Project Stage-II

on

"FIELDS OF RESILIENCE: IOT BASED SOLUTION FOR CROP WELLNESS"

Submitted

in partial fulfilment of the requirements for

the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

(Artificial Intelligence and Machine Learning)

by

MARAM ROHITH REDDY (20261A6635)

NEHA DEEPTHI KANDI (20261A6638)

Under the guidance of

Dr. B. Yadaiah (Assistant Professor)

Mr. M. Srikanth Sagar (Assistant Professor)

DEPARTMENT OF EMERGING TECHNOLOGIES

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

Autonomous | (Affiliated to Jawaharlal Nehru Technological University Hyderabad)
Gandipet, Hyderabad- 500075, Telangana (India)

2023 – 2024

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

Autonomous | (Affiliated to Jawaharlal Nehru Technological University Hyderabad)
Gandipet, Hyderabad-500075, Telangana

CERTIFICATE

This is to certify that the Project Stage-1 entitled "**FIELDS OF RESILIENCE: IOT BASED SOLUTION FOR CROP WELLNESS**", being submitted by **MARAM ROHITH REDDY, NEHA DEEPTHI KANDI** bearing **Roll No: 20261A6635, 20261A6638** in partial fulfilment of the requirements for the Award of the **Degree of Bachelor of Technology** in **Computer Science and Engineering (Artificial Intelligence and Machine Learning)** to **Jawaharlal Nehru Technological University Hyderabad** is a record of bonafide work carried out under the supervision.

The results in this have not been submitted to any other University or Institute for the award of any degree or diploma.

Project Guide | **Head of the Department**

Dr. B. Yadaiah
Assistant Professor

Dr. M. Rama Bai
Professor & HoD

Mr. M. Srikanth Sagar
Assistant Professor

External Examiner

DECLARATION

We hereby declare that the work reported in the project titled **"FIELDS OF RESILIENCE: IOT BASED SOLUTION FOR CROP WELLNESS"** is original and bonafide work carried out by us as a part of partial fulfilment for Bachelor of Technology in Computer Science and Engineering (Artificial Intelligence and Machine Learning), Mahatma Gandhi Institute of Technology, Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by us and not copied from any other source.

MARAM ROHITH REDDY	**NEHA DEEPTHI KANDI**
(20261A6635)	**(20261A6638)**

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the project.

We would like to express our sincere thanks to **Dr. G. ChandraMohan Reddy, Principal MGIT**, for providing the work facilities in the college.

We are also thankful to **Dr. M. Rama Bai, Professor and HoD**, **Dept. of ET**, MGIT for providing excellent infrastructure and a conducive atmosphere for completing the project successfully.

We are also extremely thankful to our Project Coordinators, **Dr. D. Koteswara Rao, Assistant Professor, Dept. of ET** and **Mr. R. Srinivas, Assistant Professor, Dept. of ET,** for their valuable suggestions and interest throughout the course of the project.

We would like to express our sincere gratitude and indebtedness to our Project Guide **Dr. B. Yadaiah, Assistant Professor, Dept. of ET** and **Mr. M. Srikanth Sagar, Assistant Professor, Dept. of ET** who has supported us throughout the project with patience and knowledge.

We convey our heartfelt thanks to the lab staff for allowing us to use the required equipment whenever needed.

Finally, we would like to take this opportunity to thank our family for their support throughout the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of the work.

MARAM ROHITH REDDY
(20261A6635)

NEHA DEEPTHI KANDI
(20261A6638)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Farming is one of the major sectors that influences a country's economic growth. In country like India, majority of the population is dependent on agriculture for their livelihood. The project involves building a crop recommendation system extended with disease prediction using machine learning algorithms and integrate it into a web application using the Flask framework.

The system integrates two crucial features: Crop Recommendation and Disease Prediction, leveraging machine learning algorithms. The Crop Recommendation module facilitates IOT data-driven decision-making by allowing farmers to input soil and environmental parameters such as soil type, rainfall, temperature, and humidity. Temperature and humidity are collected through sensor (are directly fed from soil analysis report) and rainfall values, some types of sensors would be deployed and recommends the best crop that can be grown in that particular region based on the machine learning model trained on historical data.

For Disease Prediction, a Resnet model classifies images of diseased plant leaves uploaded by farmers. The system then predicts the type of disease affecting the crop, along with its symptoms, preventive measures, and causes.

The system uses machine learning algorithms such as decision trees, random forests, and support vector machines to make predictions and recommend the best crop for the given input data. The web application created using Flask allows farmers to easily input their data and it represents a significant step towards modernizing agriculture, offering farmers personalized recommendations and early disease detection capabilities.

It is useful for farmers who want to maximize their crop yield and minimize crop failure by selecting the best crop based on the conditions of their land. The system uses machine learning algorithms such as decision trees, random forests, and support vector machines to make predictions and recommend the best crop for the given input data.

The project showcases the power of machine learning and how it can be used to solve real-world problems in the agricultural industry. The web application created using Flask allows farmers to easily input their data and it represents a significant step towards modernizing agriculture, offering farmers personalized recommendations and early disease detection capabilities.

1. INTRODUCTION

In India, the agriculture industry is extremely vital and crucial for economic and social development and jobs. In India, the agricultural sector provides a living for almost 48% of the population. Most of the Indian population depends on agriculture for their livelihood. Agriculture gives an opportunity of employment to the village people to develop a country like India on large scale and give a push in the economic sector. The majority of farmers face the problem of planting an inappropriate crop for their land based on a conventional or non-scientific approach. This is a challenging task for a country like India, where agriculture feeds approximately 42% of the population. And the outcomes for the farmer of choosing the wrong crop for land is moving towards metro city for livelihoods, suicide, quitting the agriculture and give land on lease to industrialist or use for the non-agriculture purpose.

The outcome of wrong crop selection is less yield and less profit. Machine learning, which is subpart of Artificial Intelligence, is being used to implement the proposed system. Crop recommendation is going to recommend you the best crop you can grow in your land as per the soil nutrition value and along with as per the climate in that region. Also, identification of the plant diseases is important in order to prevent the losses within the yield. It's terribly troublesome to observe the plant diseases manually. It needs tremendous quantity of labor, expertise within the plant diseases, and conjointly need the excessive time interval.

Hence, image processing and deep learning models can be employed for the detection of plant diseases. In the project, we have described the technique for the detection of plant diseases with the help of their leaves pictures. Image processing is a branch of signal processing which can extract the image properties or useful information from the image. The color of leaves, amount of damage to leaves, area of the leaf, texture parameters are used for classification. The research's main objective is to bring farming process a step closer to the digital platform.

1.1. EXISTING SYSTEM

Traditionally, farmers have relied on local knowledge and experience, often resulting in suboptimal crop choices. While there are some existing agricultural advisory services and apps, they often lack the sophistication of machine learning algorithms. These services typically provide general recommendations based on historical data but may not adapt to the specific conditions of individual farms. Thus, there is a clear need for a more advanced solution that leverages predictive modeling to offer precise and personalized crop recommendations. Previously plant disease detection is done by visual inspection of the leaves or some chemical processes by experts. For doing so, a large team of experts as well as continuous observation of plant is needed, which costs high when we do with large farms. In such conditions, the recommended system proves to be helpful in monitoring large fields of crops.

1.2. PROBLEM STATEMENT

The need for a robust crop recommendation system that harnesses the potential of machine learning algorithms to assist farmers in choosing the most suitable crops for cultivation in specific geographical regions. The recommendation system must take into account various environmental factors such as soil type, rainfall, temperature, and humidity to provide personalized crop recommendations. The primary challenge is to develop an accurate and user-friendly solution that optimizes crop yield, thereby reducing crop failure and maximizing agricultural productivity. In addition to the crop recommendation challenges, the agriculture sector grapples with plant diseases that wreak havoc on farms. Current disease detection methods often lack efficiency, resulting in substantial losses for farmers. Integrating advanced machine learning algorithms into the recommendation system can address this issue by enabling early detection of plant diseases based on visual cues and environmental data. This dual-functionality not only aids in optimal crop selection but also serves as a preventive measure, minimizing the impact of diseases and bolstering agricultural sustainability.

1.3. PROPOSED SOLUTION

Our proposed solution integrates machine learning algorithms and a user-friendly web application developed using the Flask framework. The system takes input data, including soil type, rainfall, temperature, and humidity, and employs machine learning models trained on historical data to recommend the most suitable crop for a given region. The system uses machine learning algorithms such as decision trees, random forests, and support vector machines to make predictions and recommend the best crop for the given input data.

Automatic detection of the diseases by simply seeing the symptoms on the plant leaves makes it easier as well as cheaper. The proposed solution for plant disease detection is computationally less expensive and requires less time for prediction than other deep learning based approaches since it utilizes ResNet, ensuring accurate predictions while balancing computational efficiency for real-world applicability.

1.4. Requirement Specification

These specifications provide a clear directive on the software and hardware environment necessary for the project. These requirements serve as guidelines for developers and users to ensure the proper functioning of the software.

1.4.1 Hardware Requirements:

IoT Devices	:	NodeMCU, DHT11 sensor, Soil Moisture Sensor
Processor	:	Intel Core-i5
RAM	:	4 GB
ROM	:	1 TB

1.4.2 Software Requirements:

Operating System	: Windows family
Programing language	: Python, C++, HTML, Javascript, PHP
IDE	: Arduino IDE, Visual Studio Code
Web Framework	: Flask

2. LITERATURE SURVEY

“RANet: Lightweight Crop Disease Identification Model based on Reparameterization and Attention Mechanism” [1] is inspired by MobileNetV2, RANet introduces a re-parameterizable basic module called RABlock. The structure of RABlock differs between training and inference, reducing model parameters and computational requirements while maintaining recognition accuracy and improving inference speed. Additionally, RANet incorporates the ECANet attention mechanism to enhance the model’s focus on disease regions, further improving recognition performance. On a custom-built leaf disease dataset, RANet achieves an identification accuracy of 97.98% with only 1.8MB of parameters and a computational complexity of 0.179G. The inference time for a single-leaf disease image is 2.5ms. RANet outperforms lightweight models like MobileNetV3, striking a balance between recognition accuracy and speed. These results provide valuable insights for deploying lightweight deep learning models on edge devices.

In **“Diseases Identification Using ConvNet In Sugarcane Crops”[2]** the paper presents the automated recognition of sugarcane diseases using a deep learning method ConvNet (CNNs). The proposed model leverages the power of deep learning to analyze images of sugarcane leaves and classify them into different disease categories. The results demonstrate the efficiency of the deep learning model in accurately identifying sugarcane diseases, paving the way for improved disease management practices.

In **“Healthy Harvest: Crop Prediction and Disease Detection System”[3]** it suggests a system that would assist farmers in combining their current farming practices with cutting-edge technologies like deep learning, machine learning, and remote sensing built using an open technology stack of Angular for the frontend, Flask for the backend, and MySQL and Google Earth for the database.

“FarmCare: Location-based Profitable Crop Recommendation System”[4] paper has identified four main problems. First, it was wrong crop cultivation. This is the main reason crops and cultivation are destroyed. To give a solution to that problem, we suggest the five most suitable crops to cultivate according to their location. The second problem is a lack of knowledge about future market prices. As a solution to that problem, we predict prices for each cop for the next 12 months. Another problem is an inability to sell their product at a reasonable price. Here, they directly connect buyers and sellers by removing intermediaries. The last problem is the difficulty to identify diseases affected by crops. Using our mobile app farmers can identify which disease affected their crops by uploading an image to the app. To give solutions to the above-mentioned problems Machine Learning algorithms

In **“Plant Disease Detection Using Convolution Neural Network”[5],** the global rise in population has led to a shortage of raw materials and food supplies. The agricultural sector has become the primary and most vital source to overcome this particular constraint. However, the industry itself is facing the challenge of pests and various crop diseases. Battling this has been the significant focus of the sector for decades. Still, due to the technology gap that existed earlier, there existed a constraint on identifying the diseased crops on a massive scale. Nevertheless, today, with the improvement of technologies such as drones, IoT devices, and higher processing speeds combined with data analysis and machine learning, the problem of identification can be resolved quickly. This paper aims to provide a brief description of existing solutions that have been published and focuses on the more efficient machine learning model based on conventional neural networks (CNN) that we have developed. This machine learning model can be deployed on IoT devices, mobile phones, and drones and cameras that farmers can utilize to identify the diseased crops on a massive scale and take the necessary precautions not to let the disease spread and affect the supply produced. The proposed model using CNN was trained using images from plant village dataset and attained an accuracy of 94.87% in identifying the diseased plants with the help of image processing by OpenCV. Finally, the paper showcases the detailed analysis of the proposed scheme and results obtained by the model.

In “**Efficient Crop Yield Prediction A Backing Vector Recommendation System Using Supervised Machine Learning for Digital Farming**”[6], research proposes an efficient crop yield prediction system leveraging a Backing Vector Recommendation System empowered by the Backing Vector Machine (BVM), a variant of the Supervised Machine Learning algorithm Support Vector Machine (SVM).Digital farming is at the forefront of agricultural innovation, employing advanced technologies to revolutionize traditional farming practices. Our methodology involves collecting diverse datasets, encompassing variables such as weather patterns, soil quality, and historical crop yields. Through rigorous preprocessing, including data cleaning and feature engineering, prepare the datasets for training the BVM. The BVM's robust classification capabilities are harnessed to create a reliable recommendation system for predicting crop yields in various agricultural scenarios. The research contributes to the ongoing evolution of digital farming by introducing a powerful tool for precise crop yield prediction.

As future work, we recommend exploring additional machine learning techniques, integrating real-time data for enhanced accuracy, and adapting the system for scalability with large agricultural datasets. This research sets the stage for advancements in agricultural technology, paving the way for more informed and sustainable farming practices.

In "**Plant Leaf Detection and Disease Recognition using Deep Learning"[7],** the latest improvements in computer vision formulated through deep learning have paved the method for how to detect and diagnose diseases in plants by using a camera to capture images as basis for recognizing several types of plant diseases. This study provides an efficient solution for detecting multiple diseases in several plant varieties. The system was designed to detect and recognize several plant varieties specifically apple, corn, grapes, potato, sugarcane, and tomato. The system can also detect several diseases of plants. Comprised of 35,000 images of healthy plant leaves and infected with the diseases, the researchers were able to train deep learning models to detect and recognize plant diseases and the absence these of diseases. The trained model has achieved an accuracy rate of 96.5% and the system was able to register up to 100% accuracy in detecting and recognizing the plant variety and the type of diseases the plant was infected.

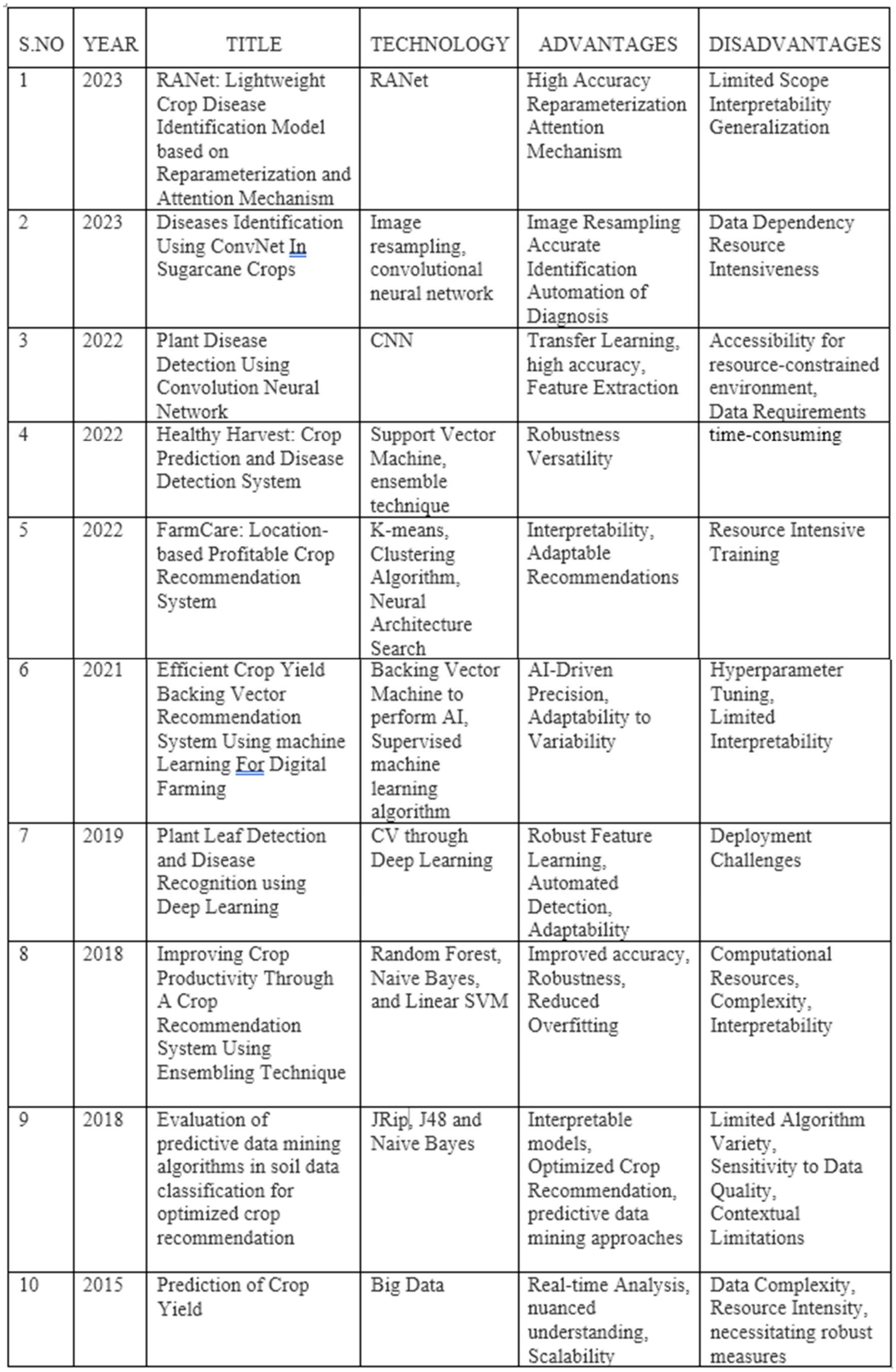
In **"Improving Crop Productivity Through a Crop Recommendation System Using Ensembling Technique"[8],** Agriculture plays a predominant role in the economic growth and development of the country. The major and serious setback in the crop productivity is that the farmers do not choose the right crop for cultivation. In order to improve the crop productivity, a crop recommendation system is to be developed that uses the ensembling technique of machine learning. The ensembling technique is used to build a model that combines the predictions of multiple machine learning models together to recommend the right crop based on the soil specific type and characteristics with high accuracy. The independent base learners used in the ensemble model are Random Forest, Naive Bayes, and Linear SVM. Each classifier provides its own set of class labels with an acceptable accuracy. The class labels of individual base learners are combined using the majority voting technique. The crop recommendation system classifies the input soil dataset into the recommendable crop type, Kharif and Rabi. The dataset comprises of the soil specific physical and chemical characteristics in addition to the climatic conditions such as average rainfall and the surface temperature samples. The average classification accuracy obtained by combining the independent base learners is 99.91%.

In **“Evaluation of predictive data mining algorithms in soil data classification for optimized crop recommendation”[9],** Agricultural research has strengthened the optimized economical profit, internationally and is very vast and important field to gain more benefits. However, it can be enhanced by the use of different technological resources, tool, and procedures. Today, the term data mining is an interdisciplinary process of analyzing, processing and evaluating the real-world datasets and prediction on the basis of the findings. Our case-based analysis provides empirical evidence that we can use different data mining classification algorithms to classify the dataset of agricultural regions on the basis of soil properties. Additionally, we have investigated the most performing algorithm having powerful prediction accuracy to recommend the best crop for better yield.

In **“Prediction of Crop Yield Using Big Data”[10],** Quantifying the yield is essential to optimize policies to ensure food security. This paper aims at providing a new method to predict the crop yield based on big-data analysis technology, which differs with traditional methods in the structure of handling data and in the means of modeling. Firstly, the method can make full use of the existing massive agriculture relevant datasets and can be still utilized with the volume of data growing rapidly, due to big-data friendly processing structure. Secondly, the "nearest neighbors" modeling, which employs results gained from the former data processing structure, provides a well-balanced result on the account of accuracy and prediction time in advance. Numerical examples on actual crop dataset in China from 1995-2014 have showed a better performance and an improved prediction accuracy of the proposed method compared with traditional ones

2.1 Summary of Literature Survey

Table 2.1: Literature Survey



S.NO	YEAR	TITLE	TECHNOLOGY	ADVANTAGES	DISADVANTAGES
1	2023	RANet: Lightweight Crop Disease Identification Model based on Reparameterization and Attention Mechanism	RANet	High Accuracy Reparameterization Attention Mechanism	Limited Scope Interpretability Generalization
2	2023	Diseases Identification Using ConvNet In Sugarcane Crops	Image resampling, convolutional neural network	Image Resampling Accurate Identification Automation of Diagnosis	Data Dependency Resource Intensiveness
3	2022	Plant Disease Detection Using Convolution Neural Network	CNN	Transfer Learning, high accuracy, Feature Extraction	Accessibility for resource-constrained environment, Data Requirements
4	2022	Healthy Harvest: Crop Prediction and Disease Detection System	Support Vector Machine, ensemble technique	Robustness Versatility	time-consuming
5	2022	FarmCare: Location-based Profitable Crop Recommendation System	K-means, Clustering Algorithm, Neural Architecture Search	Interpretability, Adaptable Recommendations	Resource Intensive Training
6	2021	Efficient Crop Yield Backing Vector Recommendation System Using machine Learning For Digital Farming	Backing Vector Machine to perform AI, Supervised machine learning algorithm	AI-Driven Precision, Adaptability to Variability	Hyperparameter Tuning, Limited Interpretability
7	2019	Plant Leaf Detection and Disease Recognition using Deep Learning	CV through Deep Learning	Robust Feature Learning, Automated Detection, Adaptability	Deployment Challenges
8	2018	Improving Crop Productivity Through A Crop Recommendation System Using Ensembling Technique	Random Forest, Naive Bayes, and Linear SVM	Improved accuracy, Robustness, Reduced Overfitting	Computational Resources, Complexity, Interpretability
9	2018	Evaluation of predictive data mining algorithms in soil data classification for optimized crop recommendation	JRip, J48 and Naive Bayes	Interpretable models, Optimized Crop Recommendation, predictive data mining approaches	Limited Algorithm Variety, Sensitivity to Data Quality, Contextual Limitations
10	2015	Prediction of Crop Yield	Big Data	Real-time Analysis, nuanced understanding, Scalability	Data Complexity, Resource Intensity, necessitating robust measures

3. DESIGN AND METHODOLOGY

The section provides a detailed description of how the project will be executed, including the design, methodologies, techniques, tools, and resources employed throughout its lifecycle.

