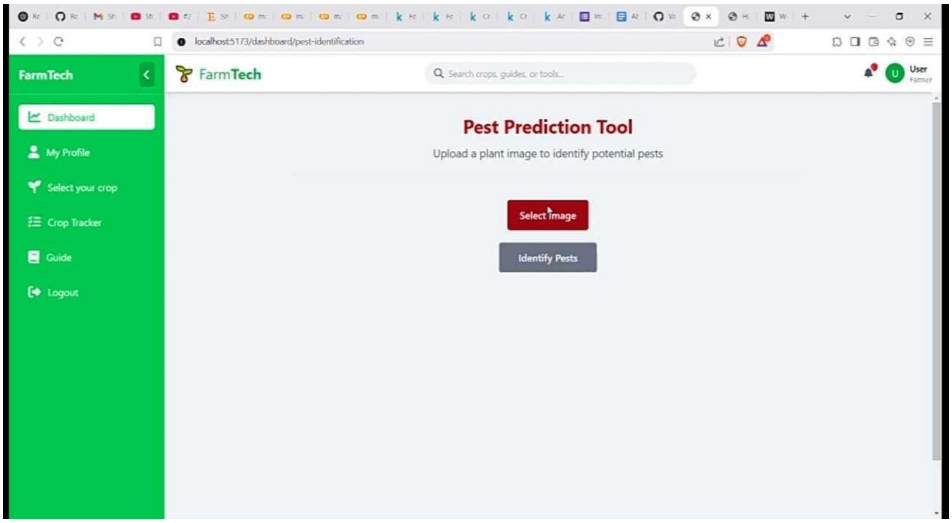
Screenshot 9.5 Weeds Identification



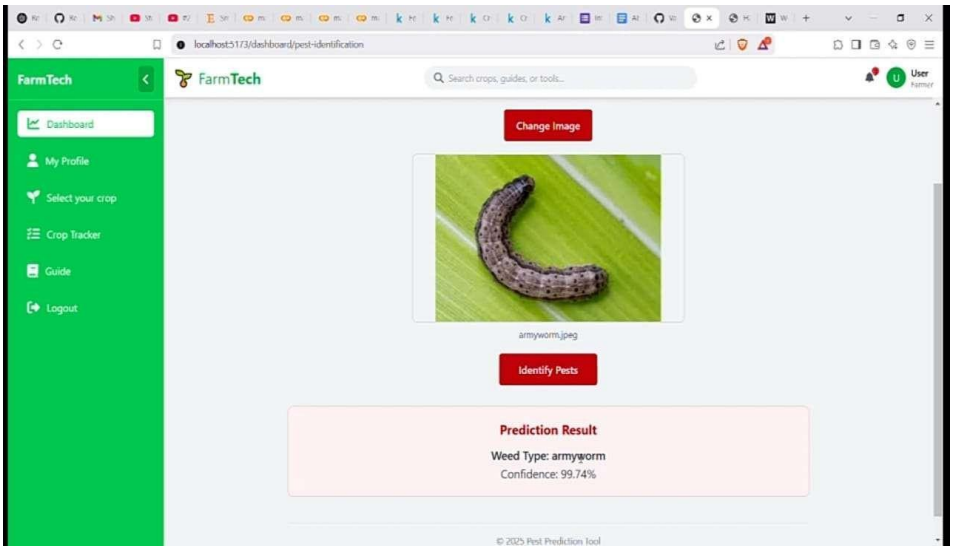
Screenshot 9.6 Weeds Identification

3) Pest Identification:

Identifies pests affecting crops using deep learning models. Identifies pests using deep learning for early intervention.