3.1 SYSTEM ARCHITECTURE

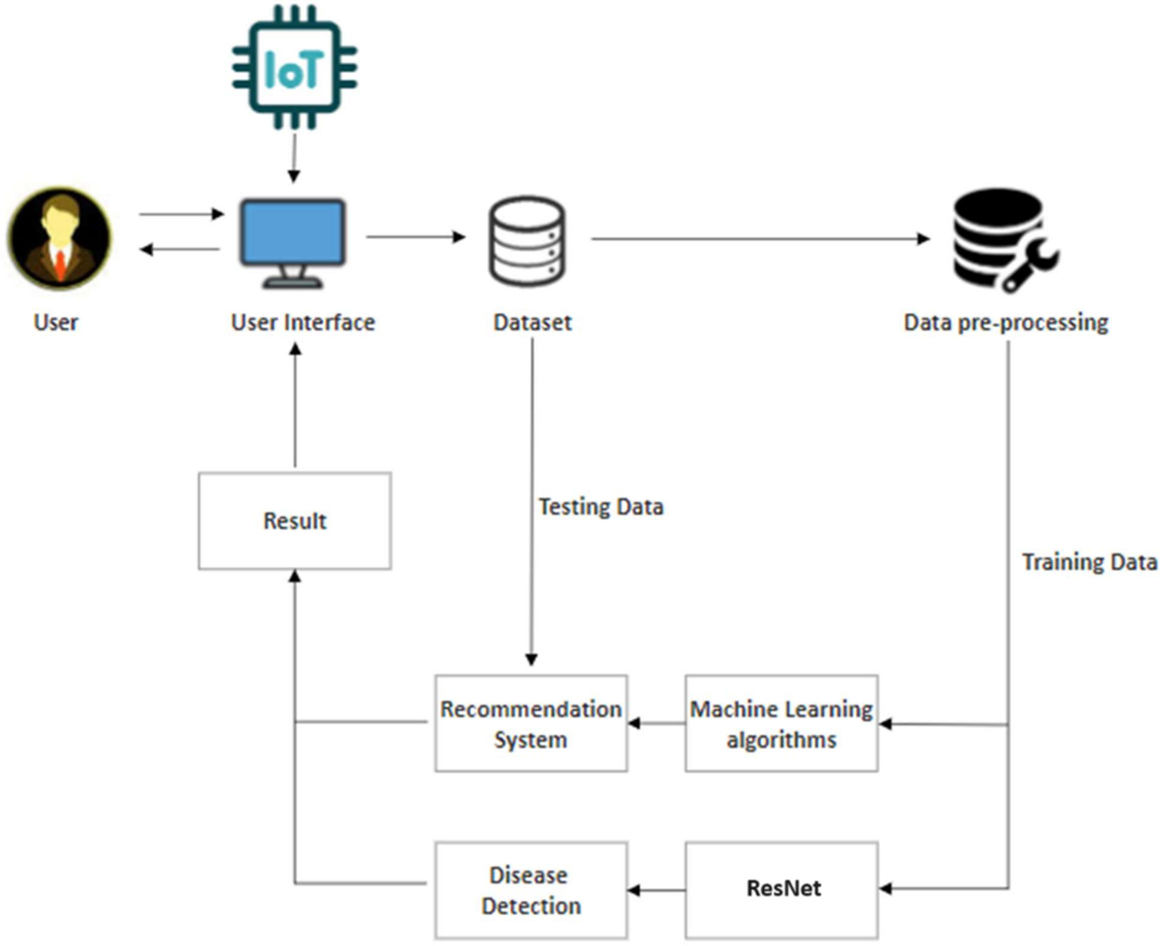


Figure 3.1: System Architecture

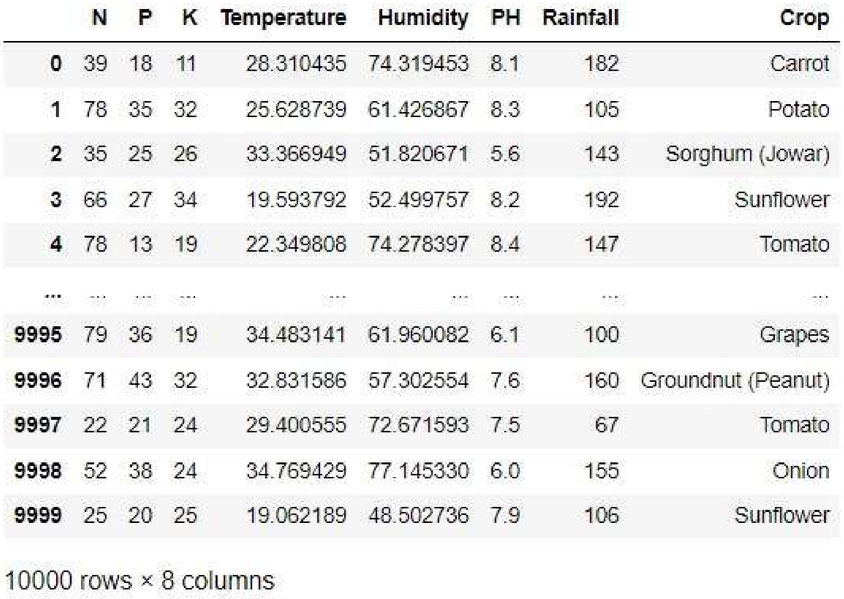
Figure 3.1 illustrates a machine learning system designed for disease detection, employing a user-friendly interface for data input. The process begins with the collection of a dataset, which undergoes pre-processing to prepare it for analysis. This data is then divided into testing and training sets. The training data is utilized to train various machine learning algorithms, creating a model capable of making predictions. These predictions are tested against the testing data to evaluate the model's accuracy. The system includes a recommendation feature that leverages the model's predictions to suggest potential actions to the user. The final component of the system is the disease detection function, which utilizes ResNet to enhance prediction accuracy. This architecture integrates user interaction, data processing, machine learning, and disease detection into a cohesive system aimed at providing actionable insights for disease management.

3.2 DATASET

The section describes the datasets used in the project, i.e., Crop recommendation system dataset and Disease detection dataset.

3.2.1 CROP RECOMMENDATION SYSTEM

The dataset was built by augmenting datasets of rainfall, climate and soil data available for India. This data is relatively simple with very few but useful features unlike the complicated features affecting the yield of the crop. The data has Nitrogen, Phosphorous, Pottasium and pH values of the soil. Also, it also contains the humidity, temperature and rainfall required for a particular crop (shown in figure 3.2).



	N	P	K	Temperature	Humidity	PH	Rainfall	Crop
0	39	18	11	28.310435	74.319453	8.1	182	Carrot
1	78	35	32	25.628739	61.426867	8.3	105	Potato
2	35	25	26	33.366949	51.820671	5.6	143	Sorghum (Jowar)
3	66	27	34	19.593792	52.499757	8.2	192	Sunflower
4	78	13	19	22.349808	74.278397	8.4	147	Tomato
...	...	...	...	...	...	...	...	...
9995	79	36	19	34.483141	61.960082	6.1	100	Grapes
9996	71	43	32	32.831586	57.302554	7.6	160	Groundnut (Peanut)
9997	22	21	24	29.400555	72.671593	7.5	67	Tomato
9998	52	38	24	34.769429	77.145330	6.0	155	Onion
9999	25	20	25	19.062189	48.502736	7.9	106	Sunflower

10000 rows × 8 columns

Figure 3.2: Dataset of crop recommendation

3.2.2 DISEASE DETECTION DATASET

The dataset is recreated using offline augmentation from the original dataset. This dataset consists of about 1,7K RGB images of healthy and diseased crop leaves which is categorized into 38 different classes. The total dataset is divided into 80/20 ratio of training and validation set preserving the directory structure. A new directory containing 33 test images is created later for prediction purpose.

3.3 UML DIAGRAMS

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system. UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modelling workflows as well as a wider range of features to improve readability and efficiency.

3.3.1 DATA FLOW DIAGRAM

The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
The Data flow diagram consists of two main parts:

- Frontend
- Sensors
- Backend

Frontend:

In the frontend, we have developed a user-friendly web interface using Flask (shown in figure 3.3). The interactive webpage allows users to input soil data for crop recommendation and image of defected leaf for disease detection easily. The clean and responsive design enhances the user experience, providing a seamless platform for interaction. The user inputs soil details and uploads an image of a leaf to the web application. The web application uses the soil details and the leaf image to predict the crop yield and the plant disease. The web application displays the output to the user.

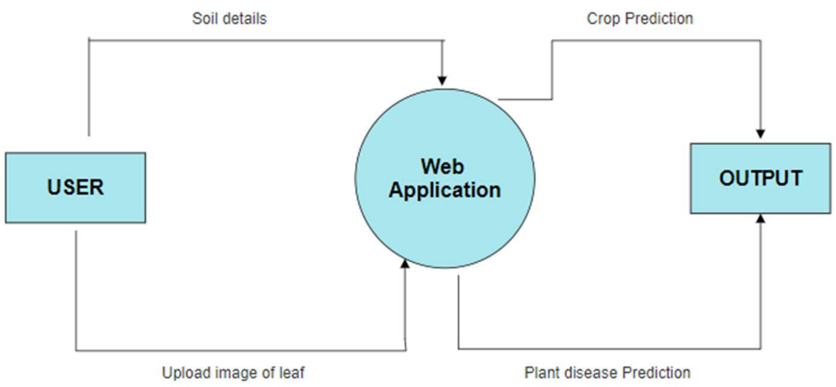


Figure 3.3: Data Flow diagram of Frontend

Sensors:

The data flow diagram of Sensors (Figure 3.4) showcases a smart agriculture system that uses sensors to collect environmental data. The DHT11 Sensor measures temperature and humidity, while the Soil Moisture Sensor assesses soil water content. This data is gathered by the NODE MCU ESP8266, a microcontroller that sends the information to a XAMPP Server for processing. The outcome is valuable weather and soil data used for optimizing farming practices. This setup represents a basic yet effective approach to precision agriculture, enabling real-time monitoring and informed decision-making for crop management.

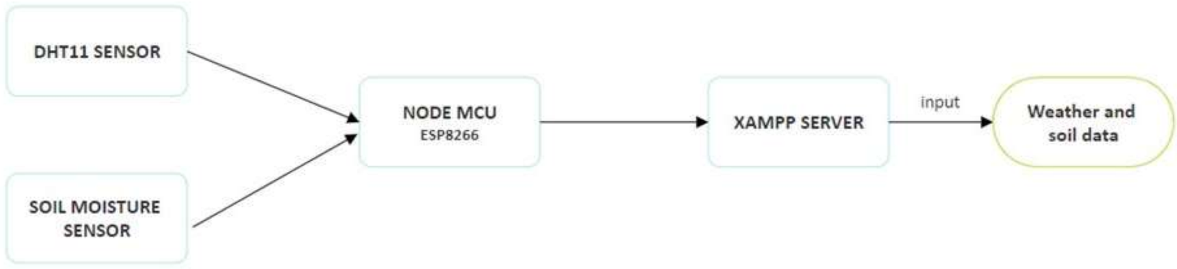


Figure 3.4: Data Flow diagram of Sensors

Backend:

The data flow diagram of Backend (Figure 3.5) is showing how the crop recommendation and disease detection models work in the backend of a web application.

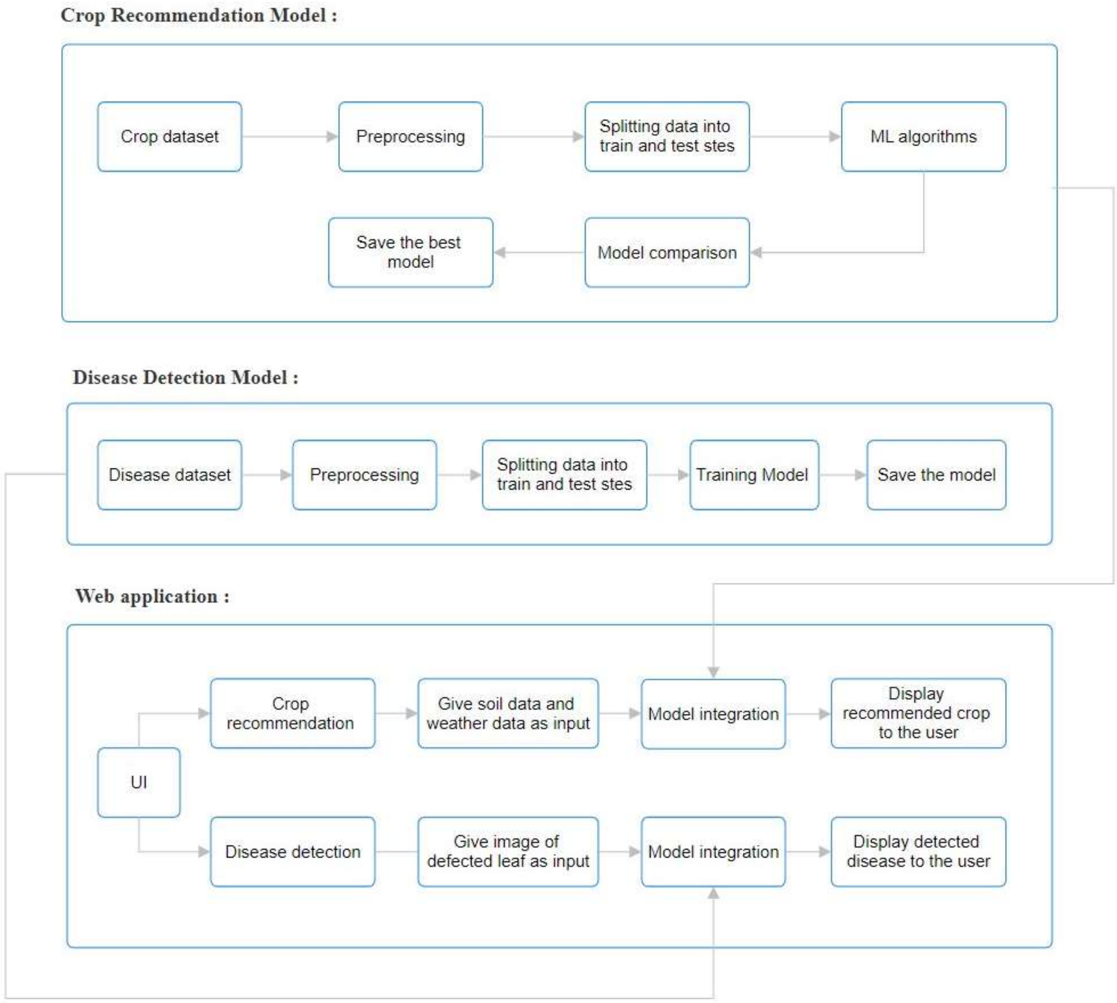

Crop Recommendation Model :
Crop dataset
Preprocessing
Splitting data into train and test stes
ML algorithms
Save the best model
Model comparison
Disease Detection Model :
Disease dataset
Preprocessing
Splitting data into train and test stes
Training Model
Save the model
Web application :
UI
Crop recommendation
Give soil data and weather data as input
Model integration
Display recommended crop to the user
Disease detection
Give image of defected leaf as input
Model integration
Display detected disease to the user

Figure 3.5: Data flow diagram of Backend

The diagram has three main components:

- **Crop Recommendation Model:** The section shows how the crop dataset is preprocessed, split into train and test sets, and then run through machine learning algorithms to generate a crop recommendation model.
- **Disease Detection Model:** The section shows how the disease dataset is preprocessed, split into train and test sets, and then run through a deep learning model to generate a disease detection model.
- **Web Application:** The section shows how the user inputs data into the UI, how the model integration works, and how the recommended crop and disease detection results are displayed to the user.

3.3.2 SEQUENCE DIAGRAM

The flow of interaction in our project can be described through the following sequence figure illustrated 3.6. The sequence diagram illustrates the interaction between the user, the web interface, the crop recommendation system, the disease detection system, and the IoT sensors.

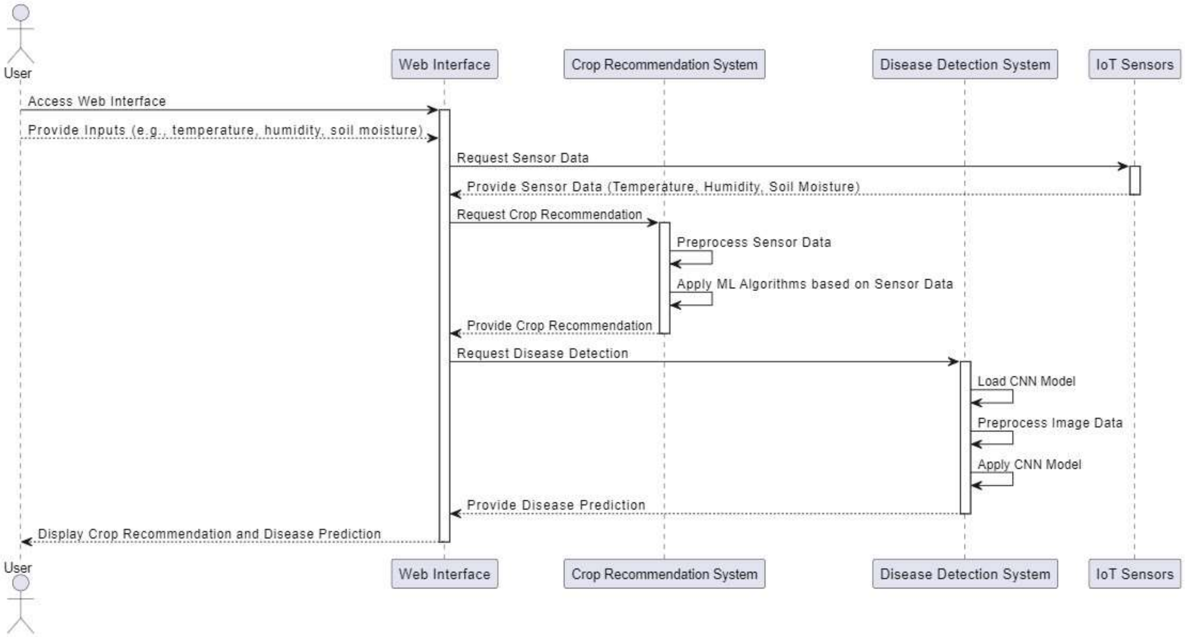

User
Web Interface
Crop Recommendation System
Disease Detection System
IoT Sensors
Access Web Interface
Provide Inputs (e.g., temperature, humidity, soil moisture)
Request Sensor Data
Provide Sensor Data (Temperature, Humidity, Soil Moisture)
Request Crop Recommendation
Preprocess Sensor Data
Apply ML Algorithms based on Sensor Data
Provide Crop Recommendation
Request Disease Detection
Load CNN Model
Preprocess Image Data
Apply CNN Model
Provide Disease Prediction
Display Crop Recommendation and Disease Prediction
User
Web Interface
Crop Recommendation System
Disease Detection System
IoT Sensors

Figure 3.6: Sequence Diagram

Sequence Description:

1. **User Interaction:** The process begins with the user accessing the web interface.
2. **Providing Inputs:** The user provides inputs such as temperature, humidity, and soil moisture through the web interface.
3. **Sensor Data Retrieval:** The web interface requests sensor data from the IoT sensors.
4. **Sensor Data Provision:** The IoT sensors provide sensor data (temperature, humidity, soil moisture) to the web interface.
5. **Crop Recommendation:** Upon receiving the sensor data, the web interface requests a crop recommendation from the crop recommendation system.
6. **Preprocessing and ML:** The crop recommendation system preprocesses the sensor data and applies machine learning algorithms (e.g., SVM, Decision Tree, Random Forest) to generate a crop recommendation.

7. **Disease Detection:** Additionally, the web interface requests disease detection from the disease detection system.
8. **ResNet Model Application**: The disease detection system loads a Resnet model, preprocesses image data, applies the ResNet model, and provides a disease prediction.
9. **User Output:** Finally, the web interface displays the crop recommendation and disease prediction to the user.

3.4 IOT Configuration

IoT, or the Internet of Things, revolutionizes how devices communicate and interact, enabling seamless connectivity and data exchange across various domains. By embedding sensors, actuators, and communication technologies into everyday objects, IoT systems gather valuable data from the physical world, enabling real-time monitoring, control, and automation. This interconnected network of devices facilitates smarter decision-making, enhances operational efficiency, and enables innovative applications across industries such as agriculture, healthcare, smart homes, and industrial automation. With IoT, the potential for transforming industries, improving quality of life, and driving innovation is vast, ushering in a new era of connected devices and intelligent systems.

3.4.1 DHT11 sensor

The DHT11 sensor is a fundamental component in many DIY electronics projects, offering a straightforward solution for monitoring temperature and humidity levels. Its simplicity lies in its single-wire digital interface, making it accessible even to beginners in electronics and programming. With a range spanning from 0°C to 50°C for temperature readings and 20% to 90% for relative humidity, the DHT11 sensor (Figure 3.7) provides sufficient accuracy for a wide array of applications, from environmental monitoring to home automation projects. Its low cost and widespread availability make it a go-to choice for hobbyists and educators.

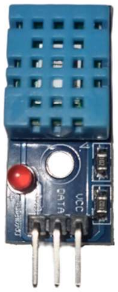


Figure 3.7: DHT11 Sensor

However, while the DHT11 sensor offers affordability and ease of use, it does come with limitations. Its accuracy, with a margin of error of around ±2°C for temperature and ±5% for humidity, may not be suitable for applications requiring high precision measurements. Additionally, its relatively slow sampling rate of approximately 2 seconds per reading may not be suitable for applications requiring real-time data acquisition. Despite these drawbacks, the DHT11 sensor remains a popular choice for prototyping and experimentation due to its simplicity and affordability, serving as an entry point into the world of sensor-based electronics projects.

3.4.2 Soil Moisture sensor

Soil moisture sensors are vital tools in agriculture, environmental monitoring, and automated gardening systems. These sensors measure the volumetric water content in soil, providing valuable data for optimizing irrigation schedules, preventing overwatering or underwatering, and monitoring soil health. Typically, soil moisture sensors work based on the principle of measuring the electrical conductivity or dielectric permittivity of the soil, which changes with variations in moisture content. They often consist of two electrodes or probes as shown in Figure 3.8 that are inserted into the soil, with the resistance or capacitance between them varying according to the soil's moisture level.

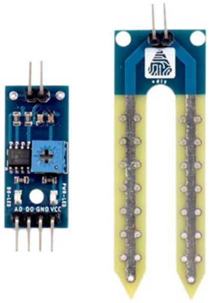


Figure 3.8: Soil Moisture Sensor

One of the key advantages of soil moisture sensors is their ability to provide real-time, localized data on soil moisture levels. This information allows farmers, gardeners, and researchers to make informed decisions about watering regimes, crop management, and environmental conservation efforts. Moreover, advancements in sensor technology have led to the development of wireless and IoT-enabled soil moisture sensors, enabling remote monitoring and control of irrigation systems. By integrating soil moisture sensors into automated irrigation systems or IoT platforms, users can optimize water usage, improve crop yields, and reduce water waste, contributing to more sustainable agricultural practices and resource management.

3.4.3 NodeMCU

NodeMCU is an open-source firmware and development kit based on the ESP8266 WiFi module. It allows for easy development of WiFi-enabled IoT projects and applications. The NodeMCU firmware is built on the Lua scripting language, which simplifies the programming process by providing a high-level scripting interface for interacting with the ESP8266 hardware.

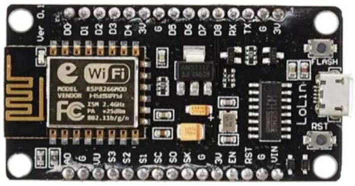


Figure 3.9: NodeMCU

One of the key features of NodeMCU is its ease of use and accessibility for both beginners and experienced developers. With its built-in USB-to-serial interface, developers can quickly upload code and interact with the device using the Lua interpreter, eliminating the need for additional hardware such as FTDI adapters. Additionally, NodeMCU supports a wide range of sensors, actuators, and communication protocols, making it versatile for various IoT applications as shown in Figure 3.9.

NodeMCU's integration with the ESP8266 module provides built-in WiFi connectivity, enabling devices to connect to local networks and the internet. This capability opens up possibilities for remote monitoring, control, and data logging applications. Furthermore, NodeMCU's compatibility with the Arduino IDE allows developers to leverage the extensive Arduino ecosystem, libraries, and community support while benefiting from the ESP8266's WiFi capabilities.

3.4.4 Hardware setup

- Connect the DHT11 sensor to the NodeMCU board. The DHT11 typically has three pins: VCC (power), data, and ground. Connect VCC to a 3.3V pin on the NodeMCU, data to any digital pin (e.g., D2), and ground to a ground pin.
- Similarly, connect the soil moisture sensor to the NodeMCU. It usually has two probes that need to be inserted into the soil. Connect VCC to a 3.3V pin on the NodeMCU, data to any analog pin (e.g., A0), and ground to a ground pin. The hardware setup is shown in Figure 3.10.

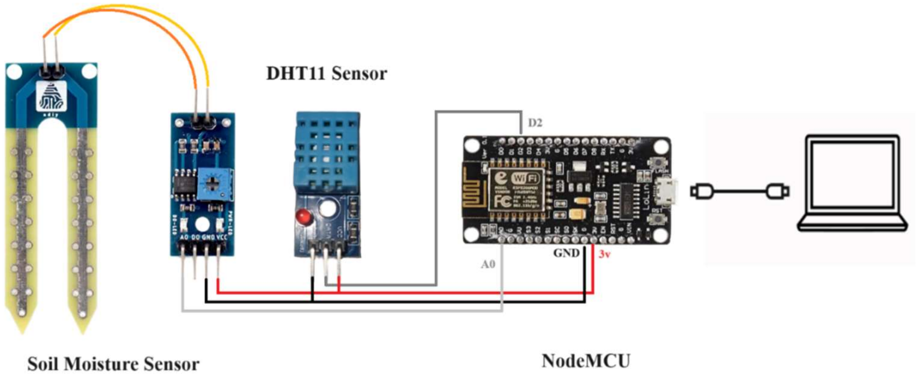


Figure 3.10: IoT Hardware Setup

3.5 Software setup

The section outlines the installation process of necessary software such as Arduino IDE and libraries, as well as the configuration steps for the NodeMCU board. Additionally, it details the setup of XAMPP server, including Apache, MySQL, PHP, and Perl, along with the configuration of a MySQL database for storing sensor data and the integration of a PHP script for data storage.

3.5.1 Installation of Arduino IDE and Libraries

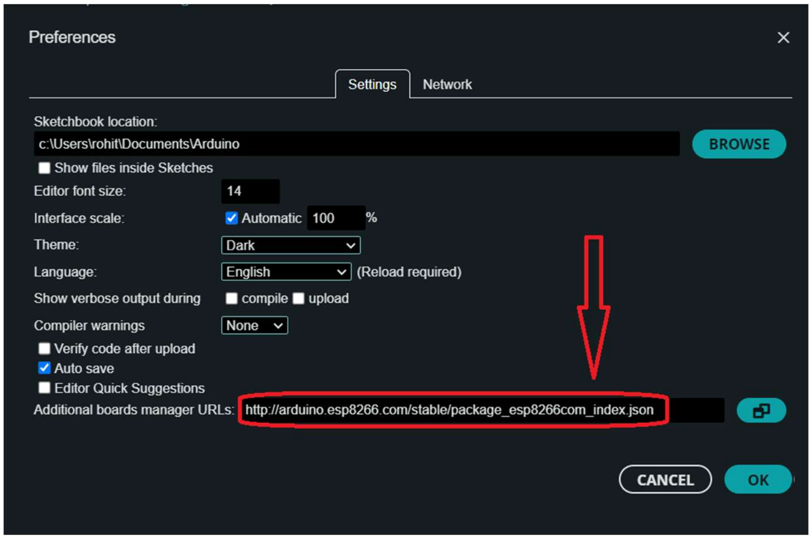
Arduino IDE Installation: We installed the Arduino Integrated Development Environment (IDE) from the official Arduino website (arduino.cc). This software provides the necessary tools for writing, compiling, and uploading code to microcontroller boards like the NodeMCU.
Library Installation:

- **Adafruit Unified Sensor:** To interface with sensors, we installed the Adafruit Unified Sensor library via the Arduino IDE Library Manager. This library offers a standardized interface for working with various sensors, simplifying the development process.
- **DHT Sensor Library:** For the DHT11 temperature and humidity sensor, we utilized the DHT sensor library by Adafruit. This library facilitates communication with DHT series sensors, enabling us to retrieve accurate temperature and humidity readings.
- **MySQL Connector Arduino:** To establish communication between the NodeMCU and our MySQL database, we installed the MySQL Connector Arduino library by Dr. Charles Bell. This library enables the NodeMCU to insert data into the MySQL database via PHP scripts.

3.5.2 Configuration of NodeMCU Board in Arduino IDE

We configured the NodeMCU board in the Arduino IDE by following official documentation.

- Go to **File > Preferences**
- In the Preferences window, find the field labeled **“Additional Board Manager URLs”**.
- This field typically contains one or more URLs separated by commas.
- To add a new URL, click on the icon at the right end of the field (it looks like a small rectangle with an arrow pointing to the right). This will open a new window where you can enter the URL.
- Enter the URL of the additional board manager package index file you want to add (http://arduino.esp8266.com/stable/package_esp8266com_index.json) as shown in Figure3.11.



Sketchbook location:
c:\Users\rohit\Documents\Arduino

☐ Show files inside Sketches

Editor font size: 14

Interface scale: ☑ Automatic 100 %

Theme: Dark

Language: English (Reload required)

Show verbose output during ☐ compile ☐ upload

Compiler warnings None

☐ Verify code after upload

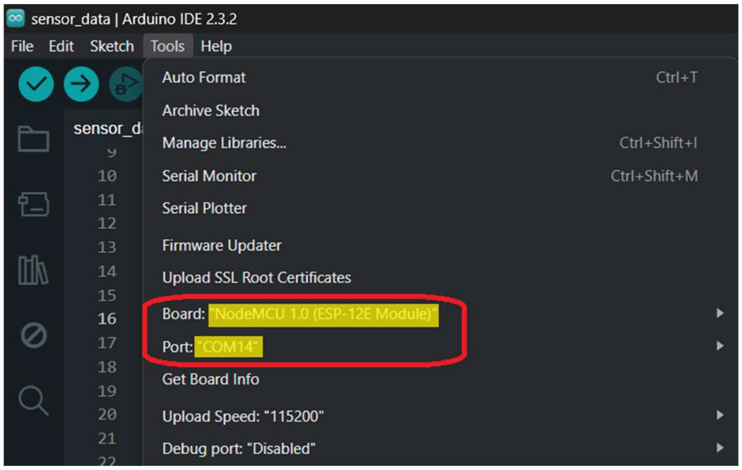
☑ Auto save

☐ Editor Quick Suggestions

Additional boards manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

Figure 3.11: ESP8266 Package

- Once you've added the URL(s) of the additional package index files, click OK to close the Preferences window.
- Now, go to **Tools > Board > Boards Manager**.
- The Arduino IDE will connect to the URLs you provided and fetch the package index files.
- In the Boards Manager window, you should see additional board packages listed alongside the default ones.
- Click on the desired board package (ESP8266) and click the "Install" button to install it.


sensor_data | Arduino IDE 2.3.2
File
Edit
Sketch
Tools
Help
sensor_d
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Get Board Info

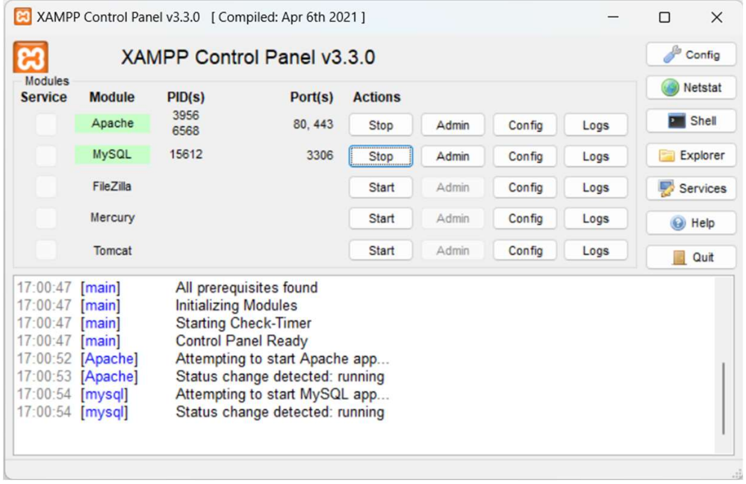
Board: "NodeMCU 1.0 (ESP-12E Module)"
Port: "COM14"
Upload Speed: "115200"
Debug port: "Disabled"

Figure 3.12: NodeMCU Board Setup

- After the installation is complete, you can select the newly installed boards from the **Tools > Board** menu, select **"NodeMCU 1.0 (ESP-12E Module)"** in board menu and desired port in port menu as shown in Figure 3.12.

3.5.3 XAMPP Server Setup

Installation: XAMPP, a software package containing Apache, MySQL, PHP, and Perl, was installed on our local machine. We downloaded the latest version compatible with our operating system from "https://www.apachefriends.org" and followed the provided installation instructions.


XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]
XAMPP Control Panel v3.3.0
Modules
Config
Netstat
Shell
Explorer
Services
Help
Quit

Service	Module	PID(s)	Port(s)	Actions			
	Apache	3956 6568	80, 443	Stop	Admin	Config	Logs
	MySQL	15612	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

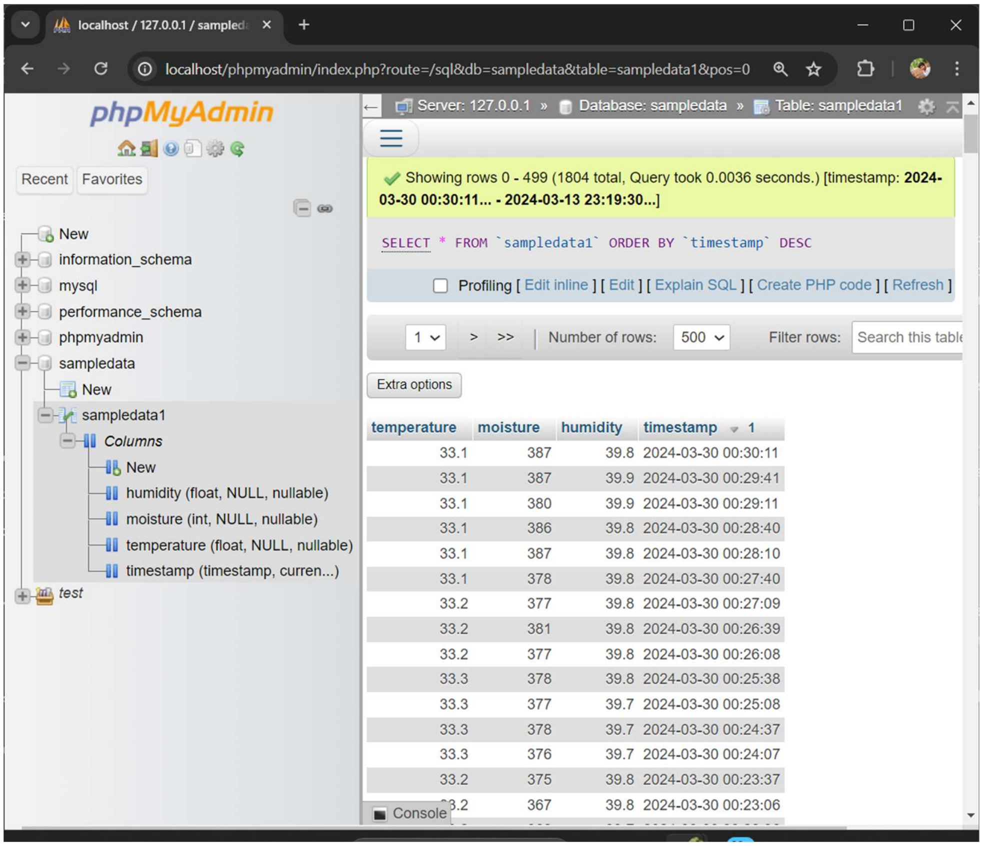
```
17:00:47  [main]     All prerequisites found
17:00:47  [main]     Initializing Modules
17:00:47  [main]     Starting Check-Timer
17:00:47  [main]     Control Panel Ready
17:00:52  [Apache]   Attempting to start Apache app...
17:00:53  [Apache]   Status change detected: running
17:00:54  [mysql]    Attempting to start MySQL app...
17:00:54  [mysql]    Status change detected: running
```

Figure 3.13: Apache and MySQL Servers

We utilized the XAMPP Control Panel to start the Apache and MySQL servers, providing the necessary environment for hosting web applications and managing databases.

3.5.4 MySQL Database Configuration

Using phpMyAdmin as shown in Figure 3.14, accessible through the XAMPP Control Panel or via **“http://localhost/phpmyadmin”** in a web browser, we created a new database named **“sampledata”**. Within this database, we established a table named **“sensordata1”** to store sensor data, defining appropriate fields for temperature, humidity, and moisture readings.



localhost / 127.0.0.1 / sampled
localhost/phpmyadmin/index.php?route=/sql&db=sampledata&table=sampledata1&pos=0
phpMyAdmin
Recent
Favorites
Server: 127.0.0.1 » Database: sampledata » Table: sampledata1

- New
- information_schema
- mysql
- performance_schema
- phpmyadmin
- sampledata
  - New
  - sampledata1
    - *Columns*
      - New
      - humidity (float, NULL, nullable)
      - moisture (int, NULL, nullable)
      - temperature (float, NULL, nullable)
      - timestamp (timestamp, curren...)
- *test*