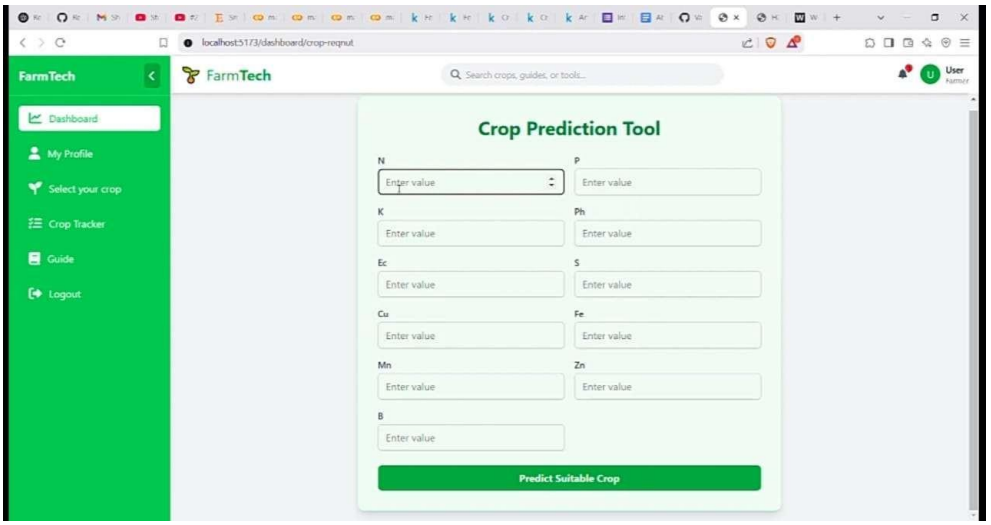
Screenshot 9.7 Pest Identification



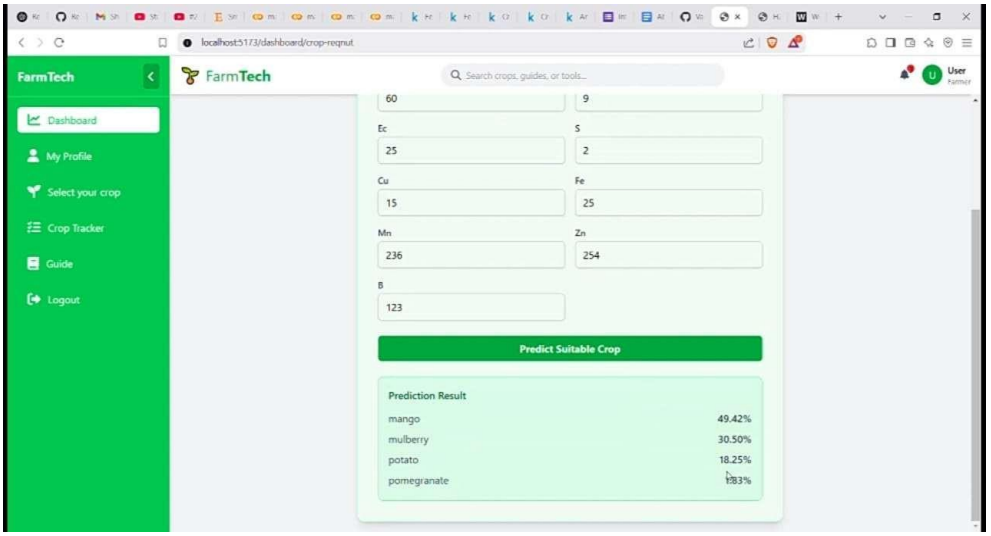
Screenshot 9.8 Pest Identification

4) Crop Nutrient Requirement:

Recommends necessary nutrients for optimal crop growth.