✓ Showing rows 0 - 499 (1804 total, Query took 0.0036 seconds.) [timestamp: **2024-03-30 00:30:11... - 2024-03-13 23:19:30...**]

```
SELECT * FROM `sampledata1` ORDER BY `timestamp` DESC
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
1
>
>>
Number of rows: 500
Filter rows: Search this table
Extra options
Console

temperature	moisture	humidity	timestamp 1
33.1	387	39.8	2024-03-30 00:30:11
33.1	387	39.9	2024-03-30 00:29:41
33.1	380	39.9	2024-03-30 00:29:11
33.1	386	39.8	2024-03-30 00:28:40
33.1	387	39.8	2024-03-30 00:28:10
33.1	378	39.8	2024-03-30 00:27:40
33.2	377	39.8	2024-03-30 00:27:09
33.2	381	39.8	2024-03-30 00:26:39
33.2	377	39.8	2024-03-30 00:26:08
33.3	378	39.8	2024-03-30 00:25:38
33.3	377	39.7	2024-03-30 00:25:08
33.3	378	39.7	2024-03-30 00:24:37
33.3	376	39.7	2024-03-30 00:24:07
33.2	375	39.8	2024-03-30 00:23:37
33.2	367	39.8	2024-03-30 00:23:06

Figure 3.14: Database Setup

PHP Script for Data Storage

- The provided PHP script (store_data.php) was integrated into our XAMPP server's web root directory ("C:\xampp\htdocs\store_data.php"). This script serves as the endpoint for receiving sensor data from the NodeMCU and inserting it into the MySQL database.
- To ensure seamless data insertion, we updated the PHP script with the MySQL database credentials, including the server address, username, password, database name, and table name.

3.6 MODEL BUILDING

The section deals with various machine learning algorithms, including Random Forest, Support Vector Machine, k-Nearest Neighbors (KNN), Decision Tree, and XGBoost, are considered from crop recommendation and deep learning model ResNet for image classification to detect disease. Each algorithm has distinct characteristics and applications. The discussion involves their implementation, parameter tuning, and evaluation to identify the most suitable model for the specific task or problem.

3.6.1 LOGISTIC REGRESSION

Logistic regression is a supervised machine learning algorithm used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not. Logistic regression is a statistical algorithm which analyze the relationship between two data factors.

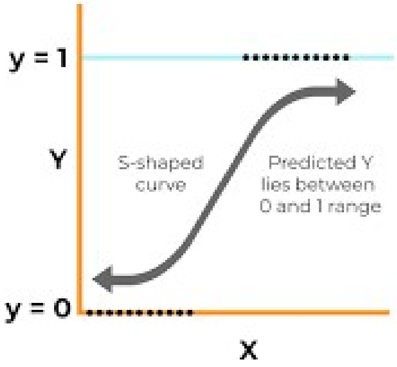


Figure 3.15: Showcasing Logistic regression

It is used for binary classification where we use sigmoid function, that takes input as independent variables and produces a probability value between 0 and 1. Logistic regression as shown in Figure 3.15 predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value.

3.6.2 RANDOM FOREST

Random Forest Algorithm widespread popularity stems from its user-friendly nature and adaptability, enabling it to tackle both classification and regression problems effectively. The algorithm’s strength lies in its ability to handle complex datasets and mitigate overfitting, making it a valuable tool for various predictive tasks in machine learning.

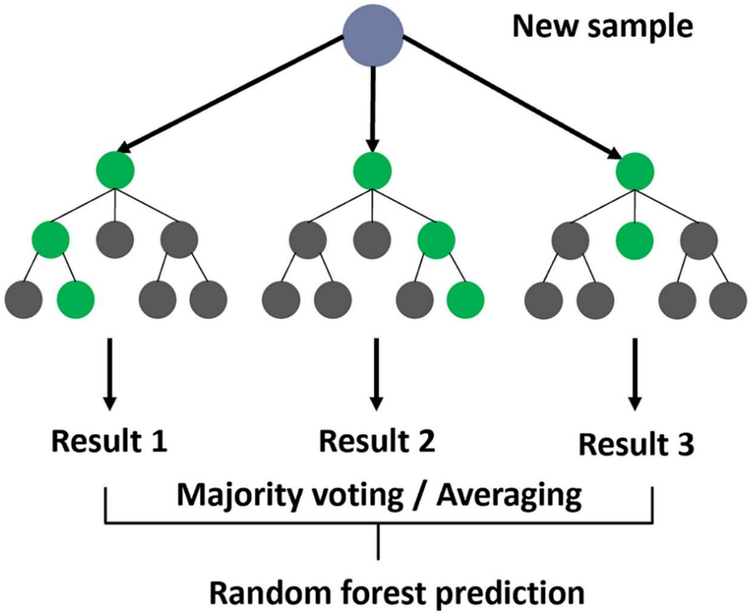


Figure 3.16: Tree of Random Forest

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables, as in the case of regression, and categorical variables**,** as in the case of classification. It performs better for classification and regression tasks. In the figure 3.16, we will understand the working of random forest and implement random forest on a classification task.

3.6.3 SUPPORT VECTOR CLASSIFIER

Support Vector Classifier (SVC) is a type of supervised learning algorithm used for classification tasks. It belongs to the family of Support Vector Machines (SVM), which is a powerful and versatile algorithm for both classification and regression.

The main idea behind SVC is to find the optimal hyperplane that best separates the different classes in the feature space. This hyperplane is chosen to maximize the margin between the classes, where the margin is defined as the distance between the hyperplane and the nearest data points from each class, known as support vectors.

In essence, SVC aims to find the decision boundary that maximizes the margin while minimizing classification errors. It can handle linearly separable as well as non-linearly separable data by using different kernel functions such as linear, polynomial, radial basis function (RBF), or sigmoid.

During training, SVC learns the parameters (coefficients and intercept) of the hyperplane by solving an optimization problem based on the training data and their corresponding labels. Once trained, the SVC model can be used to predict the class labels of new data points based on their feature representations.

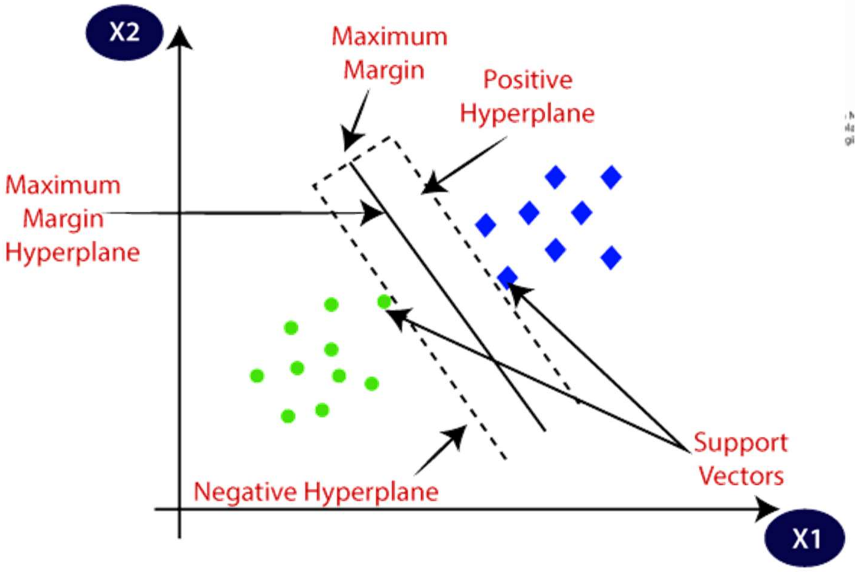


Figure 3.17: Representation of SVC

Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (as shown in the figure 3.17).

3.6.4 KNN

The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today. It operates on the principle of similarity, where a new data point is classified based on the majority class of its k nearest neighbors in the feature space. However, its performance can be sensitive to the choice of the distance metric and the value of k. Despite this, KNN (as shown in the figure 3.18) remains a versatile tool in various domains, particularly when dealing with small to medium-sized datasets where its computational overhead is manageable.

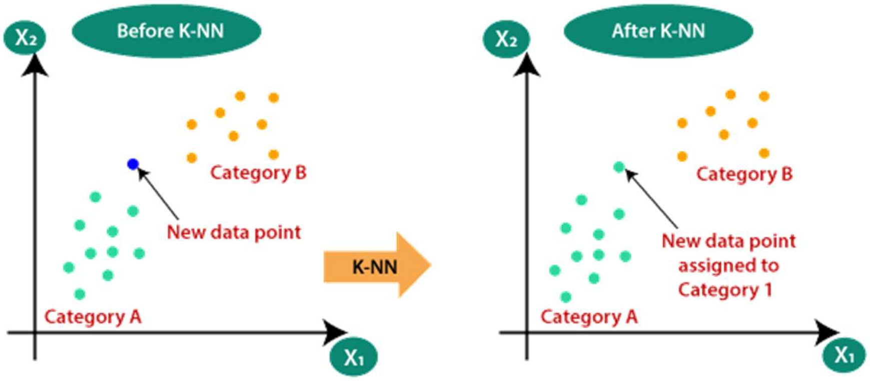


Figure 3.18: Illustration of KNN

3.6.5 LightGBM

LightGBM, a powerful gradient boosting framework, offers an array of parameters and hyperparameters to fine-tune its models. These parameters play a critical role in determining the model's performance, speed, and generalization ability. It constructs a strong learner by sequentially adding weak learners (as shown in Figure 3.19) in a gradient descent manner. It’s focus on optimizing the training process for speed and resource utilization sets it apart. Also optimizes memory usage and training time with techniques like Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) drastically reduce computation time and memory usage without compromising model accuracy. Moreover, LightGBM's histogram-based splitting technique accelerates the search for optimal split points during tree construction. This efficiency extends to parallel and distributed computing, allowing seamless scaling across multiple machines and leveraging GPU resources for accelerated training. Also, the Figure 3.20 shows the LightGBM’s Architecture.

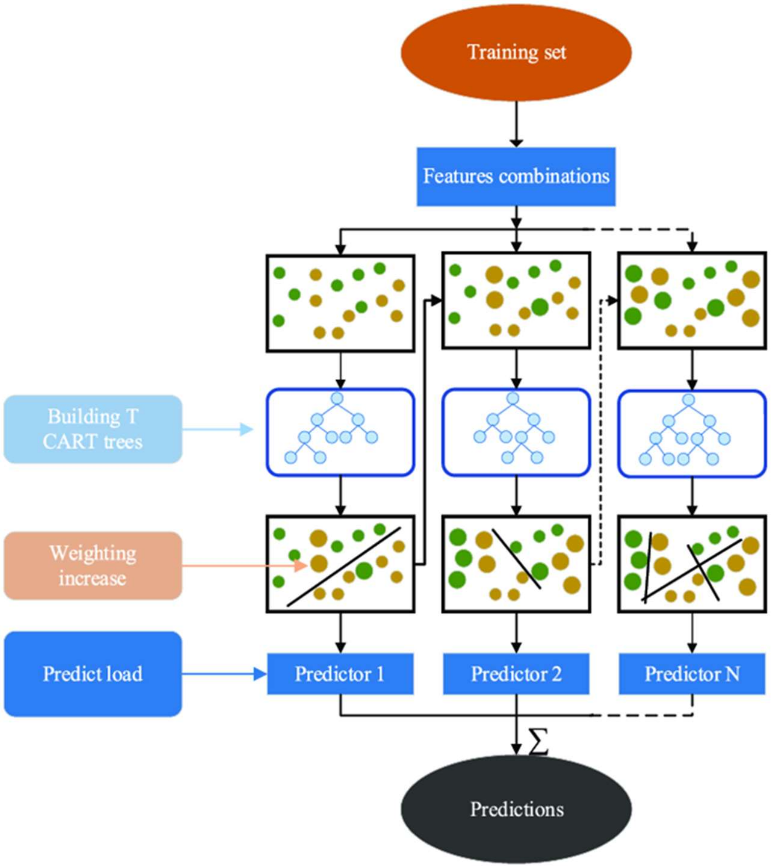


Figure 3.19: Illustration of LightGBM

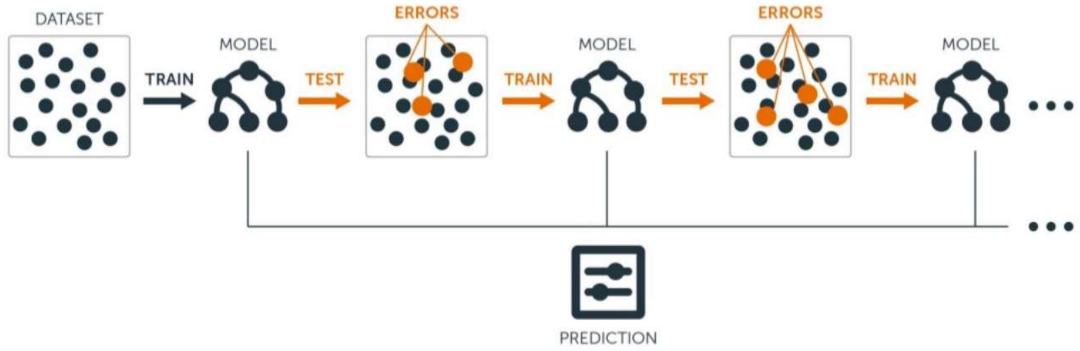


Figure 3.20: LightGBM Architecture

3.6.6 XGBoost

XGBoost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction as illustrated in Figure 3.21. XGBoost stands for "Extreme Gradient Boosting" and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression.

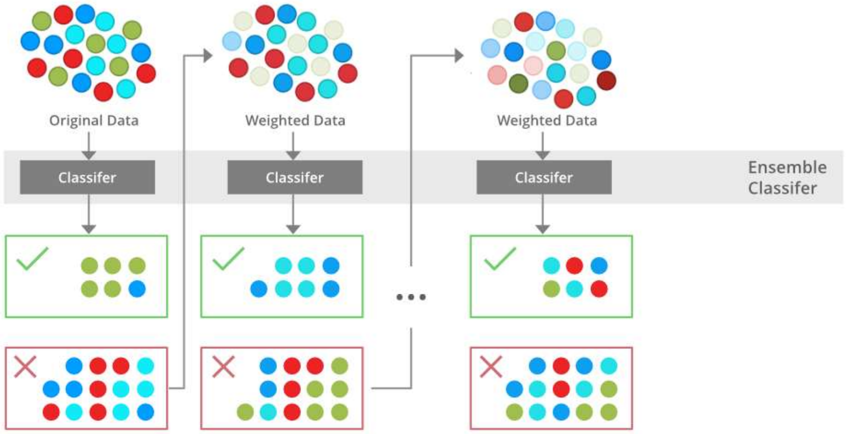


Figure 3.21: Illustration of XGBoost

One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.

3.6.7 ResNet

In contrast to traditional neural networks, which rely on direct connections from one layer to the next, ResNet employs residual blocks. These blocks enable each layer to feed not only into the subsequent layer but also directly into layers positioned approximately 2–3 hops away, thus facilitating skip connections. This architectural feature aims to mitigate issues such as overfitting, where validation loss stagnates and then increases despite declining training loss, and it also helps address the vanishing gradient problem, allowing for the effective training of deep neural networks.

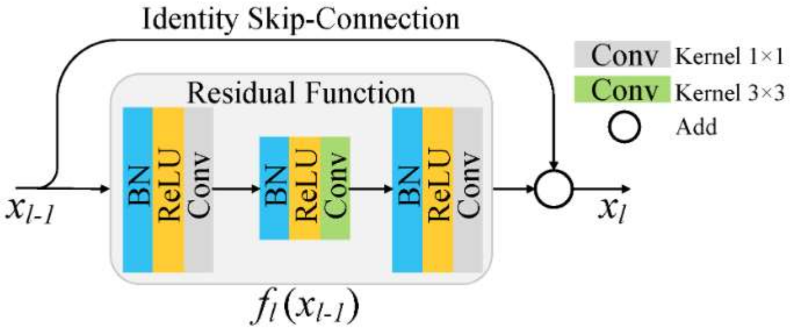


Figure 3.22: Residual Block

In figure 3.22, residual Blocks are shown which are skip-connection blocks that learn residual functions with reference to the layer inputs, instead of learning unreferenced functions

F(x) := H(x) - x which gives H(x) := F(x) + x

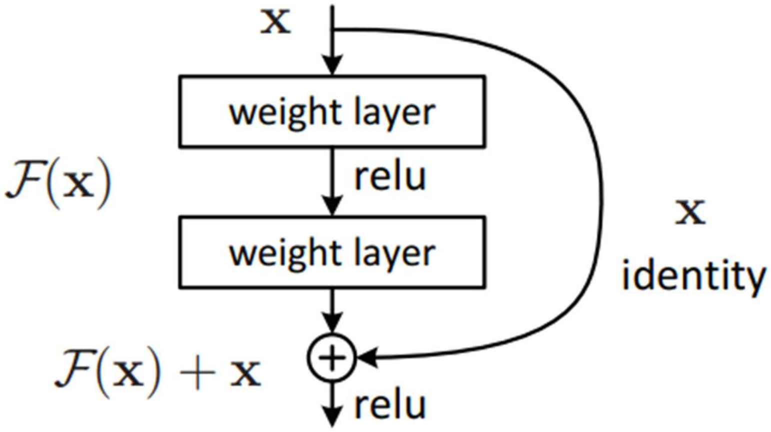

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 3.23: Skip (Shortcut) connection

The key innovation of ResNet is the use of residual blocks, which allow for the training of very deep networks without encountering the vanishing gradient problem. The vanishing gradient problem occurs when gradients become too small as they are backpropagated through the layers of a deep neural network, making it difficult to update the weights of earlier layers and hindering the learning process.

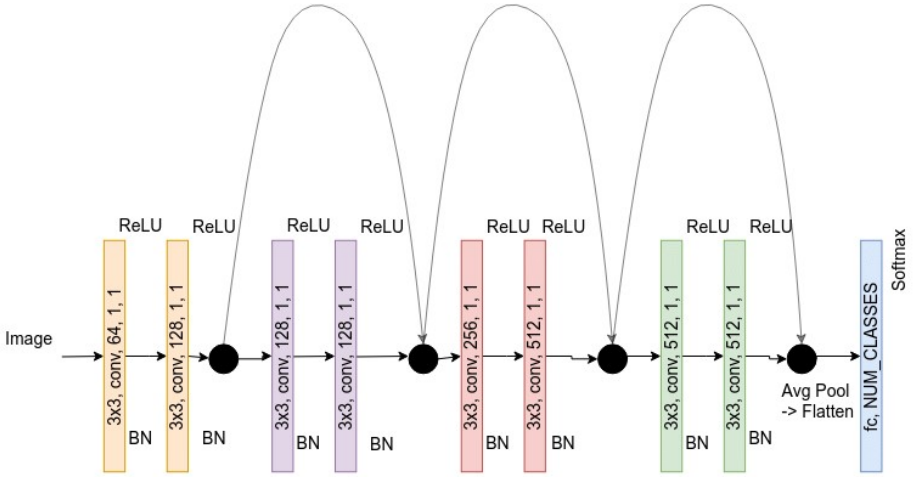


Figure 3.24: ResNet architecture

In a standard neural network as in figure 3.24, the output of each layer is directly connected to the next layer. Residual networks introduce skip connections, also known as shortcut connections, that skip one or more layers. The idea is to learn the residual, or the difference between the input and the output of a layer, and add this residual to the output of the layer. This way, even if a layer contributes little to the overall performance of the network, the information can still be passed through the skip connection.

The architecture of a ResNet is often depicted in terms of blocks. A basic building block consists of two convolutional layers with batch normalization and ReLU activation functions, and the input is added to the output via a skip connection. ResNet architectures come in different depths, such as ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, with the number indicating the total number of layers.

ResNet architectures have been widely used for image classification tasks and have achieved state-of-the-art results on various benchmarks. They have also been applied to other computer vision tasks, such as object detection and segmentation. The residual learning approach introduced by ResNet has influenced the design of subsequent deep neural network architectures.

4. IMPLEMENTATION AND TEST CASES

4.1 Metrics Evaluation

In the realm of machine learning, the evaluation of classifiers is paramount, serving as the litmus test for their effectiveness in solving real-world problems. So, evaluating the performance of various classifiers stands as a pivotal step in ensuring the efficacy and reliability of our solution. One of the most fundamental metrics in this evaluation process is accuracy, which measures the overall correctness of a model's predictions. However, accuracy alone can be deceiving, especially in scenarios with imbalanced datasets.

To delve deeper into the performance of classifiers, we explore additional metrics such as precision, recall, and F1-score. Precision illuminates the classifier's ability to correctly identify positive samples, while recall sheds light on its capacity to capture all positive instances. The F1-score, a harmonic mean of precision and recall, offers a balanced view of a classifier's performance, particularly beneficial in scenarios where false positives and false negatives carry differing weight.

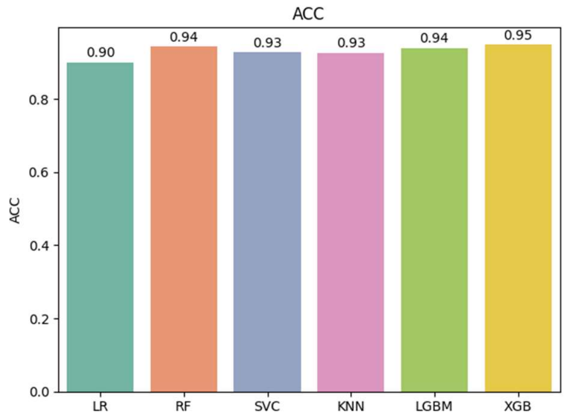


Figure 4.1: Accuracy

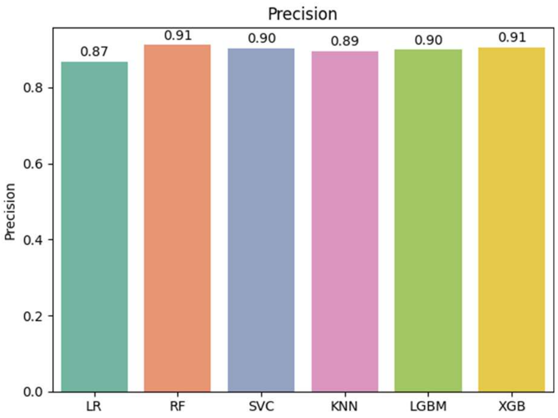


Figure 4.2: Precision

It looks like all models perform relatively well in terms of accuracy (Figure 4.1) with Random Forest (RF) and XGBoost (XGB) performing slightly better than the others, achieving accuracies of 94.32% and 94.77% respectively.

In the case pf Figure 4.2, XGBoost (XGB) has the highest precision of 90.59%, indicating that it has a good ability to avoid false positives.

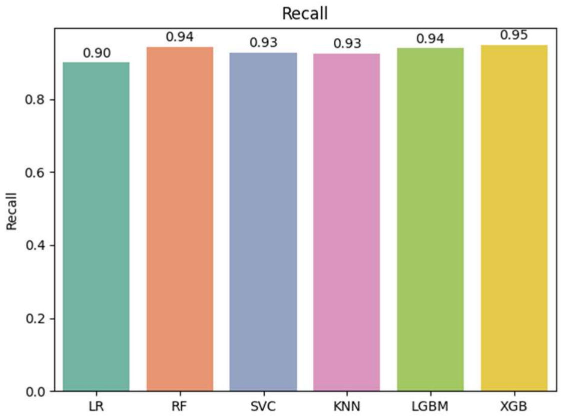


Figure 4.3: Recall

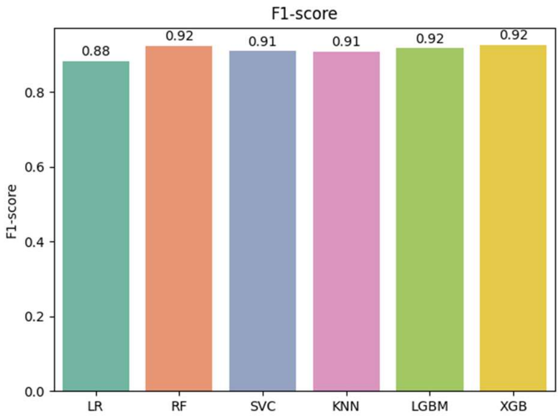


Figure 4.4: F1-score

Recall in Figure 4.3 measures the ability of a classifier to find all positive samples where XGBoost (XGB) again performs the best in terms of recall, achieving a score of 94.77%, indicating that it is able to capture most of the positive samples.

Also in Figure 4.4, F1-score providing a balance between the two metrics. XGBoost (XGB) has the highest F1-score of 92.49%, indicating that it has a good balance between precision and recall.

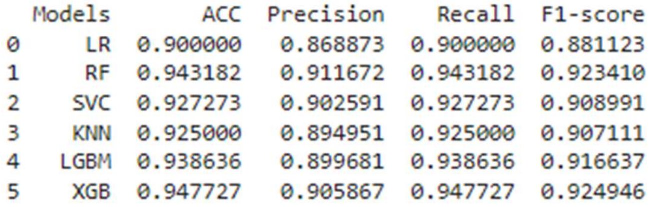

```
  Models       ACC  Precision    Recall  F1-score
0     LR  0.900000   0.868873  0.900000  0.881123
1     RF  0.943182   0.911672  0.943182  0.923410
2    SVC  0.927273   0.902591  0.927273  0.908991
3    KNN  0.925000   0.894951  0.925000  0.907111
4   LGBM  0.938636   0.899681  0.938636  0.916637
5    XGB  0.947727   0.905867  0.947727  0.924946
```

Figure 4.5: Metrics Evaluation Across Models

Overall, XGBoost (XGB) seems to be the best-performing model across all metrics (Figure 4.5), followed closely by Random Forest (RF). Both models exhibit strong performance in accuracy, precision, recall, and F1-score. These insights highlight the strengths of XGB and LGBM for this classification task, while also revealing potential areas for improvement or ensemble techniques involving the other models.

4.2 Distribution

The part visualizes the distributions of three key variables in our dataset: 'N', 'P', and 'K'. These variables represent crucial factors in our analysis and understanding their distributions is fundamental to grasping the characteristics of our data.

NITROGEN CONTENT (N):

The distribution plot for nitrogen content provides insights into the spread and central tendency of nitrogen levels across the dataset. Understanding the distribution of nitrogen content is essential as it is a primary nutrient affecting plant growth and soil fertility. There are two significant peaks, suggesting high frequencies for certain values of N around 40 and 30, with counts nearing or surpassing 300.Following the peak at 'N' = 60, there is a noticeable decline in count, indicating a decrease in the frequency of samples with nitrogen content around this value. However, this decline is followed by a smaller peak around 'N' = 90, suggesting a secondary cluster of samples with relatively higher nitrogen content.

PHOSPHORUS CONTENT (P):

By visualizing the distribution of phosphorus content, we gain an understanding of the variability and concentration levels of phosphorus within our dataset. Phosphorus is a vital nutrient for plant growth, and its distribution informs us about its availability in the soil samples. The distribution of phosphorus content reveals intriguing patterns within the dataset. A notable peak occurs around the 'P' = 60 mark on the x-axis, where the count exceeds 200, suggesting a high frequency for this range of phosphorus levels.

POTASSIUM CONTENT (K):

The distribution plot for potassium content illustrates the distribution of potassium levels in our dataset. Potassium is another essential nutrient for plants, influencing various physiological processes. Examining its distribution aids in understanding the potassium dynamics within the soil samples. Two notable peaks are observed around 'K' = 20 on the x-axis, where the count exceeds 300, suggesting a high frequency for this range of potassium levels. The observation indicates that crops tend to grow well in soil samples with potassium content around this value.

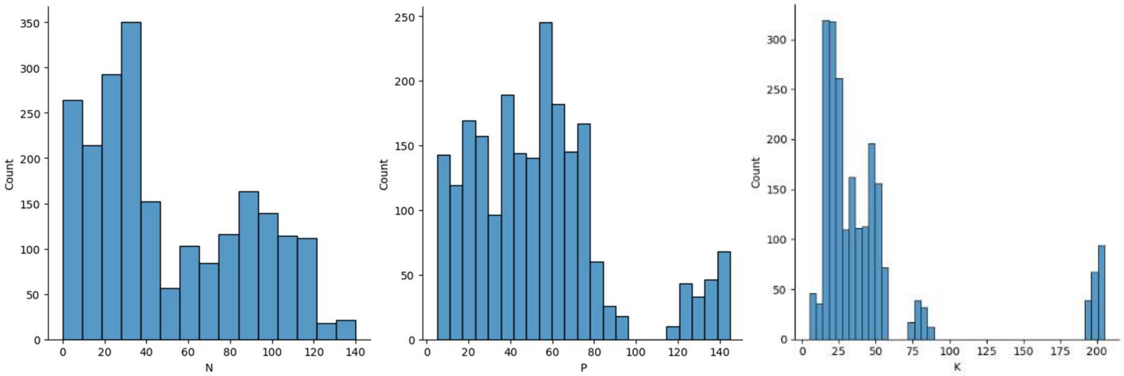


Figure 4.6: Distributions of N, P and K

These distribution plots (above Figure 4.1) serve as initial exploratory analyses, allowing us to identify any potential outliers, assess the overall shape of the distributions, and gain insights into the variability of the nutrient levels across the samples. Such visualizations are foundational for further statistical analyses and modeling efforts.

4.3 Correlation Heatmap

The correlation heatmap analysis reveals valuable insights into the relationships between environmental factors and crop growth. Key findings include a moderate positive correlation between rainfall and pH, indicating that areas with higher rainfall may have slightly higher soil pH levels. Additionally, temperature shows weak associations with humidity and pH, suggesting potential impacts on soil conditions. Notably, strong positive correlations are observed between potassium and nitrogen, as well as phosphorus and nitrogen, highlighting the interconnectedness of these essential nutrients for crop growth (seen in Figure 4.4). These insights are crucial for crop recommendation, guiding stakeholders to identify regions with optimal environmental conditions for maximizing agricultural productivity.

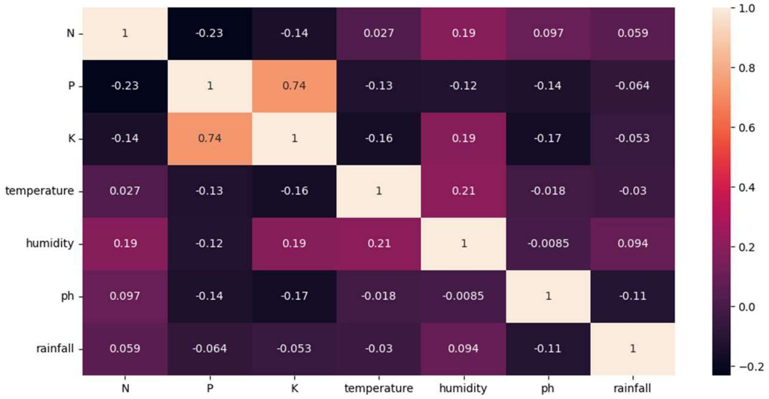
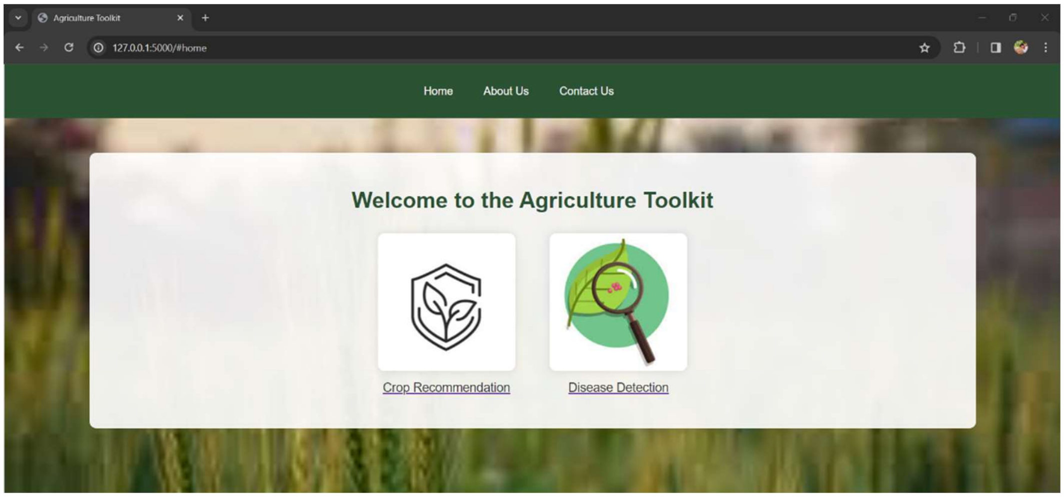


Figure 4.7: Correlation Map

5. RESULTS

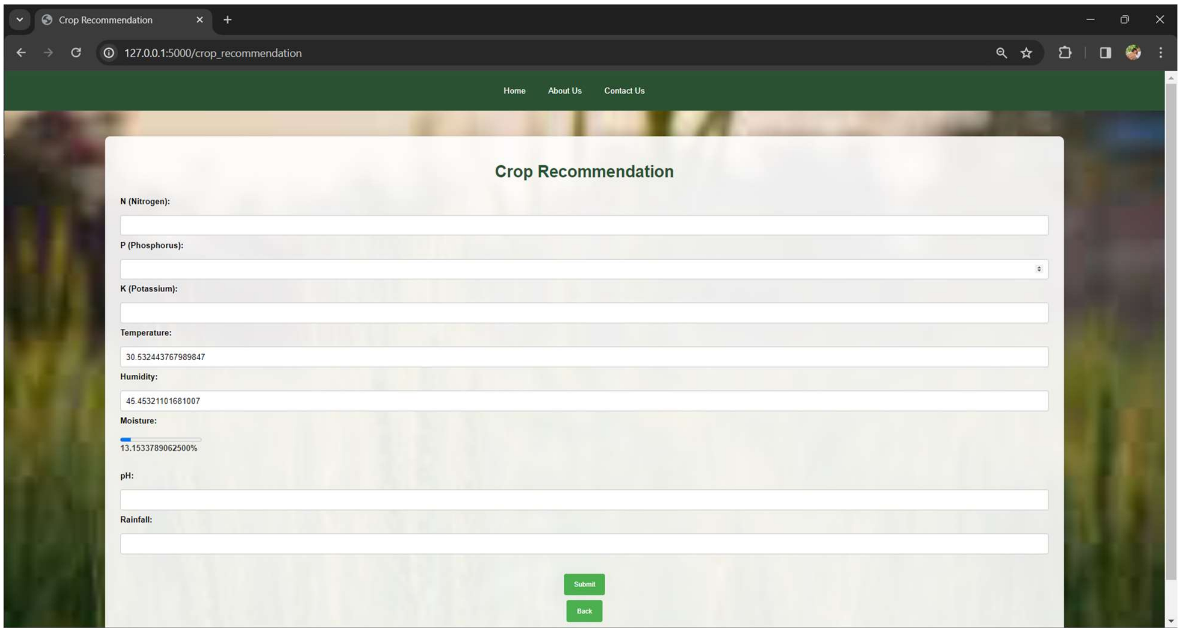
Creating a web interface using Flask, HTML, and CSS provides users with an interactive platform to explore agricultural data and make informed decisions. By offering two distinct options - crop recommendation and disease detection shown below in Figure 5.1.



Welcome to the Agriculture Toolkit

Figure 5.1: Web Interface

1) Users have the flexibility to choose the functionality that best suits their needs. By providing both manual input options and the potential for automatic sensor data retrieval, the interface accommodates various user preferences and data availability scenarios as in Figure 5.2.



Crop Recommendation

N (Nitrogen):

P (Phosphorus):

K (Potassium):

Temperature:

30.532443767989847

Humidity:

45.4532110168100

Moisture:

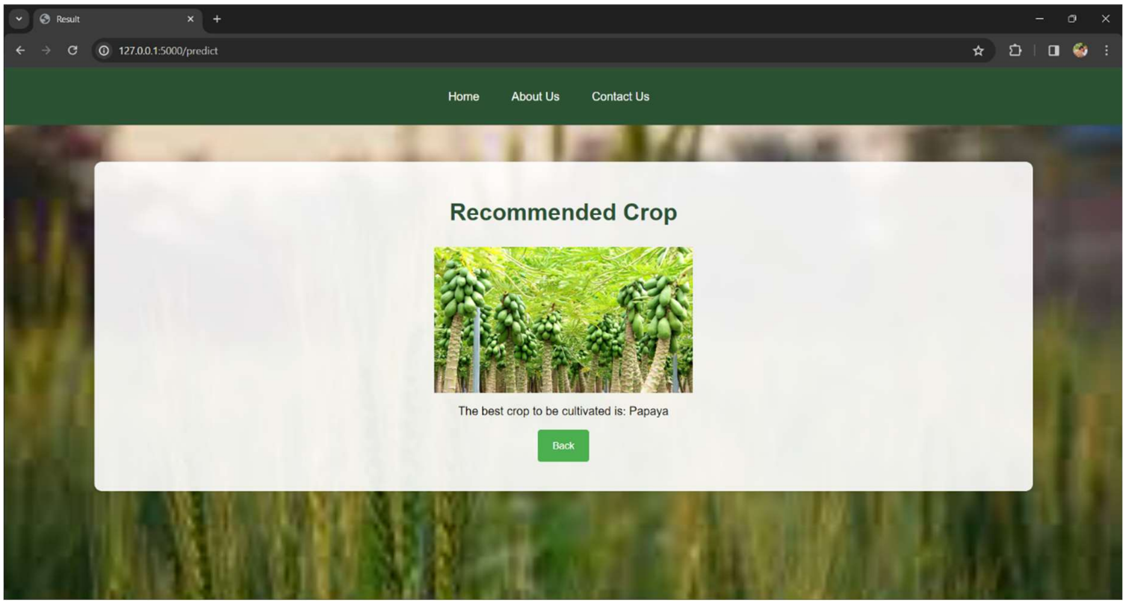
13.1533789062500%

pH:

Rainfall:

Figure 5.2: Crop Recommendation

Upon entering these parameters, the system utilizes a recommendation algorithm to suggest suitable crops based on the provided environmental conditions. The personalized recommendation feature enhances user experience and facilitates informed decision-making in agricultural practices.

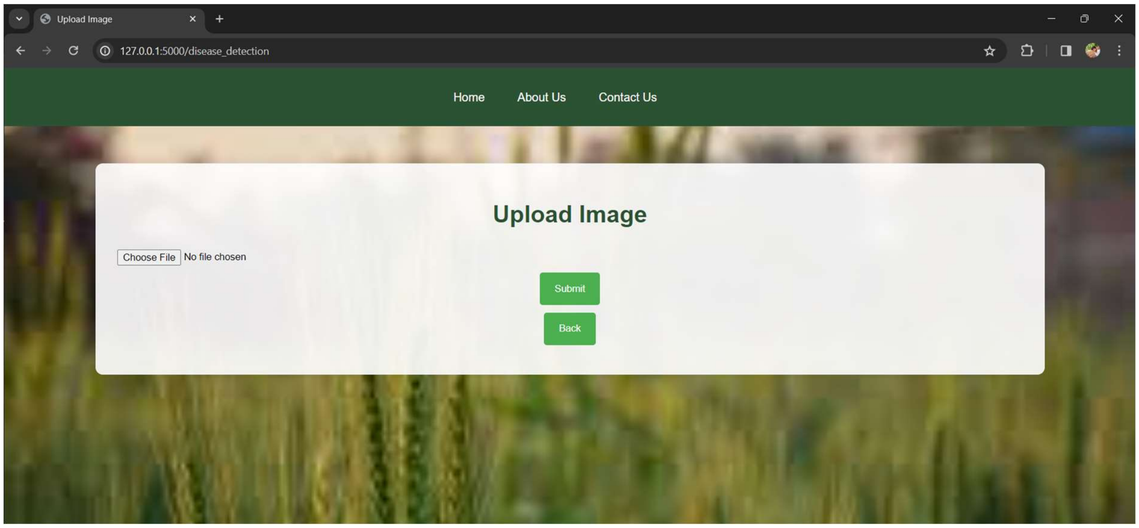


Recommended Crop

The best crop to be cultivated is: Papaya

Figure 5.3: Recommended Crop (output)

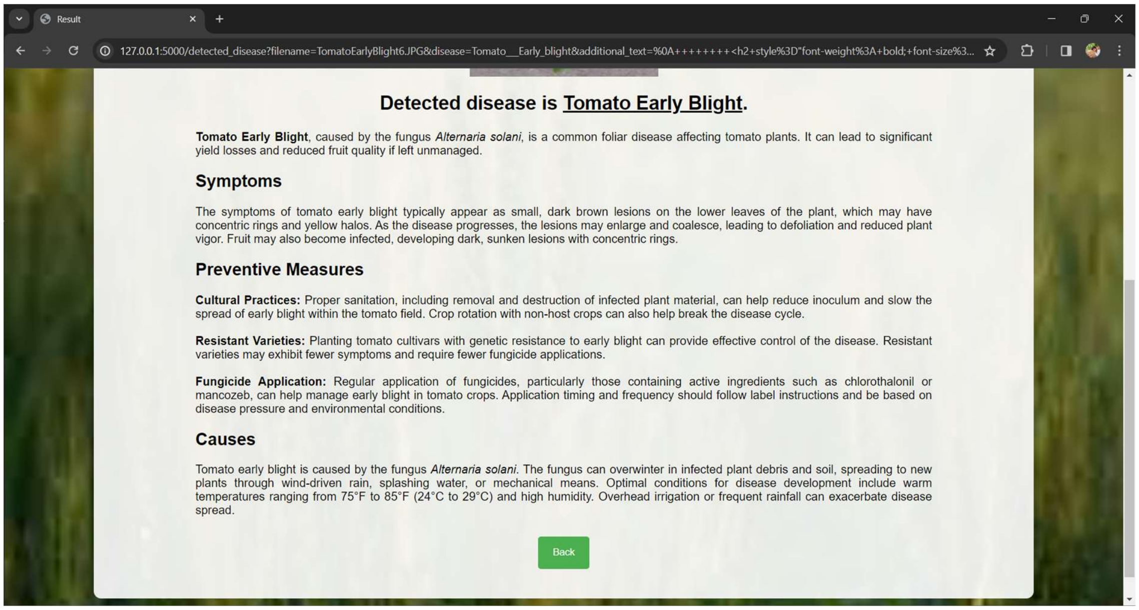
2) On other hand of functionality, users can upload images of plant leaves affected by diseases as in Figure 5.4. The system will then analyze the images using image processing techniques to detect symptoms of diseases.



Upload Image

No file chosen

Figure 5.4: Predictive Interface


Result
127.0.0.1:5000/detected_disease?filename=TomatoEarlyBlight6.JPG&disease=Tomato___Early_blight&additional_text=%0A++++++++<h2+style%3D"font-weight%3A+bold;+font-size%3...

Detected disease is Tomato Early Blight.

Tomato Early Blight, caused by the fungus *Alternaria solani*, is a common foliar disease affecting tomato plants. It can lead to significant yield losses and reduced fruit quality if left unmanaged.

Symptoms

The symptoms of tomato early blight typically appear as small, dark brown lesions on the lower leaves of the plant, which may have concentric rings and yellow halos. As the disease progresses, the lesions may enlarge and coalesce, leading to defoliation and reduced plant vigor. Fruit may also become infected, developing dark, sunken lesions with concentric rings.

Preventive Measures

Cultural Practices: Proper sanitation, including removal and destruction of infected plant material, can help reduce inoculum and slow the spread of early blight within the tomato field. Crop rotation with non-host crops can also help break the disease cycle.

Resistant Varieties: Planting tomato cultivars with genetic resistance to early blight can provide effective control of the disease. Resistant varieties may exhibit fewer symptoms and require fewer fungicide applications.

Fungicide Application: Regular application of fungicides, particularly those containing active ingredients such as chlorothalonil or mancozeb, can help manage early blight in tomato crops. Application timing and frequency should follow label instructions and be based on disease pressure and environmental conditions.

Causes

Tomato early blight is caused by the fungus *Alternaria solani*. The fungus can overwinter in infected plant debris and soil, spreading to new plants through wind-driven rain, splashing water, or mechanical means. Optimal conditions for disease development include warm temperatures ranging from 75°F to 85°F (24°C to 29°C) and high humidity. Overhead irrigation or frequent rainfall can exacerbate disease spread.

Figure 5.5: Disease prediction Analysis

After detection, the system can provide preventive measures and symptoms associated with the identified disease as in Figure 5.5. Users can utilize this information for timely intervention and management of plant diseases, thus improving crop yield and overall agricultural productivity.

6. Conclusion and Future Scope

The web application represents a significant advancement in addressing agricultural challenges by leveraging cutting-edge technologies. Integrating machine learning and IoT capabilities, it offers farmers personalized crop recommendations based on real-time data, enhancing productivity and efficiency.

Incorporating ResNet for disease detection is a pivotal advancement in crop health management. By quickly identifying and diagnosing diseases through image analysis, farmers can promptly implement mitigation measures, safeguarding crop health and minimizing yield losses. This proactive approach preserves crop quality and fosters sustainable agricultural practices.

The application's emphasis on data-driven decision-making also promotes soil conservation and resource optimization. Utilizing data analytics, farmers gain insights into soil health, moisture levels, and nutrient requirements, enabling precision agriculture techniques. This targeted approach reduces unnecessary resource usage, minimizes environmental impact, and supports sustainable land management practices. Several avenues hold promise for advancing the crop recommendation application.

Firstly, expanding features to include pest detection and irrigation scheduling could enhance its utility for farmers, providing a comprehensive toolkit for crop cultivation management.

Secondly, integrating external systems, such as satellite imagery and weather forecasting, could offer more comprehensive insights into crop health and environmental conditions. This data integration would enable farmers to make more informed decisions, optimizing agricultural practices in response to changing conditions.

Lastly, collaborating with agricultural research institutions and governmental agencies could facilitate the dissemination of knowledge and adoption of advanced practices among farmers. By partnering with key stakeholders, the application could leverage collective expertise and resources to drive meaningful change, benefiting individual farmers and contributing to the resilience and sustainability of the agricultural sector.

REFERENCES

[1] X. Li and S. Li, “RANet: Lightweight Crop Disease Identification Model based on Reparameterization and Attention Mechanism,” 2023 Asia Conference on Advanced Robotics, Automation, and Control Engineering (ARACE), Chengdu, China, 2023.

[2] A. M. Kotekan, V. S. Kakaraddi and A. R. Jamakhandi, “Diseases Identification Using ConvNet In Sugarcane Crops,” 2023 International Conference on Recent Advances in Information Technology for Sustainable Development (ICRAIS), Manipal, India, 2023, pp. 266-270.

[3] G. Singh and D. s. Chabra, “Plant Disease Detection using Convolution Neural Network Approach,” 2023 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal, 2023, pp. 352-355.

[4] S. Bhansali, P. Shah, J. Shah, P. Vyas and P. Thakre, “Healthy Harvest:Crop Prediction And Disease Detection System,” 2022 IEEE 7th International conference for Convergence in Technology (I2CT), Mumbai,India, 2022, pp. 1-5.

[5] W. M. M. S. Weerasooriya, A. D. Wanigaratne, H. G. O. D. Silva, S.A. H. Hansaka, J. Perera and L. Rukgahakotuwa, “FarmCare: Locationbased Profitable Crop Recommendation System with Disease Identification”.

[6] Dr.G.Suresh, Dr.A.Senthil Kumar, Dr.S.Lekashri, Dr.R.Manikandan(2021). Efficient Crop Yield Recommendation System Using Machine Learning For Digital Farming. International Journal of Modern Agriculture, 10(1), 906 – 914.

[7] S. V. Militante, B. D. Gerardo and N. V. Dionisio, “Plant Leaf Detection and Disease Recognition using Deep Learning,” 2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), Yunlin, Taiwan, 2019, pp. 579-582.

[8] N. H. Kulkarni, G. N. Srinivasan, B. M. Sagar and N. K. Cauvery, “Improving Crop Productivity Through A Crop Recommendation System Using Ensembling Technique”.

[9] A. Arooj, M. Riaz and M. N. Akram, “Evaluation of predictive data mining algorithms in soil data classification for optimized crop recommendation”.

[10] W. Fan, C. Chong, G. Xiaoling, Y. Hua and W. Juyun, “Prediction of Crop Yield Using Big Data,” 2015 8th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 2015, pp. 255-260.

Appendix

Store_data.php

```
<?php
$servername = "localhost";
$username = "root"; // Your MySQL username
$password = ""; // Your MySQL password
$dbname = "sampledata"; // Your database name
$table = "sampledata1"; // Your table name

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// Extract sensor data from the GET request
$temperature = $_GET['temperature'];
$humidity = $_GET['humidity'];
$moisture = $_GET['moisture'];

// Insert data into the database
$sql = "INSERT INTO $table (temperature, humidity, moisture) VALUES ('$temperature',
'$humidity', '$moisture')";

if ($conn->query($sql) === TRUE) {
   echo "New record created successfully";
} else {
   echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

NodeMCU code

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266HTTPClient.h>
#include <DHT.h>

#define DHTPIN D2     // Pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11
#define SOIL_MOISTURE_PIN A0 // Pin connected to the soil moisture sensor
```

```
const char *ssid = "Pakkaki velli aaduko"; //Wi-Fi SSID
const char *password = "get.lost.."; //Wi-Fi Password
const char *serverAddress = "192.168.214.64"; //Server IP address
const int serverPort = 80; //server port number


DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  delay(10);
  dht.begin();
  connectToWiFi();
}

void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  int moisture = analogRead(SOIL_MOISTURE_PIN);

  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  sendSensorData(temperature, humidity, moisture);
  delay(30000);
}

void connectToWiFi() {
  Serial.println("Connecting to WiFi...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(10000);
    Serial.println("Connecting...");
  }
  Serial.println("Connected to WiFi");
}

void sendSensorData(float temperature, float humidity, int moisture) {
  WiFiClient client;

  if (!client.connect(serverAddress, serverPort)) {
    Serial.println("Connection failed!");
    return;
  }

  String url = "/store_data.php";
  url += "?temperature=";
  url += temperature;
  url += "&humidity=";
  url += humidity;
  url += "&moisture=";
  url += moisture;

  Serial.print("Requesting URL: ");
  Serial.println(url);

  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + serverAddress + "\r\n" +
               "Connection: close\r\n\r\n");