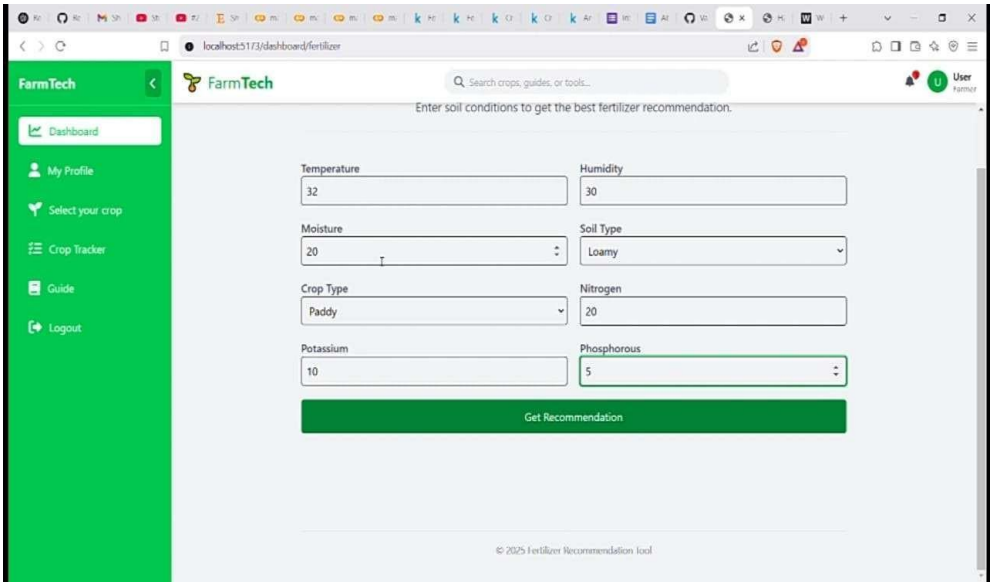
Screenshot 9.9 Crop Nutrient Requirements



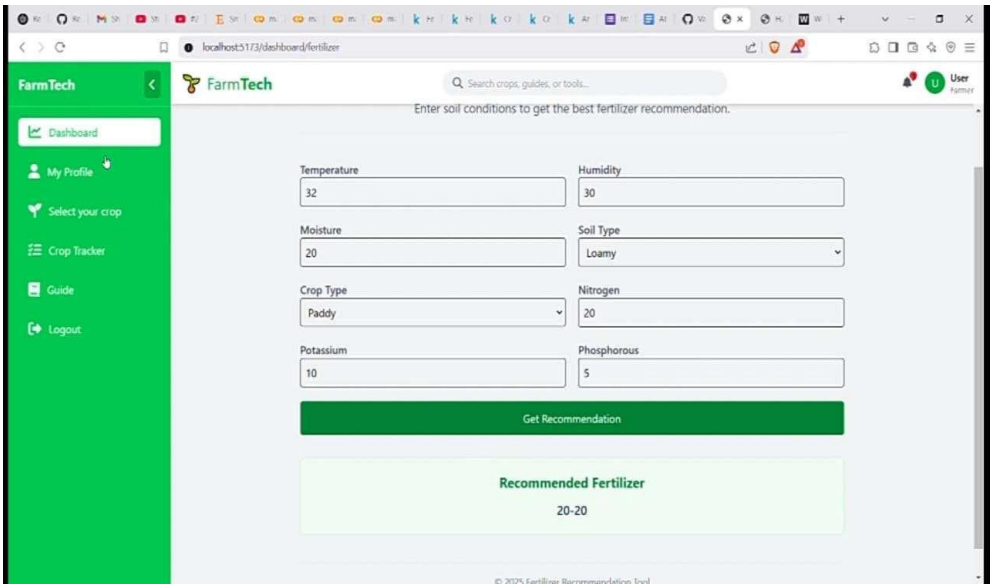
Screenshot 9.10 Crop Nutrient Requirements

5) Fertilizer Recommendation:

Suggests the best fertilizers based on soil and crop conditions.



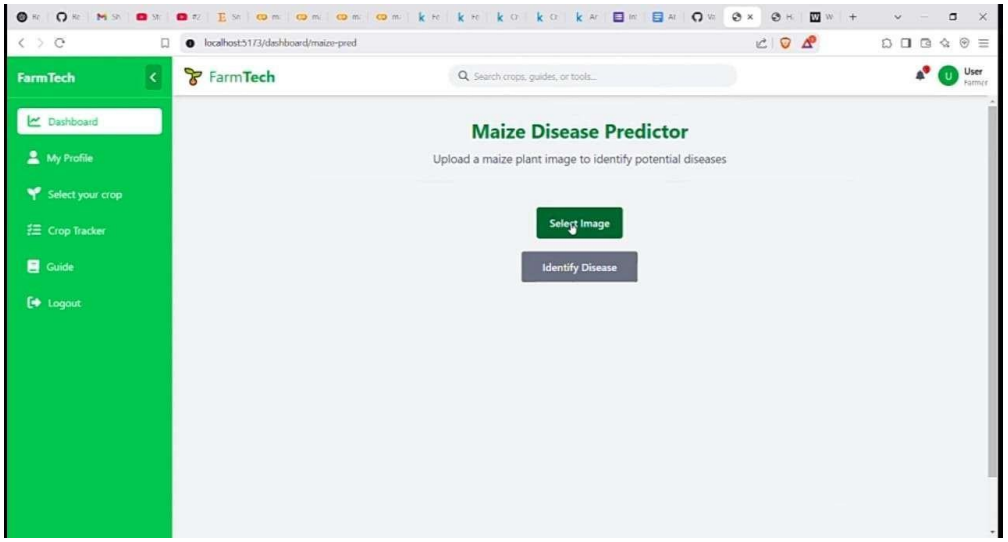
Screenshot 9.11 Fertilizer Recommendation



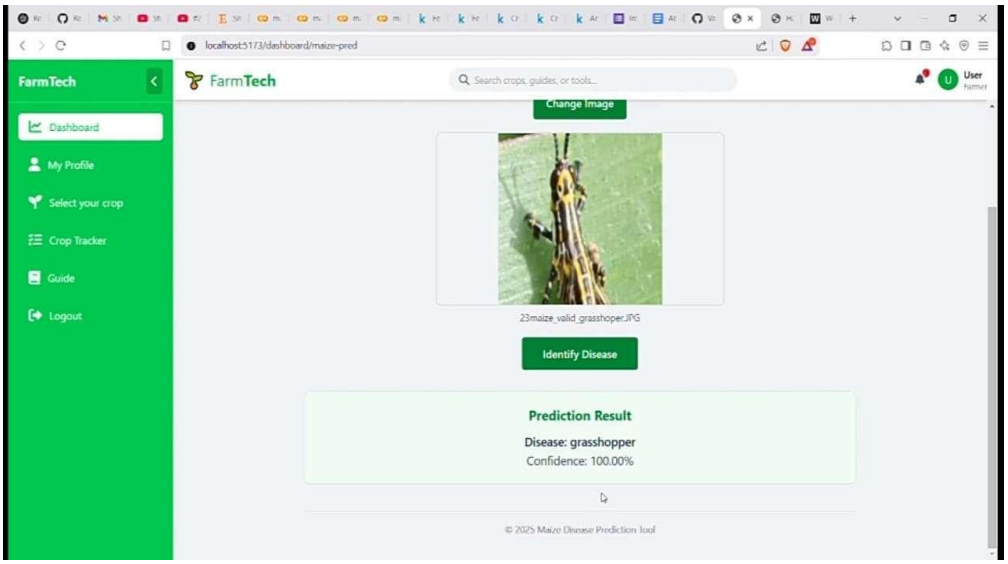
Screenshot 9.12 Fertilizer Recommendation

6) Maize Disease Prediction:

Detects and classifies maize plant diseases. Detects and predicts maize plant diseases with high accuracy.