  Serial.println("Request sent!");

  while (client.connected()) {
    if (!client.available()) {
      delay(1);
    }
    while (client.available()) {
      Serial.print((char)client.read());
    }
  }
  Serial.println("");
  Serial.println("Closing connection");
  client.stop();
}
```

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Agriculture Toolkit</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <!-- Navigation Links -->
    <nav class="navbar">
        <ul>
            <li><a href="#home" onclick="showSection('home')">Home</a></li>
            <li><a href="#about" onclick="showSection('about')">About Us</a></li>
            <li><a href="#contact" onclick="showSection('contact')">Contact Us</a></li>
        </ul>
    </nav>
    <div class="container">
        <h1>Welcome to the Agriculture Toolkit</h1>
        <div class="options">
            <div class="option">
                <a href="{{ url_for('crop_recommendation') }}">
                    <img src="{{ url_for('static', filename='crop_recommendation.jpg') }}" alt="Crop
Recommendation">
                    <p>Crop Recommendation</p>
                </a>
            </div>
            <div class="option">
                <a href="{{ url_for('disease_detection') }}">
                    <img src="{{ url_for('static', filename='disease_detection.jpg') }}" alt="Disease
Detection">
                    <p>Disease Detection</p>
                </a>
            </div>
        </div>
    </div>

    <!-- Sections -->

    <section id="about" style="display: none;">
        <h2 style="color: #edfaf0;">About Us</h2>
        <p style="color: #edfaf0;">Farming is one of the major sectors that influences a country's
economic growth.<br>In country like India, majority of the population is dependent on
agriculture for their livelihood.<br>The system integrates two crucial features: Crop
Recommendation and Disease Prediction.<br>Allows farmers to input soil and environmental
parameters such as soil type, rainfall, temperature, humidity and recommends the best crop that
can be grown in that particular region.<br>Farmers can upload images of plant leaves
displaying signs of disease, and the system predicts the type of disease affecting the crop.</p>
    </section>

    <section id="contact" style="display: none;">
        <h2 style="color: #edfaf0;">Contact Us</h2>
        <p style="color: #edfaf0;"></p>
    </section>

    <!-- Background Image -->
    <style>
        /* Add background image to body */
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            background-image: url('static/bg_farm.jpg'); /* Background image for the entire page
*/
            background-size: cover; /* Cover the entire background */
```

```
      background-repeat: no-repeat; /* Do not repeat the background image */
      color: #fff; /* White text color */
    }


  </style>

  <!-- JavaScript to show sections -->
  <script>
    function showSection(sectionId) {
      // Hide all sections
      var sections = document.querySelectorAll('section');
      sections.forEach(function(section) {
        section.style.display = 'none';
      });

      // Show the selected section
      var selectedSection = document.getElementById(sectionId);
      selectedSection.style.display = 'block';
    }
  </script>
</body>
</html>
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Crop Recommendation</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
  <style>
    /* CSS to increase the size of input boxes */

  </style>
</head>
<body>
  <!-- Navigation Links -->
  <nav class="navbar">
    <ul>
      <li><a href="/#home" onclick="showSection('home')">Home</a></li>
      <li><a href="/#about" onclick="showSection('about')">About Us</a></li>
      <li><a href="/#contact" onclick="showSection('contact')">Contact Us</a></li>
    </ul>
  </nav>
```

```
    <div class="container">
        <h1>Crop Recommendation</h1>
        <form action="/predict" method="post">
            <label for="N">N (Nitrogen):</label><br>
            <input type="number" id="N" name="N" min="0" max="140" required><br>

            <label for="P">P (Phosphorus):</label><br>
            <input type="number" id="P" name="P" min="0" max="145" required><br>

            <label for="K">K (Potassium):</label><br>
            <input type="number" id="K" name="K" min="0" max="205" required><br>

            <label for="temperature">Temperature:</label><br>
            <input type="number" id="temperature" name="temperature" value="{{ temperature
}}" min="8.0" max="45.0" step="0.000000000000000001" required><br>

            <label for="humidity">Humidity:</label><br>
            <input type="number" id="humidity" name="humidity" value="{{ humidity }}"
min="15.0" max="100.0" step="0.000000000000000001" required><br>

            <label for="moisture">Moisture:</label><br>
            <div class="moisture-container">
                <progress id="moisture" value="{{ moisture_percentage }}"
max="100"></progress><br>
                <span class="moisture-percentage">{{ moisture_percentage }}%</span>
            </div>
            <br><br>

            <label for="ph">pH:</label><br>
            <input type="number" id="ph" name="ph" min="3.0" max="10.0" step="0.00000001"
required><br>

            <label for="rainfall">Rainfall:</label><br>
            <input type="number" id="rainfall" name="rainfall" min="20" max="300"
required><br><br>

            <input type="submit" value="Submit">
        </form>

        <button onclick="window.location.href='/'">Back</button>

    </div>

    <!-- Background Image -->
    <style>
        /* Add background image to body */
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            background-image: url('static/bg_farm.jpg'); /* Background image for the entire page
*/
            background-size: cover; /* Cover the entire background */
            background-repeat: no-repeat; /* Do not repeat the background image */
            color: #fff; /* White text color */
        }


    </style>
</body>
</html>
```

result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Result</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <!-- Navigation Links -->
    <nav class="navbar">
        <ul>
            <li><a href="/#home" onclick="showSection('home')">Home</a></li>
            <li><a href="/#about" onclick="showSection('about')">About Us</a></li>
            <li><a href="/#contact" onclick="showSection('contact')">Contact Us</a></li>
        </ul>
    </nav>
    <div class="container">
        <h1>Recommended Crop</h1>
        <div class="crop-image-container">
            <img src="{{ url_for('static', filename='crop_images/' + crop_image) }}" alt="{{ crop
}}">
        </div>
        <p class="result">The best crop to be cultivated is: {{ crop }}</p>

        <button onclick="window.location.href='/crop_recommendation'">Back</button>

    </div>
    <!-- Background Image -->
    <style>
        /* Add background image to body */
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            background-image: url('static/bg_farm.jpg'); /* Background image for the entire page
*/
            background-size: cover; /* Cover the entire background */
            background-repeat: no-repeat; /* Do not repeat the background image */
            color: #000000; /* White text color */
        }


    </style>
</body>
</html>
```

upload.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Upload Image</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">

</head>
<body>
    <!-- Navigation Links -->
    <nav class="navbar">
        <ul>
            <li><a href="/#home" onclick="showSection('home')">Home</a></li>
            <li><a href="/#about" onclick="showSection('about')">About Us</a></li>
            <li><a href="/#contact" onclick="showSection('contact')">Contact Us</a></li>
        </ul>
    </nav>
    <div class="container">
        <h1>Upload Image</h1>
        <form action="{{ url_for('disease_detection') }}" method="post"
enctype="multipart/form-data">
            <input type="file" name="file" accept=".jpg, .jpeg, .png" required>
            <button type="submit">Submit</button>
        </form>
        <button onclick="window.location.href='/'">Back</button>
    </div>
    <!-- Background Image -->
    <style>
        /* Add background image to body */
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            background-image: url('static/bg_farm.jpg'); /* Background image for the entire page
*/
            background-size: cover; /* Cover the entire background */
            background-repeat: no-repeat; /* Do not repeat the background image */
            color: #000000; /* White text color */
        }


    </style>

</body>
</html>
```

detected_disease.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Result</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">

</head>
<body>
    <!-- Navigation Links -->
    <nav class="navbar">
        <ul>
            <li><a href="/#home" onclick="showSection('home')">Home</a></li>
            <li><a href="/#about" onclick="showSection('about')">About Us</a></li>
            <li><a href="/#contact" onclick="showSection('contact')">Contact Us</a></li>
        </ul>
    </nav>
    <div class="container">
        <h1>Result</h1>
        <p>Disease Detected: {{ disease }}</p>
        <img src="{{ url_for('uploaded_file', filename=image_name) }}" alt="Uploaded Image"
style="max-width: 500px;">
        <button onclick="window.location.href='/disease_detection'">Back</button>
    </div>
    <!-- Background Image -->
    <style>
        /* Add background image to body */
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            background-image: url('static/bg_farm.jpg'); /* Background image for the entire page
*/
            background-size: cover; /* Cover the entire background */
            background-repeat: no-repeat; /* Do not repeat the background image */
            color: #000000; /* White text color */
        }
        .container {
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            min-height: 10vh; /* Set container height to full viewport height */
        }

        /* CSS for centering text */
        .disease-text {
            text-align: center;
        }

    </style>
</body>
</html>
```

style.css

```
/* Global styles */
body {
    font-family: 'Arial', sans-serif;
    margin: 0;
    padding: 0;
    background-color: #eaf7ed; /* Soft greenery background */
    background-image: url('static/bg_farm.jpg'); /* Background image related to farming */
    background-size: cover; /* Cover the entire background */
    background-repeat: no-repeat; /* Do not repeat the background image */
}

.container {
    width: 80%; /* Center the container */
    margin: 50px auto;
    padding: 30px;
    background-color: rgba(255, 255, 255, 0.9); /* Transparent white container background */
    border-radius: 10px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);

}

h1 {
  text-align: center;
  color: #2c5234; /* Dark greenery heading color */
  margin-bottom: 30px;
}

.options {
  display: flex;
  justify-content: center;
  gap: 50px;
}

.option {
  text-align: center;
}

.option img {
  width: 200px;
  height: auto;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  transition: transform 0.3s ease-in-out;
}

.option img:hover {
  transform: scale(1.05);
}

.option p {
  font-size: 18px;
  margin-top: 10px;
  color: #333;
}

/* Adjust background for the home section */
#home {
  background-color: transparent;
  padding: 0;
}
/* Style for the navigation bar */
.navbar {
  background-color: #2c5234; /* Dark greenery background */
  padding: 30px 0;
  text-align: center;
}

.navbar ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
```

```
}

.navbar ul li {
  display: inline;
  margin-right: 40px;
}

.navbar ul li a {
  color: #fff; /* White text color */
  text-decoration: none;
  font-size: 16px;
}

.navbar ul li a:hover {
  text-decoration: underline;
}
/* CSS to increase the size of input boxes */
input[type="number"] {
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
  margin-bottom: 10px;
  font-size: 16px;
}

/* CSS for the moisture percentage text */
.moisture-percentage {
  text-align: center;
  margin-top: 10px;
  font-weight: bold;
  color: #333;
}
/* CSS to change the color of labels to black */
label {
  color: #000;
  display: block;
  font-weight: bold;
}
/* CSS for the submit button */
input[type="submit"],button {
  background-color: #4CAF50;
  color: white;
  padding: 14px 20px;
  margin: 10px auto;
  border: none;
  border-radius: 4px;
  cursor: pointer;
```

```
    width: auto;
    display: block;
    text-align: center;
}
/* CSS for the result image */
.crop-image-container img {
    max-width: 100%; /* Set maximum width to 100% of container */
    height: auto; /* Maintain aspect ratio */
    display: block; /* Ensure image is displayed as block element */
    margin: 0 auto; /* Center the image horizontally */
    max-height: 200px; /* Set maximum height to desired value */
}
/* CSS for the result text paragraph */
.result {
    text-align: center; /* Align text to the center */
}
```

app.py

```
from flask import Flask, render_template, request, redirect, url_for, send_from_directory
import pickle
import numpy as np
import mysql.connector
from datetime import datetime, timedelta
import os
import cv2
import numpy as np
import tensorflow as tf
from werkzeug.utils import secure_filename
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads'
# Load pre-trained disease detection model
cnn = tf.keras.models.load_model('trained_plant_disease_model.h5')
class_names = ['Apple___Apple_scab', 'Apple___Black_rot', 'Apple___Cedar_apple_rust',
'Apple___healthy', 'Blueberry___healthy', 'Cherry_(including_sour)___Powdery_mildew',
'Cherry_(including_sour)___healthy', 'Corn_(maize)___Cercospora_leaf_spot
Gray_leaf_spot', 'Corn_(maize)___Common_rust_', 'Corn_(maize)___Northern_Leaf_Blight',
'Corn_(maize)___healthy', 'Grape___Black_rot', 'Grape___Esca_(Black_Measles)',
'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)', 'Grape___healthy',
'Orange___Haunglongbing_(Citrus_greening)', 'Peach___Bacterial_spot', 'Peach___healthy',
'Pepper,_bell___Bacterial_spot', 'Pepper,_bell___healthy', 'Potato___Early_blight',
'Potato___Late_blight', 'Potato___healthy', 'Raspberry___healthy', 'Soybean___healthy',
'Squash___Powdery_mildew', 'Strawberry___Leaf_scorch', 'Strawberry___healthy',
'Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomato___Late_blight',
'Tomato___Leaf_Mold', 'Tomato___Septoria_leaf_spot', 'Tomato___Spider_mites Two-
spotted_spider_mite', 'Tomato___Target_Spot',
'Tomato___Tomato_Yellow_Leaf_Curl_Virus', 'Tomato___Tomato_mosaic_virus',
'Tomato___healthy'] # Define your class names here
```

```
# Load crop recommendation model
model = pickle.load(open('model.pkl', 'rb'))

# Function to connect to MySQL database and fetch mean temperature, humidity, and moisture
readings over the last 7 days
def fetch_mean_sensor_data():
    try:
        connection = mysql.connector.connect(host='localhost',
                                             database='sampledata',
                                             user='root',
                                             password=")
        if connection.is_connected():
            cursor = connection.cursor()
            # Calculate start date for fetching data (7 days ago from current date)
            start_date = datetime.now() - timedelta(days=7)
            # Format start date as string
            start_date_str = start_date.strftime('%Y-%m-%d')
            # Fetch mean temperature, humidity, and moisture readings over the last 7 days
            query = f"SELECT AVG(temperature), AVG(humidity), AVG(moisture) FROM
sampledata1 WHERE timestamp >= '{start_date_str}'"
            cursor.execute(query)
            row = cursor.fetchone()
            if row:
                temperature, humidity, moisture = row
                return temperature, humidity, moisture
            else:
                return None, None, None
    except mysql.connector.Error as error:
        print("Error while connecting to MySQL", error)
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()

# Function to preprocess input and make crop recommendation
def recommend_crop(N, P, K, temperature, humidity, ph, rainfall):
    # Preprocess input
    features = np.array([[N, P, K, temperature, humidity, ph, rainfall]])
    # Make prediction
    prediction = model.predict(features)

    # Convert prediction to scalar value
    predicted_crop = prediction[0] + 1

    # Map prediction to crop name
    crop_dict = {1: "Rice", 2: "Maize", 3: "Jute", 4: "Cotton", 5: "Coconut", 6: "Papaya", 7:
"Orange",
                 8: "Apple", 9: "Muskmelon", 10: "Watermelon", 11: "Grapes", 12: "Mango", 13:
"Banana",
                14: "Pomegranate", 15: "Lentil", 16: "Blackgram", 17: "Mungbean", 18:
"Mothbeans",
                19: "Pigeonpeas", 20: "Kidneybeans", 21: "Chickpea", 22: "Coffee"}

    if predicted_crop in crop_dict:
        crop = crop_dict[predicted_crop]
        return crop, f"{predicted_crop}.jpg"
    else:
        return "Sorry, unable to recommend a proper crop for this environment"

# Function to predict disease
def predict_disease(image_path):
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (224,224))
    input_arr = np.array([image])
    predictions = cnn.predict(input_arr)
    result_index = np.argmax(predictions)
    return class_names[result_index]

from disease_details import get_additional_text_for_disease

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/crop_recommendation')
def crop_recommendation():
    # Fetch mean temperature, humidity, and moisture readings over the last 7 days from the
database
    temperature, humidity, moisture = fetch_mean_sensor_data()
    # Calculate moisture percentage
    if moisture is not None:
        moisture_percentage = 100 - ((moisture / 1024) * 100)  # Assuming 0 is wet and 1024 is
dry
    else:
        moisture_percentage = None
    return render_template('index.html', temperature=temperature, humidity=humidity,
moisture_percentage=moisture_percentage)

@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        N = float(request.form['N'])
        P = float(request.form['P'])
        K = float(request.form['K'])
        temperature = float(request.form['temperature'])
        humidity = float(request.form['humidity'])
        ph = float(request.form['ph'])
        rainfall = float(request.form['rainfall'])

        recommended_crop, crop_image = recommend_crop(N, P, K, temperature, humidity, ph,
rainfall)
        return render_template('result.html', crop=recommended_crop, crop_image=crop_image)

@app.route('/disease_detection', methods=['GET', 'POST'])
def disease_detection():
    if request.method == 'POST':
        file = request.files['file']
        if file:
            filename = secure_filename(file.filename)
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(file_path)
            # Detect disease
            disease = predict_disease(file_path)
            # Additional text based on the detected disease
            additional_text = get_additional_text_for_disease(disease)
            return redirect(url_for('detected_disease', filename=filename, disease=disease,
additional_text=additional_text))
    return render_template('upload.html')

@app.route('/detected_disease', methods=['GET', 'POST'])
def detected_disease():
    filename = request.args.get('filename')
    disease = request.args.get('disease')
    additional_text = request.args.get('additional_text')  # Ensure additional_text is retrieved
from request
    if filename:
        image_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        return render_template('detected_disease.html', image_name=filename, disease=disease,
additional_text=additional_text)
    # Redirect to the disease detection page if filename is not provided
    return redirect(url_for('disease_detection'))


@app.route('/uploads/<filename>')
def uploaded_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)

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

disease_texts.py

```
def get_additional_text_for_disease(disease):
    disease_to_text = {
```

```
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is
<u>Apple Scab</u>.</h2>
    <p><Strong></Strong></p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p></p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>  </strong><br><br>
    <strong>  </strong><br><br>
    <strong>  </strong></p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p></p>
    """


    'Apple___Apple_scab': """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is
<u>Apple Scab</u>.</h2>
    <p><strong>Apple Scab</strong> is a common and widespread fungal disease that
primarily affects apple trees, though it can also infect pear trees to a lesser extent. It is prevalent
in temperate regions worldwide and can lead to significant economic losses in apple
orchards.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>The primary symptoms of apple scab appear as olive-green to black lesions on leaves,
fruit, and occasionally on young twigs. These lesions often have a velvety or scaly appearance.
Infected leaves may become distorted or drop prematurely, reducing the tree's ability to
photosynthesize effectively. On fruit, scab lesions can cause surface cracking and deformities,
rendering them unmarketable.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Cultural Practices:</strong> Proper orchard management techniques, such
as pruning to improve air circulation, maintaining proper tree spacing, and removing infected
plant material, can help reduce the spread of apple scab.<br><br>
    <strong>Fungicide Application:</strong> Regular application of fungicides during the
growing season can help prevent apple scab infection. Fungicides containing active ingredients
such as mancozeb, captan, or sulfur are commonly used for control.<br><br>
    <strong>Resistant Varieties:</strong> Planting apple tree varieties that are resistant to
apple scab can be an effective long-term strategy for managing the disease.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Apple scab is caused by the fungus Venturia inaequalis. The fungus overwinters in
infected leaves and fruit on the ground, producing spores (ascospores) in the spring. These
spores are then carried by wind or rain to infect newly emerging leaves and fruit. The disease
thrives in cool, wet conditions, with optimal temperatures for infection ranging from 60°F to
75°F (15°C to 24°C), and it requires moisture on the leaf surface for spore germination and
infection.</p>
    """
    ,


    'Apple___Black_rot': """
```

```
<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Apple Black Rot</u>.</h2>
<p><strong>Apple black rot</strong> is a serious fungal disease that primarily affects apple trees but can also infect pear and other pome fruit trees. It is prevalent in regions with warm, humid climates and can cause significant losses in fruit production and quality.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
<p>The symptoms of apple black rot typically appear on fruit, although leaves and shoots can also be affected. Initially, small, brown to black lesions develop on the fruit, often near the stem or calyx end. These lesions gradually expand and become sunken, with a wrinkled or leathery appearance. As the disease progresses, the entire fruit may become mummified, shriveled, and covered with black fungal spores (conidia). On leaves and shoots, black rot can cause brown lesions and dieback.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
<p><strong>Sanitation: </strong>Remove and destroy any infected fruit, leaves, or branches to reduce the spread of the disease.<br><br>
<strong>Pruning: </strong>Prune trees to improve air circulation and sunlight penetration, which can help reduce humidity and create less favorable conditions for fungal growth.<br><br>
<strong>Fungicide Application: </strong>Apply fungicides preventatively during the growing season, especially during periods of warm, wet weather when the disease is most active. Fungicides containing active ingredients such as captan, thiophanate-methyl, or mancozeb are commonly used for control.<br><br>
<strong>Fruit Bagging: </strong>Bagging fruit can provide a physical barrier against infection by preventing contact with fungal spores.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
<p>Apple black rot is caused by the fungus Botryosphaeria obtusa. The fungus overwinters in infected fruit mummies, dead branches, and cankers on the tree. In spring, fungal spores (conidia) are produced on these overwintering structures and are spread by wind, rain, or insects to infect newly developing fruit and leaves. Warm, humid conditions favor disease development, with optimal temperatures for infection ranging from 75°F to 85°F (24°C to 29°C). Rain or overhead irrigation provides the moisture necessary for spore germination and infection.</p>

"""

,

'Apple___Cedar_apple_rust':
"""
<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Apple Cedar Rust</u>.</h2>
<p><Strong>Apple cedar rust</Strong> primarily affects apple and crabapple trees, as well as various species of juniper and cedar trees. It is a common fungal disease in regions where these host trees are present, causing significant damage to fruit crops and ornamental trees.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
<p>The symptoms of apple cedar rust typically appear on leaves, fruit, and sometimes on twigs of apple and crabapple trees. Early symptoms include small yellow-orange spots on the
```

upper surfaces of leaves, which gradually enlarge and develop a bright orange-red color as the disease progresses. These spots may also appear on fruit, causing deformities and reducing fruit quality. On juniper and cedar trees, apple cedar rust forms galls or swellings on the branches, which release spores during wet weather conditions.</p>

<h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>

<p><strong>Pruning: </strong>Remove and destroy any galls or infected branches on juniper and cedar trees to reduce the spread of the disease.<br><br>

<strong>Sanitation: </strong>Remove and destroy any infected leaves, fruit, or branches from apple and crabapple trees to reduce the source of fungal spores.<br><br>

<strong>Fungicide Application: </strong>Apply fungicides preventatively during the growing season, especially during periods of wet weather when spore production and infection are most likely to occur. Fungicides containing active ingredients such as myclobutanil or mancozeb are commonly used for control.</p>

<h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>

<p>Apple cedar rust requires two host plants to complete its life cycle: apple or crabapple trees (genus Malus) and various species of juniper or cedar trees (genus Juniperus). The fungal pathogen overwinters as galls on juniper and cedar trees, where it produces spores (teliospores) in the spring. These spores are then carried by wind or rain to infect apple and crabapple trees, where they cause the characteristic orange-red lesions on leaves and fruit. Secondary spores (basidiospores) produced on apple and crabapple trees can then re-infect juniper and cedar trees, completing the disease cycle.</p>

"""

,

'Apple___healthy': """<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Your crop is healthy🥳</h2>""",

'Blueberry___healthy': """<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Your crop is healthy🥳</h2>""",

'Cherry_(including_sour)___Powdery_mildew':

"""

<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Cherry Powdery Mildew</u>.</h2>

<p><Strong>Cherry powdery mildew </Strong>primarily affects cherry trees, including sweet cherry (Prunus avium) and sour cherry (Prunus cerasus) varieties. It is a common fungal disease in regions with temperate climates, causing significant damage to foliage and reducing fruit yield and quality.</p>

<h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>

<p>The symptoms of cherry powdery mildew typically appear as a white to grayish powdery fungal growth on the surfaces of leaves, shoots, flowers, and fruit. This powdery coating consists of fungal spores and mycelium. Infected leaves may become distorted, curl upwards, and eventually drop prematurely. Severe infections can lead to defoliation, weakening the tree and reducing its ability to produce fruit.</p>