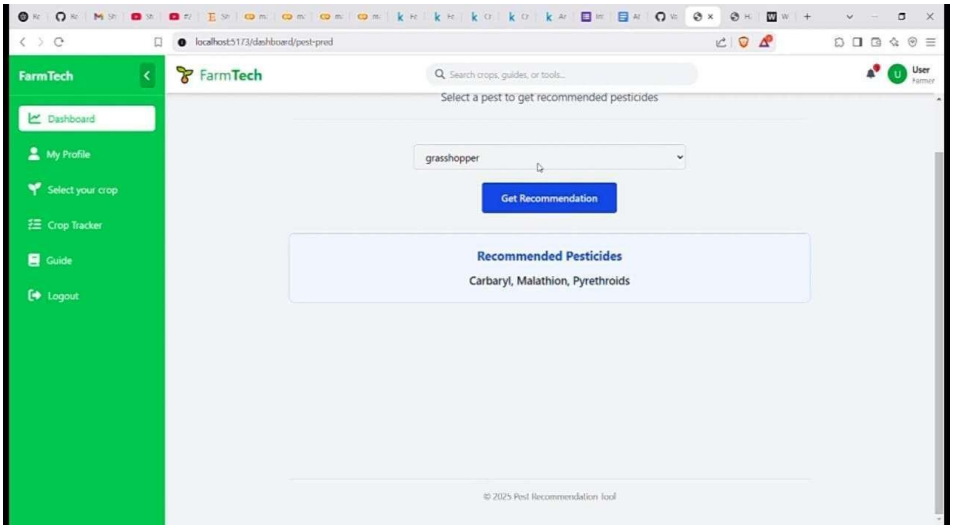
Screenshot 9.13 Maize disease prediction



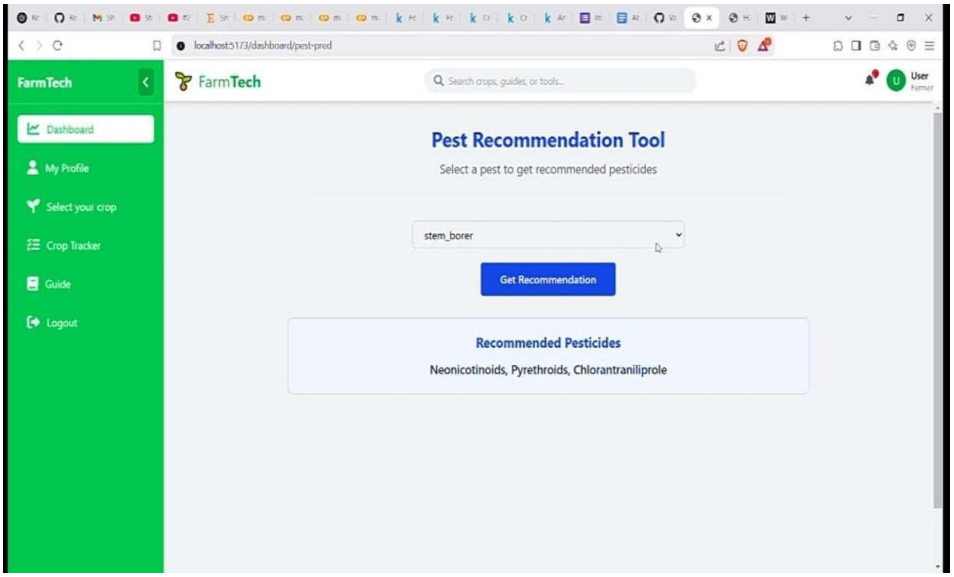
Screenshot 9.14 Maize disease prediction

7) Pest Control Recommendation:

Provides AI-based pest control solutions for better yield protection.



Screenshot 9.15 Pest Control Recommendation



Screenshot 9.16 Pest Control Recommendation

10.CONCLUSION AND FUTURE SCOPE

10.1 CONCLUSION

The Precision Agriculture System using Machine Learning presents a transformative approach to modern farming by integrating IoT sensors, machine learning, and automation to enhance productivity, optimize resource utilization, and promote sustainable agricultural practices. By collecting real-time environmental and crop data, the system enables data-driven decision-making, providing farmers with actionable insights through predictive analytics and smart recommendations. The automation of key farming operations, such as irrigation, fertilization, and pest control, reduces manual labor, minimizes resource wastage, and increases efficiency. Additionally, the integration of a user-friendly dashboard and alert system ensure that farmers receive timely updates and can take necessary actions to protect their crops and maximize yields.