<h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>

<p><strong>Cultural Practices: </strong>Practices such as proper spacing of trees, pruning to improve air circulation, and removal of infected plant debris can help reduce the

```
spread of cherry powdery mildew.<br><br>
    <strong>Fungicide Application: </strong>Apply fungicides preventatively during the growing season, especially during periods of warm, dry weather when powdery mildew is most active. Fungicides containing active ingredients such as sulfur, potassium bicarbonate, or synthetic fungicides like myclobutanil are commonly used for control.<br><br>
    <strong>Resistant Varieties: </strong>Planting cherry tree varieties that are resistant to powdery mildew can help reduce the risk of infection.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Cherry powdery mildew is caused by the fungus Podosphaera clandestina. The fungus overwinters as dormant structures (cleistothecia) on infected cherry tree debris. In spring, when conditions are favorable (moderate temperatures and high humidity), the fungus produces airborne spores (conidia) that are carried by wind to infect new cherry foliage and fruit. Unlike some other fungal diseases, cherry powdery mildew thrives in warm, dry weather conditions, with optimal temperatures for infection ranging from 68°F to 77°F (20°C to 25°C).</p>
    """
    ,
    'Cherry_(including_sour)___healthy': """<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Your crop is healthy🥳</h2>""",


    'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Corn(Maize) Cercospora leaf spot and Gray leaf spot</u>.</h2>
    <p>In Corn (Maize) <Strong>cercospora leaf spot</strong> and <strong>gray leaf spot</Strong> are two distinct fungal diseases that affect maize (corn) plants.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>Symptoms include small, dark brown to black spots with a yellow halo on maize leaves. These spots may merge, leading to large necrotic areas, which can result in premature leaf death.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Crop rotation: </strong>Avoid planting maize in the same field continuously.<br><br>
    <strong>Tillage: </strong>Deep tillage can help bury infected crop residues, reducing the source of inoculum.<br><br>
    <strong>Resistant varieties: </strong>Plant maize hybrids that are resistant or tolerant to Cercospora leaf spot.</p>
    <strong>Fungicide application: </strong>Fungicides containing active ingredients such as strobilurins or triazoles can be applied preventatively to control the disease.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Cercospora leaf spot is caused by the fungus Cercospora zeae-maydis.</p>
    <p>Gray leaf spot is caused by the fungus Cercospora zeae-maydis (same as cercospora leaf spot) or Cercospora zeina.</p>


    """
    ,
    'Corn_(maize)___Common_rust_':
```

```
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Corn Common Rust</u>.</h2>
    <p><strong>Corn Common Rust</strong> is a fungal disease caused by the pathogen <i>Puccinia sorghi</i> that affects corn (maize) plants. It is a widespread disease in regions where corn is cultivated, particularly in temperate and tropical climates.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>The characteristic symptoms of corn common rust include small, round to elongated, reddish-brown to black pustules or lesions on the leaves, stalks, and husks of corn plants. These pustules may appear powdery and can coalesce, covering large areas of the plant tissue. Severe infections can lead to stunted growth, reduced photosynthesis, and decreased yield.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Resistant Varieties:</strong> Planting corn cultivars that have genetic resistance to common rust can help mitigate the impact of the disease. Many modern corn hybrids have been bred for resistance to this fungal pathogen.<br><br>
    <strong>Crop Rotation:</strong> Practicing crop rotation can disrupt the life cycle of the fungus and reduce the build-up of inoculum in the soil, decreasing the risk of infection in subsequent corn crops.<br><br>
    <strong>Fungicide Application:</strong> In severe cases or when susceptible varieties are planted, fungicides can be applied preventively to protect corn plants from common rust infection. Fungicides containing active ingredients such as triazoles, strobilurins, or azoles are commonly used for control.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Corn common rust is caused by the fungus <i>Puccinia sorghi</i>. The disease spreads through airborne spores, which are produced in abundance within the pustules on infected plants. These spores can be carried by wind currents over long distances, facilitating the spread of the disease within and between fields. Optimal conditions for common rust development include moderate temperatures ranging from 60°F to 80°F (15°C to 27°C) and high humidity levels.</p>
    """
    ,

    'Corn_(maize)___Northern_Leaf_Blight':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Corn Northern Leaf Blight</u>.</h2>
    <p><strong>Corn Northern Leaf Blight</strong>, caused by the fungus <i>Exserohilum turcicum</i>, is a devastating foliar disease of corn (maize) plants. It is particularly prevalent in regions with humid and warm climates, posing a significant threat to corn production.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>The symptoms of corn northern leaf blight typically appear as long, elliptical lesions on the leaves of corn plants. These lesions may start as small, water-soaked spots and gradually enlarge, turning tan to brown in color with irregular margins. Severe infections can lead to widespread blighting of the leaves, reducing photosynthetic capacity and ultimately impacting yield.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Resistant Varieties:</strong> Planting corn hybrids with genetic resistance to northern leaf blight is an effective strategy for managing the disease. Many commercial corn varieties offer varying levels of resistance to this fungal pathogen.<br><br>
```

```
<strong>Crop Rotation:</strong> Rotating crops with non-host species can help reduce the build-up of inoculum in the soil and lower the risk of northern leaf blight infection in subsequent corn crops.<br><br>
<strong>Fungicide Application:</strong> When environmental conditions favor disease development or when susceptible varieties are planted, fungicides can be applied preventively to protect corn plants from northern leaf blight. Fungicides containing active ingredients such as azoxystrobin, propiconazole, or chlorothalonil are commonly used for control.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
<p>Corn northern leaf blight is caused by the fungus <i>Exserohilum turcicum</i>. The disease spreads through airborne spores, which are produced in abundance on infected corn debris and spread by wind or rain splash. Optimal conditions for disease development include moderate temperatures ranging from 68°F to 77°F (20°C to 25°C) and extended periods of leaf wetness or high humidity.</p>
"""


,
'Corn_(maize)___healthy': """<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Your crop is healthy🥳</h2>""",

'Grape___Black_rot':
"""
<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Grape Black Rot</u>.</h2>
<p><strong>Grape Black Rot</strong>, caused by the fungus <i>Guignardia bidwellii</i>, is a destructive disease affecting grapevines, particularly in regions with warm, humid climates. It can lead to significant economic losses in vineyards by reducing yield and quality of grapes.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
<p>The symptoms of grape black rot first appear as small, water-soaked spots on leaves, which gradually enlarge and turn brown or black. These lesions may become necrotic and cause defoliation, weakening the vine. On fruit, black rot manifests as circular, sunken lesions that start green and then turn brown to black as they expand. Infected grapes often shrivel and become mummified on the vine.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
<p><strong>Cultural Practices:</strong> Good vineyard management practices, such as proper pruning to improve air circulation, removal of infected plant material, and canopy management to reduce humidity around the grape clusters, can help minimize the spread of black rot.<br><br>
<strong>Fungicide Application:</strong> Regular application of fungicides, especially during periods of high disease pressure, can effectively control black rot. Fungicides containing active ingredients such as captan, mancozeb, or myclobutanil are commonly used for management.<br><br>
<strong>Fruit Mummification Removal:</strong> Prompt removal and destruction of mummified fruit from the vine and surrounding areas can reduce inoculum and prevent future infections.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
<p>Grape black rot is caused by the fungus <i>Guignardia bidwellii</i>. The fungus overwinters in infected plant debris, such as fallen leaves and mummified fruit, and produces spores (conidia) in the spring. These spores are spread by rain, wind, or insects and can infect
```

```
new growth and fruit during wet conditions. Optimal temperatures for disease development
range from 77°F to 86°F (25°C to 30°C), with prolonged periods of leaf wetness or high
humidity favoring infection.</p>
    """

    ,

    'Grape___Esca_(Black_Measles)':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is
<u>Grape Esca (Black Measles)</u>.</h2>
    <p><strong>Grape Esca</strong>, also known as <strong>Black Measles</strong>, is a
complex disease syndrome affecting grapevines worldwide. It is caused by a combination of
fungi, including <i>Phaeoacremonium spp.</i>, <i>Phaeomoniella chlamydospora</i>, and
others. Esca is a significant concern for grape growers due to its destructive nature and the
difficulty in managing its spread.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>Grape esca exhibits a range of symptoms, including foliar, vascular, and wood
symptoms. Foliar symptoms often include yellowing and browning of the leaves, as well as the
appearance of dark streaks or spots. Vascular symptoms involve discoloration and necrosis of
the vascular tissues, leading to wilting and dieback of shoots. Wood symptoms manifest as
dark streaks or lesions in the wood, commonly referred to as "tiger stripes."</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Vineyard Sanitation:</strong> Removal and destruction of infected plant
material, including pruning wood showing symptoms of esca, can help reduce inoculum and
slow the spread of the disease within the vineyard.<br><br>
    <strong>Canopy Management:</strong> Proper canopy management practices, such as
adequate pruning to improve air circulation and reduce humidity within the canopy, can help
minimize the risk of esca infection.<br><br>
    <strong>Grafting:</strong> Some grapevine rootstocks are more resistant to esca than
others. Grafting onto resistant rootstocks may help mitigate the impact of the disease in
vineyards.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Grape esca is caused by a complex of fungi, including <i>Phaeoacremonium spp.</i>,
<i>Phaeomoniella chlamydospora</i>, and others. These fungi typically enter the vine through
pruning wounds or natural openings and establish themselves within the vascular tissues, where
they spread systemically throughout the plant. Environmental stressors, such as drought or heat
stress, can exacerbate esca symptoms. Optimal conditions for disease development include
warm temperatures and periods of high humidity.</p>
    """


    ,


    'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is
<u>Grape Leaf Blight (Isariopsis Leaf Spot)</u>.</h2>
    <p><strong>Grape Leaf Blight</strong>, also known as <strong>Isariopsis Leaf
Spot</strong>, is a fungal disease caused by the pathogen <i>Isariopsis griseola</i>. It
```

```
primarily affects grapevines, leading to leaf damage and potentially reducing yield and fruit quality.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>The symptoms of grape leaf blight typically appear as small, circular to irregularly shaped spots on the leaves. These spots may start as water-soaked lesions and gradually turn brown to grayish in color with a dark border. Severe infections can cause defoliation, weaken the vine, and impact fruit ripening.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Vineyard Sanitation:</strong> Removal and destruction of infected plant material, such as diseased leaves and debris, can help reduce inoculum and limit the spread of grape leaf blight within the vineyard.<br><br>
    <strong>Fungicide Application:</strong> Regular application of fungicides, especially during periods of high disease pressure, can help manage grape leaf blight. Fungicides containing active ingredients such as copper compounds, mancozeb, or chlorothalonil are commonly used for control.<br><br>
    <strong>Canopy Management:</strong> Proper canopy management practices, including pruning to improve air circulation and reduce humidity within the canopy, can help minimize the risk of grape leaf blight infection.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Grape leaf blight is caused by the fungus <i>Isariopsis griseola</i>. The disease spreads through spores produced on infected plant material, which can be disseminated by wind, rain splash, or through human activities such as pruning. Optimal conditions for disease development include warm temperatures ranging from 68°F to 86°F (20°C to 30°C) and extended periods of leaf wetness or high humidity.</p>
    """
    ,

    'Grape___healthy': """<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Your crop is healthy🥳</h2>""",

    'Orange___Haunglongbing_(Citrus_greening)':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Orange Huanglongbing (Citrus Greening)</u>.</h2>
    <p><strong>Orange Huanglongbing</strong>, commonly known as <strong>Citrus Greening</strong>, is a devastating bacterial disease affecting citrus trees, including oranges. It is caused by the bacterium <i>Candidatus</i> Liberibacter spp. and is transmitted by the Asian citrus psyllid (<i>Diaphorina citri</i>). Citrus greening poses a severe threat to citrus production worldwide.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>The symptoms of citrus greening are diverse and include yellowing of leaves, often with asymmetrical blotchy mottling. Leaves may also exhibit a pattern of "green islands" where small patches of healthy tissue remain within the yellowed areas. Infected trees may produce small, lopsided, and bitter-tasting fruit with aborted seeds. Premature fruit drop and dieback of branches are common symptoms as the disease progresses.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Vector Control:</strong> Managing populations of the Asian citrus psyllid, the primary vector of citrus greening, is crucial for disease prevention. This can be achieved through insecticide applications, biological control methods, and cultural practices to reduce
```

```
psyllid habitat.<br><br>
    <strong>Early Detection and Removal:</strong> Prompt detection of infected trees and
removal of symptomatic plants can help prevent the spread of citrus greening within orchards.
Quarantine measures may be implemented in affected areas to limit the movement of infected
plant material.<br><br>
    <strong>Tree Nutrition and Care:</strong> Maintaining optimal tree health through
proper nutrition, irrigation, and cultural practices can help citrus trees withstand the effects of
greening and reduce their susceptibility to infection.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Citrus greening is caused by the bacterium <i>Candidatus</i> Liberibacter spp.,
which is transmitted by the Asian citrus psyllid (<i>Diaphorina citri</i>). Once infected, citrus
trees typically remain asymptomatic for several months to years before exhibiting symptoms.
The bacterium resides in the phloem tissues of the tree, disrupting nutrient transport and leading
to the characteristic symptoms of greening. Optimal conditions for disease transmission include
warm temperatures and the presence of infected psyllids.</p>
    """
    ,

    'Peach___Bacterial_spot':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is
<u>Peach Bacterial Spot</u>.</h2>
    <p><strong>Peach Bacterial Spot</strong>, caused by the bacterium <i>Xanthomonas
arboricola pv. pruni</i>, is a common and destructive disease affecting peach and other stone
fruit trees. It can lead to significant economic losses in orchards, particularly in regions with
warm and humid climates.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>The symptoms of peach bacterial spot typically appear as small, water-soaked lesions
on leaves, fruit, and stems. These lesions may turn dark brown to black and develop a raised,
rough texture as they enlarge. Infected leaves may exhibit angular necrotic spots with yellow
halos, leading to premature defoliation and reduced tree vigor. On fruit, lesions can cause
surface cracking and corky tissue, rendering them unmarketable.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Cultural Practices:</strong> Proper orchard sanitation, including removal of
infected plant material, can help reduce inoculum and slow the spread of bacterial spot within
the orchard. Pruning to improve air circulation and reduce humidity within the canopy can also
aid in disease management.<br><br>
    <strong>Copper Sprays:</strong> Application of copper-based bactericides during the
dormant season and throughout the growing season can help suppress bacterial populations and
reduce disease severity.<br><br>
    <strong>Resistant Varieties:</strong> Planting peach cultivars with genetic resistance to
bacterial spot can provide effective long-term control of the disease.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Peach bacterial spot is caused by the bacterium <i>Xanthomonas arboricola pv.
pruni</i>. The bacterium can overwinter in infected plant debris and spread to new growth
during periods of rain or overhead irrigation. Optimal conditions for disease development
include warm temperatures ranging from 75°F to 85°F (24°C to 29°C) and high humidity.
Bacterial spot can also be spread through mechanical means, such as pruning tools and
equipment.</p>
```

```
    """
    ,

    'Peach___healthy': """<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Your crop is healthy🥳</h2>""",

    'Pepper,_bell___Bacterial_spot':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Pepper Bell Bacterial Spot</u>.</h2>
    <p><strong>Pepper Bell Bacterial Spot</strong> is a bacterial disease caused by <i>Xanthomonas campestris</i> pv. <i>vesicatoria</i>, which affects pepper plants, including bell peppers. It is a significant concern for pepper growers, leading to reduced yield and fruit quality.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>The symptoms of pepper bell bacterial spot include small, water-soaked lesions on leaves, stems, and fruit. These lesions may turn dark brown to black and develop a raised, rough texture as they enlarge. Infected leaves may exhibit angular necrotic spots with yellow halos, leading to premature defoliation and reduced plant vigor. On fruit, lesions can cause surface blemishes and may lead to fruit drop.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Cultural Practices:</strong> Proper sanitation and management of crop residues can help reduce inoculum and slow the spread of bacterial spot within the pepper field. Crop rotation with non-host crops can also help break the disease cycle.<br><br>
    <strong>Pathogen-free Seed:</strong> Planting certified disease-free pepper seeds can help reduce the risk of introducing bacterial spot into the field.<br><br>
    <strong>Fungicide Application:</strong> Copper-based bactericides or other registered bactericides can be applied preventively to protect pepper plants from bacterial spot infection. Application timing and frequency should follow label instructions.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Pepper bell bacterial spot is caused by the bacterium <i>Xanthomonas campestris</i> pv. <i>vesicatoria</i>. The bacterium can survive in infected plant debris and spread to new plants through rain splash, wind-driven rain, or mechanical means. Optimal conditions for disease development include warm temperatures ranging from 75°F to 85°F (24°C to 29°C) and high humidity. Overhead irrigation or frequent rainfall can exacerbate disease spread.</p>
    """
    ,

    'Pepper,_bell___healthy': """<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Your crop is healthy🥳</h2>""",

    'Potato___Early_blight':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Potato Early Blight</u>.</h2>
    <p><strong>Potato Early Blight</strong>, caused by the fungus <i>Alternaria solani</i>, is a common foliar disease affecting potato plants. It can lead to significant yield losses and reduced tuber quality if left unmanaged.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
```

```
<p>The symptoms of potato early blight typically appear on the lower leaves of the plant and progress upward. Initial symptoms include small, dark brown spots or lesions on the leaves, which may develop concentric rings and yellow halos as they enlarge. Severe infections can cause leaf yellowing, premature defoliation, and a reduction in photosynthetic capacity.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
<p><strong>Sanitation:</strong> Removal and destruction of infected plant material, including affected leaves and debris, can help reduce inoculum and slow the spread of early blight within the potato field.<br><br>
<strong>Resistant Varieties:</strong> Planting potato cultivars with genetic resistance to early blight can provide effective control of the disease. Resistant varieties may exhibit fewer symptoms and require fewer fungicide applications.<br><br>
<strong>Fungicide Application:</strong> Regular application of fungicides, particularly during periods of high disease pressure, can help manage early blight in potato crops. Fungicides containing active ingredients such as chlorothalonil, mancozeb, or azoxystrobin are commonly used for control.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
<p>Potato early blight is caused by the fungus <i>Alternaria solani</i>. The fungus overwinters in infected plant debris and soil, producing spores (conidia) that are spread by wind, rain splash, or mechanical means. Optimal conditions for disease development include warm temperatures ranging from 75°F to 85°F (24°C to 29°C) and extended periods of leaf wetness or high humidity.</p>
"""
,

'Potato___Late_blight':
"""
<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Potato Late Blight</u>.</h2>
<p><strong>Potato Late Blight</strong>, caused by the oomycete pathogen <i>Phytophthora infestans</i>, is one of the most destructive diseases affecting potato plants worldwide. It was historically responsible for devastating famines, such as the Irish Potato Famine in the 19th century, and continues to pose a significant threat to potato production.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
<p>The symptoms of potato late blight typically appear as water-soaked lesions on leaves, which rapidly expand and turn brown to black in color. These lesions may exhibit a fuzzy or moldy appearance under humid conditions. Infected foliage may become necrotic and undergo rapid defoliation. Tubers can also be infected, leading to dark, firm lesions that can rot in storage.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
<p><strong>Resistant Varieties:</strong> Planting potato cultivars with genetic resistance to late blight can provide effective control of the disease. Resistant varieties may exhibit fewer symptoms and require fewer fungicide applications.<br><br>
<strong>Fungicide Application:</strong> Regular application of fungicides is essential for managing late blight in potato crops, especially during periods of high disease pressure. Fungicides containing active ingredients such as chlorothalonil, mancozeb, or metalaxyl are commonly used for control.<br><br>
<strong>Cultural Practices:</strong> Good crop rotation practices, removal and destruction of infected plant material, and proper irrigation management can help reduce the risk of late blight outbreaks in potato fields.</p>
```

```
<h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
<p>Potato late blight is caused by the oomycete pathogen <i>Phytophthora infestans</i>. The pathogen thrives in cool, humid conditions and can spread rapidly during periods of wet weather. Spores are produced on infected foliage and can be disseminated by wind, rain splash, or irrigation water. Optimal temperatures for disease development range from 60°F to 75°F (15°C to 24°C), with leaf wetness periods of at least 6-8 hours.</p>
"""
,

'Potato___healthy': """<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Your crop is healthy🥳</h2>""",

'Raspberry___healthy': """<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Your crop is healthy🥳</h2>""",

'Soybean___healthy': """<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Your crop is healthy🥳</h2>""",

'Squash___Powdery_mildew':
"""
<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Squash Powdery Mildew</u>.</h2>
<p><strong>Squash Powdery Mildew</strong> is a fungal disease caused by various species of the genus <i>Podosphaera</i>. It commonly affects squash plants, including zucchini, pumpkins, and other cucurbits, particularly in warm and humid climates.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
<p>The symptoms of squash powdery mildew typically appear as white to grayish powdery patches on the leaves, stems, and sometimes the fruit of affected plants. These patches may coalesce to cover large areas of the plant tissue, causing leaf distortion and reduced photosynthetic activity. Severe infections can lead to premature defoliation and reduced yield.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
<p><strong>Cultural Practices:</strong> Proper spacing between plants, adequate air circulation, and avoiding overhead irrigation can help reduce humidity levels and minimize the spread of powdery mildew in squash crops. Additionally, removing and destroying infected plant material can help prevent further disease development.<br><br>
<strong>Fungicide Application:</strong> Regular application of fungicides, particularly those containing active ingredients such as sulfur, potassium bicarbonate, or neem oil, can help manage powdery mildew in squash plants. Application timing and frequency should follow label instructions and be based on disease pressure and environmental conditions.<br><br>
<strong>Resistant Varieties:</strong> Planting squash varieties with genetic resistance to powdery mildew can provide effective long-term control of the disease.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
<p>Squash powdery mildew is caused by various species of the genus <i>Podosphaera</i>. The fungus thrives in warm temperatures and high humidity, with optimal conditions for disease development occurring between 60°F to 80°F (15°C to 27°C). Spores are produced on infected plant tissue and can be dispersed by wind to infect new plants. Overcrowded plantings and dense foliage can exacerbate disease spread by creating favorable conditions for fungal growth.</p>
```

```
"""
,

'Strawberry___Leaf_scorch':
"""
<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Strawberry Leaf Scorch</u>.</h2>
<p><strong>Strawberry Leaf Scorch</strong>, caused by the bacterium <i>Xanthomonas fragariae</i>, is a bacterial disease that affects strawberry plants. It can lead to reduced yield and quality of strawberries, particularly in warm and humid growing conditions.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
<p>The symptoms of strawberry leaf scorch typically appear as small, water-soaked lesions on the leaves, which may gradually enlarge and turn brown to black in color. These lesions may exhibit angular shapes and can coalesce to cover large areas of the leaf surface. Infected leaves may become necrotic and undergo premature defoliation, leading to reduced photosynthetic capacity and stunted growth.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
<p><strong>Cultural Practices:</strong> Proper spacing between strawberry plants, adequate air circulation, and avoiding overhead irrigation can help reduce humidity levels and minimize the spread of leaf scorch. Removing and destroying infected plant material, including affected leaves and runners, can also help prevent further disease development.<br><br>
<strong>Fungicide Application:</strong> Regular application of copper-based bactericides or other registered bactericides can help manage leaf scorch in strawberry crops. Application timing and frequency should follow label instructions and be based on disease pressure and environmental conditions.<br><br>
<strong>Resistant Varieties:</strong> Planting strawberry cultivars with genetic resistance to leaf scorch can provide effective long-term control of the disease.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
<p>Strawberry leaf scorch is caused by the bacterium <i>Xanthomonas fragariae</i>. The bacterium thrives in warm temperatures and high humidity, with optimal conditions for disease development occurring between 60°F to 80°F (15°C to 27°C). It can spread through infected plant material, splashing water, and contaminated tools or equipment. Overcrowded plantings and dense foliage can exacerbate disease spread by creating favorable conditions for bacterial growth.</p>
"""
,

'Strawberry___healthy': """<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Your crop is healthy🥳</h2>""",

'Tomato___Bacterial_spot':
"""
<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Tomato Bacterial Spot</u>.</h2>
<p><strong>Tomato Bacterial Spot</strong> is a bacterial disease caused by <i>Xanthomonas vesicatoria</i>, affecting tomato plants. It can lead to significant yield losses and reduced fruit quality if left unmanaged.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
```

```
<p>The symptoms of tomato bacterial spot typically appear as small, water-soaked lesions on the leaves, which may develop into dark, necrotic spots with a yellow halo. These spots may coalesce to form large lesions, leading to leaf yellowing, wilting, and premature defoliation. On fruit, bacterial spot lesions are typically small, dark, and sunken, often surrounded by a yellow halo.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
<p><strong>Cultural Practices:</strong> Proper sanitation, including removal and destruction of infected plant material, can help reduce inoculum and slow the spread of bacterial spot within the tomato field. Crop rotation with non-host crops can also help break the disease cycle.<br><br>
<strong>Resistant Varieties:</strong> Planting tomato cultivars with genetic resistance to bacterial spot can provide effective control of the disease. Resistant varieties may exhibit fewer symptoms and require fewer fungicide applications.<br><br>
<strong>Fungicide Application:</strong> Copper-based bactericides or other registered bactericides can be applied preventively to protect tomato plants from bacterial spot infection. Application timing and frequency should follow label instructions and be based on disease pressure and environmental conditions.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
<p>Tomato bacterial spot is caused by the bacterium <i>Xanthomonas vesicatoria</i>. The bacterium can survive in infected plant debris and soil, spreading to new plants through rain splash, wind-driven rain, or mechanical means. Optimal conditions for disease development include warm temperatures ranging from 75°F to 85°F (24°C to 29°C) and high humidity. Overhead irrigation or frequent rainfall can exacerbate disease spread.</p>
"""

,

'Tomato___Early_blight':
"""
<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Tomato Early Blight</u>.</h2>
<p><strong>Tomato Early Blight</strong>, caused by the fungus <i>Alternaria solani</i>, is a common foliar disease affecting tomato plants. It can lead to significant yield losses and reduced fruit quality if left unmanaged.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
<p>The symptoms of tomato early blight typically appear as small, dark brown lesions on the lower leaves of the plant, which may have concentric rings and yellow halos. As the disease progresses, the lesions may enlarge and coalesce, leading to defoliation and reduced plant vigor. Fruit may also become infected, developing dark, sunken lesions with concentric rings.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
<p><strong>Cultural Practices:</strong> Proper sanitation, including removal and destruction of infected plant material, can help reduce inoculum and slow the spread of early blight within the tomato field. Crop rotation with non-host crops can also help break the disease cycle.<br><br>
<strong>Resistant Varieties:</strong> Planting tomato cultivars with genetic resistance to early blight can provide effective control of the disease. Resistant varieties may exhibit fewer symptoms and require fewer fungicide applications.<br><br>
<strong>Fungicide Application:</strong> Regular application of fungicides, particularly those containing active ingredients such as chlorothalonil or mancozeb, can help manage early
```

```
blight in tomato crops. Application timing and frequency should follow label instructions and
be based on disease pressure and environmental conditions.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Tomato early blight is caused by the fungus <i>Alternaria solani</i>. The fungus can
overwinter in infected plant debris and soil, spreading to new plants through wind-driven rain,
splashing water, or mechanical means. Optimal conditions for disease development include
warm temperatures ranging from 75°F to 85°F (24°C to 29°C) and high humidity. Overhead
irrigation or frequent rainfall can exacerbate disease spread.</p>
    """

    ,

    'Tomato___Late_blight':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is
<u>Tomato Late Blight</u>.</h2>
    <p><strong>Tomato Late Blight</strong>, caused by the oomycete pathogen
<i>Phytophthora infestans</i>, is one of the most destructive diseases affecting tomato plants.
It can lead to significant yield losses and reduced fruit quality if left unmanaged.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>The symptoms of tomato late blight typically appear as water-soaked lesions on the
leaves, which rapidly expand and turn brown to black in color. These lesions may have a fuzzy
or moldy appearance under humid conditions. Infected foliage may become necrotic and
undergo rapid defoliation. Fruit can also be infected, developing dark, sunken lesions with a
firm, granular texture.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Resistant Varieties:</strong> Planting tomato cultivars with genetic
resistance to late blight can provide effective control of the disease. Resistant varieties may
exhibit fewer symptoms and require fewer fungicide applications.<br><br>
    <strong>Fungicide Application:</strong> Regular application of fungicides is essential
for managing late blight in tomato crops, especially during periods of high disease pressure.
Fungicides containing active ingredients such as chlorothalonil, mancozeb, or metalaxyl are
commonly used for control. Application timing and frequency should follow label instructions
and be based on disease pressure and environmental conditions.<br><br>
    <strong>Cultural Practices:</strong> Good crop rotation practices, removal and
destruction of infected plant material, and proper irrigation management can help reduce the
risk of late blight outbreaks in tomato fields.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Tomato late blight is caused by the oomycete pathogen <i>Phytophthora
infestans</i>. The pathogen thrives in cool, humid conditions and can spread rapidly during
periods of wet weather. Spores are produced on infected foliage and can be dispersed by wind,
rain splash, or irrigation water. Optimal temperatures for disease development range from 60°F
to 75°F (15°C to 24°C), with leaf wetness periods of at least 6-8 hours.</p>
    """

    ,

    'Tomato___Leaf_Mold':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is
<u>Tomato Leaf Mold</u>.</h2>
```

```
<p><strong>Tomato Leaf Mold</strong>, caused by the fungus <i>Fulvia fulva</i> (formerly known as <i>Cladosporium fulvum</i>), is a common fungal disease affecting tomato plants. It can lead to significant yield losses and reduced fruit quality if left unmanaged, particularly in warm and humid growing conditions.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
<p>The symptoms of tomato leaf mold typically appear as yellowish-green or pale yellow spots on the upper surface of the leaves. These spots may gradually enlarge and develop into fuzzy, grayish-white patches on the underside of the leaves, giving them a moldy appearance. Infected leaves may become distorted and curl upward, and severe infections can lead to defoliation and reduced plant vigor.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
<p><strong>Cultural Practices:</strong> Proper spacing between tomato plants, adequate air circulation, and avoiding overhead irrigation can help reduce humidity levels and minimize the spread of leaf mold within the tomato field. Removing and destroying infected plant material, including affected leaves and debris, can also help prevent further disease development.<br><br>
<strong>Fungicide Application:</strong> Regular application of fungicides, particularly those containing active ingredients such as chlorothalonil or mancozeb, can help manage leaf mold in tomato crops. Application timing and frequency should follow label instructions and be based on disease pressure and environmental conditions.<br><br>
<strong>Resistant Varieties:</strong> Planting tomato cultivars with genetic resistance to leaf mold can provide effective control of the disease. Resistant varieties may exhibit fewer symptoms and require fewer fungicide applications.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
<p>Tomato leaf mold is caused by the fungus <i>Fulvia fulva</i>. The fungus thrives in warm temperatures and high humidity, with optimal conditions for disease development occurring between 70°F to 80°F (21°C to 27°C). Spores are produced on infected plant tissue and can be disseminated by wind, water splash, or through human activities. Overhead irrigation or frequent rainfall can exacerbate disease spread.</p>
"""
,

'Tomato___Septoria_leaf_spot':
"""
<h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is <u>Tomato Septoria Leaf Spot</u>.</h2>
<p><strong>Tomato Septoria Leaf Spot</strong>, caused by the fungus <i>Septoria lycopersici</i>, is a common foliar disease affecting tomato plants. It can lead to significant yield losses and reduced fruit quality if left unmanaged.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
<p>The symptoms of tomato septoria leaf spot typically appear as small, circular lesions with dark brown centers and lighter-colored margins on the lower leaves of the plant. As the disease progresses, the lesions may increase in size and number, eventually causing leaf yellowing, wilting, and premature defoliation. Fruit can also become infected, developing dark, sunken lesions.</p>
<h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
<p><strong>Cultural Practices:</strong> Proper sanitation, including removal and destruction of infected plant material, can help reduce inoculum and slow the spread of septoria leaf spot within the tomato field. Crop rotation with non-host crops can also help break the
```

```
disease cycle.<br><br>
    <strong>Resistant Varieties:</strong> Planting tomato cultivars with genetic resistance to septoria leaf spot can provide effective control of the disease. Resistant varieties may exhibit fewer symptoms and require fewer fungicide applications.<br><br>
    <strong>Fungicide Application:</strong> Regular application of fungicides, particularly those containing active ingredients such as chlorothalonil or mancozeb, can help manage septoria leaf spot in tomato crops. Application timing and frequency should follow label instructions and be based on disease pressure and environmental conditions.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Tomato septoria leaf spot is caused by the fungus <i>Septoria lycopersici</i>. The fungus can survive in infected plant debris and soil, spreading to new plants through rain splash, wind-driven rain, or mechanical means. Optimal conditions for disease development include warm temperatures ranging from 65°F to 80°F (18°C to 27°C) and high humidity. Overhead irrigation or frequent rainfall can exacerbate disease spread.</p>
    """
    ,

    'Tomato___Spider_mites Two-spotted_spider_mite':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected pest is <u>Tomato Spider Mites</u>.</h2>
    <p><strong>Tomato Spider Mites</strong>, including the Two-Spotted Spider Mite (<i>Tetranychus urticae</i>), are common pests that can cause significant damage to tomato plants by feeding on plant tissues and sucking out cell contents. Two-spotted spider mites are particularly notorious for their ability to rapidly reproduce under warm and dry conditions, leading to widespread infestations.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>The symptoms of tomato spider mite infestation typically include stippling or yellowing of leaves caused by mite feeding, which can progress to bronzing, browning, or necrosis under severe infestations. Fine webbing may also be visible on the undersides of leaves, particularly in heavily infested areas. In severe cases, leaves may become distorted, curled, or drop prematurely, and fruit production can be reduced.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Management</h2>
    <p><strong>Cultural Practices:</strong> Maintaining proper plant hygiene by removing weeds and debris, and regular pruning to improve air circulation, can help reduce mite populations. Additionally, avoiding over-fertilization with nitrogen-rich fertilizers can help discourage mite infestations.<br><br>
    <strong>Biological Control:</strong> Predatory mites, such as <i>Phytoseiulus persimilis</i>, can be introduced to tomato plants to feed on spider mites and help control their populations.<br><br>
    <strong>Chemical Control:</strong> Insecticidal soaps, neem oil, or horticultural oils can be applied to tomato plants to smother and kill spider mites. Additionally, acaricides specifically targeting mites can be used, but care must be taken to avoid harming beneficial insects and pollinators.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Prevention</h2>
    <p>Regular monitoring of tomato plants for early signs of spider mite infestation, particularly during dry and hot weather conditions, can help prevent widespread damage. Implementing integrated pest management strategies, including cultural, biological, and chemical control methods, can effectively manage spider mite populations and minimize crop
```

```
damage.</p>
    """
    ,

    'Tomato___Target_Spot':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is
<u>Tomato Target Spot</u>.</h2>
    <p><strong>Tomato Target Spot</strong>, caused by the fungus <i>Corynespora
cassiicola</i>, is a foliar disease that affects tomato plants. While not as common as some other
tomato diseases, it can still cause significant damage under favorable conditions.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>The symptoms of tomato target spot typically include circular to irregularly shaped
lesions on the leaves, which may have a concentric ring pattern, resembling a target. These
lesions can vary in color from light brown to dark brown or black and may be surrounded by a
yellow halo. Severe infections can lead to defoliation and reduced plant vigor.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Cultural Practices:</strong> Proper sanitation, including removal and
destruction of infected plant material, can help reduce inoculum and slow the spread of target
spot within the tomato field. Crop rotation with non-host crops can also help break the disease
cycle.<br><br>
    <strong>Fungicide Application:</strong> Regular application of fungicides, particularly
those containing active ingredients such as chlorothalonil or mancozeb, can help manage target
spot in tomato crops. Application timing and frequency should follow label instructions and be
based on disease pressure and environmental conditions.<br><br>
    <strong>Resistant Varieties:</strong> Planting tomato cultivars with genetic resistance
to target spot can provide effective control of the disease. Resistant varieties may exhibit fewer
symptoms and require fewer fungicide applications.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Tomato target spot is caused by the fungus <i>Corynespora cassiicola</i>. The
fungus can survive in infected plant debris and soil, spreading to new plants through wind-
driven rain, splashing water, or mechanical means. Optimal conditions for disease development
include warm temperatures ranging from 75°F to 85°F (24°C to 29°C) and high humidity.
Overhead irrigation or frequent rainfall can exacerbate disease spread.</p>
    """
    ,

    'Tomato___Tomato_Yellow_Leaf_Curl_Virus':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is
<u>Tomato Yellow Leaf Curl Virus</u>.</h2>
    <p><strong>Tomato Yellow Leaf Curl Virus (TYLCV)</strong> is a viral disease that
affects tomato plants, causing significant damage to crops worldwide. It belongs to the genus
<i>Begomovirus</i> and is transmitted by the whitefly <i>Bemisia tabaci</i>.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>The symptoms of Tomato Yellow Leaf Curl Virus typically include yellowing and
upward curling of the leaves, along with stunted growth and reduced fruit yield. Infected plants
may also exhibit leaf thickening, puckering, and vein clearing. Fruit development may be
impaired, with small and distorted fruits being common.</p>
```

```
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Vector Control:</strong> Managing populations of the whitefly vector,
<i>Bemisia tabaci</i>, is crucial for preventing the spread of TYLCV. This can be achieved
through the use of insecticides targeting whiteflies and by implementing cultural control
practices such as removing weed hosts.<br><br>
    <strong>Resistant Varieties:</strong> Planting tomato cultivars that are resistant or
tolerant to TYLCV can help reduce the impact of the disease on crops. Resistant varieties can
withstand virus infection better and may exhibit fewer symptoms.<br><br>
    <strong>Cultural Practices:</strong> Good crop management practices, including proper
sanitation, crop rotation, and removal of infected plants, can help reduce the spread of TYLCV
within the tomato field.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Tomato Yellow Leaf Curl Virus is caused by a group of viruses belonging to the genus
<i>Begomovirus</i>. The virus is transmitted by the whitefly vector, <i>Bemisia tabaci</i>,
in a persistent, circulative manner. Infected whiteflies acquire the virus from infected plants
and transmit it to healthy plants during feeding. TYLCV can also spread through infected seed
and plant material.</p>
    """
    ,

    'Tomato___Tomato_mosaic_virus':
    """
    <h2 style="font-weight: bold; font-size: 26px; text-align: center;">Detected disease is
<u>Tomato Mosaic Virus</u>.</h2>
    <p><strong>Tomato Mosaic Virus (ToMV)</strong> is a viral disease that affects tomato
plants, causing significant damage to crops worldwide. It belongs to the genus
<i>Tobamovirus</i> and can lead to reduced yield and fruit quality.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Symptoms</h2>
    <p>The symptoms of Tomato Mosaic Virus typically include mosaic patterns of light and
dark green on the leaves, along with leaf curling, distortion, and stunting. Infected plants may
also exhibit yellowing, wilting, and necrosis of foliage. Fruit development may be impaired,
with mottling, distortion, and reduced size.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Preventive Measures</h2>
    <p><strong>Seed Treatment:</strong> Using virus-free seed and certified disease-free
transplants can help prevent the introduction of ToMV into tomato crops.<br><br>
    <strong>Vector Control:</strong> Implementing measures to control aphids, which can
transmit ToMV, can help reduce the spread of the virus. Insecticides targeting aphids and
cultural control practices such as removing weed hosts can be effective.<br><br>
    <strong>Cultural Practices:</strong> Good crop management practices, including proper
sanitation, crop rotation, and removal of infected plants, can help reduce the spread of ToMV
within the tomato field.</p>
    <h2 style="font-weight: bold; font-size: 23px; ">Causes</h2>
    <p>Tomato Mosaic Virus is caused by the <i>Tomato mosaic virus</i> belonging to the
genus <i>Tobamovirus</i>. It can be transmitted through infected seed, plant debris, and
mechanical means such as contaminated tools and hands. Aphids can also vector the virus from
infected to healthy plants during feeding.</p>
    """
    ,
    'Tomato___healthy': """<h2 style="font-weight: bold; font-size: 26px; text-align:
center;">Your crop is healthy🥳</h2>"""
    }

    # Get the corresponding text for the given disease
    return disease_to_text.get(disease, "Additional text not available")
```