Furthermore, the system enhances farm monitoring and market integration, allowing farmers to plan their production based on demand forecasts, reducing post-harvest losses, and improving profitability. With built-insecurity measures and scalability, the system can be adapted for different farm sizes and agricultural needs. Ultimately, this project demonstrates how machine learning and IoT can revolutionize farming, making it more efficient, cost-effective, and environmentally friendly. The successful implementation of this system can contribute to global food security and support sustainable agriculture for future generations.

10.2 FUTURESCOPE

Google Earth Engine (GEE) and UAV-based remote sensing play a crucial role in modern precision agriculture by enabling efficient monitoring and analysis of soil and vegetation health. These technologies help in soil pH and nutrient detection, allowing farmers to optimize fertilizer usage by identifying nutrient deficiencies through advanced spectral analysis. Vegetation health monitoring is enhanced using multispectral imaging, which detects plant stress, pest infestations, and disease outbreaks, ensuring timely interventions. Additionally, drought and irrigation planning benefit from real-time satellite data, helping farmers make informed decisions to conserve water and improve crop resilience.

Furthermore, yield prediction and climate impact analysis leverage AI-powered geospatial analytics to assess crop productivity, forecast potential losses, and understand climate variations affecting agriculture. By integrating GEE and UAV-based sensing, farmers can achieve higher efficiency, reduce resource wastage, and enhance overall agricultural sustainability.

11.REFERENCES

[1]. Dieisson Pivoto, Paulo Dabdab Waquil, Edson Talamini, Caroline Pauletto Spanhol Finocchio, Vitor Francisco Dalla Corte, Giana de Vargas Mores (2017). Scientific development of smart farming technologies and their application in Brazil. doi: 10.1016/j.inpa.2017.12.002

[2]. Shruthi, Soudha N, Khalid Akram, Mustafa Basthikodi, Ahmed RimazFaizabadi (2022). IoT based automation using Drones for Agriculture. doi: 10.1109/ICCST55948.2022.10040457

[3]. [2] Lincoln Alexandre Paz Silva , Francisco de Assis Brito Filho , Member, IEEE, and Humberto Dionísio de Andrade , Member, IEEE (2023). Soil Moisture Monitoring System Based on Metamaterial-Inspired Microwave Sensor for Precision Agriculture Applications. doi: 10.1109/JSEN.2023.3307652

[4]. YOUSSEF N. ALTHERWY, ALI ROMAN, SYED RAMEEZ NAQVI ANAS ALSUHAIBANI, AND TALLHA AKRAM (2024). Remote Sensing Insights: Leveraging Advanced Machine Learning Models and Optimization for Enhanced Accuracy in Precision Agriculture. doi: 10.1109/ACCESS.2024.3455169

[5]. KSHETRIMAYUM LOCHAN, ASIM KHAN, ISLAM ELSAYED, BHIVRAJ SUTHAR, LAKMAL SENEVIRATNE, AND IRFAN HUSSAIN, (Member, IEEE) (2024). Remote Sensing and Decision Support System Applications in Precision Agriculture: Challenges and Possibilities. doi: 10.1109/ACCESS.2024.3380830

[6]. Ramesh Reddy Donapati , Member, IEEE, Ramalingaswamy Cheruku , Member, IEEE, and Prakash Kodali , Member, IEEE (2024). Advancements in Precision Spraying of Agricultural Robots: A Comprehensive Review. doi: 10.1109/ACCESS.2024.3450904

[7]. Donapati, R. R., Cheruku, R., & Kodali, P. (2023). Real-Time Seed Detection and Germination Analysis in Precision Agriculture: A Fusion Model With U-Net and CNN on Jetson Nano. doi: 10.1109/TAFE.2023.3332495

[8]. Sachin Chandravadan Karad (2023). Smart NPK Soil Sensor: Step towards Precision Agriculture.

[9]. Devdatta A. Bondre Student, NICT Solutions & Research, Belagavi, Karnataka, India Mr. Santosh Mahagaonkar Research Head, NICT Solutions & Research, Belagavi, Karnataka, India (2019). Prediction of Crop Yield and Fertilizer Recommendation Using Machine Learning Algorithms.