crop_recommendation.ipynb

```
# Importing necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from lightgbm import LGBMClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt

# Importing dataset
dataset = pd.read_csv("Crop_recommendation.csv")

# Preprocessing
# Mapping crop labels to numeric values
crop_dict = {
    'rice': 1, 'maize': 2, 'jute': 3, 'cotton': 4, 'coconut': 5, 'papaya': 6, 'orange': 7,
    'apple': 8, 'muskmelon': 9, 'watermelon': 10, 'grapes': 11, 'mango': 12, 'banana': 13,
    'pomegranate': 14, 'lentil': 15, 'blackgram': 16, 'mungbean': 17, 'mothbeans': 18,
    'pigeonpeas': 19, 'kidneybeans': 20, 'chickpea': 21, 'coffee': 22
}
dataset['crop_num'] = dataset['label'].map(crop_dict)

# Splitting dataset into features and labels
x = dataset.drop(['crop_num', 'label'], axis=1)
y = dataset['crop_num']

# Splitting data into train and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2)

# Model Evaluation
# Initializing classifiers
classifiers = {
    'LR': LogisticRegression(), 'RF': RandomForestClassifier(random_state=0),
    'SVC': svm.SVC(), 'KNN': KNeighborsClassifier(), 'LGBM': LGBMClassifier(),
    'XGB': XGBClassifier()
}

# Evaluating each model
final_data = {'Models': [], 'ACC': []}
for name, clf in classifiers.items():
    clf.fit(x_train, y_train)
    y_pred = clf.predict(x_test)
    acc_score = accuracy_score(y_test, y_pred)
    final_data['Models'].append(name)
    final_data['ACC'].append(acc_score)

# Creating dataframe from evaluation results
final_data_df = pd.DataFrame(final_data)

# Plotting accuracy of each model
plt.figure(figsize=(4, 3))
ax = sns.barplot(x=final_data_df['Models'], y=final_data_df['ACC'])
for p in ax.patches:
    ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', xytext=(0, 10), textcoords='offset points')
plt.subplots_adjust(top=0.8)
ax.margins(y=0.2)
plt.show()

# Selecting the best model
best_model_index = final_data_df['ACC'].idxmax()
best_model = final_data_df.loc[best_model_index, 'Models']
if best_model == 'LR':
    final_model = LogisticRegression()
elif best_model == 'RF':
    final_model = RandomForestClassifier(random_state=0)
elif best_model == 'SVC':
    final_model = svm.SVC()
elif best_model == 'KNN':
    final_model = KNeighborsClassifier()
elif best_model == 'DT':
    final_model = DecisionTreeClassifier()
elif best_model == 'XGB':
    final_model = XGBClassifier()

# Training the best model
final_model.fit(x_train, y_train)

# Making predictions
y_pred = final_model.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
```

```
# Recommendation function
def recommendation(N, P, K, temperature, humidity, ph, rainfall):
    features = np.array([[N, P, K, temperature, humidity, ph, rainfall]])
    prediction = final_model.predict(features).reshape(1, -1)
    return prediction[0] + 1

# Testing recommendation function
N = 104
P = 18
K = 30
temperature = 23.603016
humidity = 60.396475
ph = 6.779833
rainfall = 140.937041
predict = recommendation(N, P, K, temperature, humidity, ph, rainfall)

# Mapping predicted numeric value to crop name
crop_dict = {1: "Rice", 2: "Maize", 3: "Jute", 4: "Cotton", 5: "Coconut", 6: "Papaya", 7:
"Orange",
             8: "Apple", 9: "Muskmelon", 10: "Watermelon", 11: "Grapes", 12: "Mango", 13:
"Banana",
             14: "Pomegranate", 15: "Lentil", 16: "Blackgram", 17: "Mungbean", 18: "Mothbeans",
             19: "Pigeonpeas", 20: "Kidneybeans", 21: "Chickpea", 22: "Coffee"}
if predict[0] in crop_dict:
    crop = crop_dict[predict[0]]
    print("{} is a recommended crop.".format(crop))
else:
    print("Sorry! We are not able to recommend a proper crop for this environment.")

# Serializing the final model
import pickle
pickle.dump(final_model, open('model.pkl', 'wb'))
```

disease_detection_model_training.ipynb

```
# Importing necessary libraries
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

# Mounting Google Drive to access dataset
from google.colab import drive
drive.mount('/content/drive')
```

```
# Extracting dataset from zip file
import zipfile
zip_ref = zipfile.ZipFile(r"/content/drive/MyDrive/plant_disease_dataset.zip", "r")
zip_ref.extractall()
zip_ref.close()

# Preprocessing: Preparing training and validation sets
training_set = tf.keras.utils.image_dataset_from_directory(
    '/content/plant disease dataset/train',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(224,224),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)

validation_set = tf.keras.utils.image_dataset_from_directory(
    '/content/plant disease dataset/valid',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(224,224),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)

# Model Building: Implementing ResNet50 architecture
# Defining identity and convolutional blocks
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization, Activation, Add, MaxPooling2D, AveragePooling2D, Flatten, Dense
from tensorflow.keras.models import Model

def identity_block(input_tensor, filters, kernel_size):
    filters1, filters2, filters3 = filters
    x = Conv2D(filters1, (1, 1))(input_tensor)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(filters2, kernel_size, padding='same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(filters3, (1, 1))(x)
    x = BatchNormalization()(x)

    x = Add()([x, input_tensor])
    x = Activation('relu')(x)
    return x

def convolutional_block(input_tensor, filters, kernel_size, strides=(2, 2)):
    filters1, filters2, filters3 = filters

    x = Conv2D(filters1, (1, 1), strides=strides)(input_tensor)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(filters2, kernel_size, padding='same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(filters3, (1, 1))(x)
    x = BatchNormalization()(x)

    shortcut = Conv2D(filters3, (1, 1), strides=strides)(input_tensor)
    shortcut = BatchNormalization()(shortcut)

    x = Add()([x, shortcut])
    x = Activation('relu')(x)
    return x

# Defining ResNet50 architecture
def ResNet50(input_shape=(224, 224, 3), num_classes=38):
    inputs = Input(shape=input_shape)

    x = Conv2D(64, (7, 7), strides=(2, 2), padding='same')(inputs)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)

    x = convolutional_block(x, filters=[64, 64, 256], kernel_size=(3, 3), strides=(1, 1))
    x = identity_block(x, filters=[64, 64, 256], kernel_size=(3, 3))
    x = identity_block(x, filters=[64, 64, 256], kernel_size=(3, 3))
    x = convolutional_block(x, filters=[128, 128, 512], kernel_size=(3, 3))
    x = identity_block(x, filters=[128, 128, 512], kernel_size=(3, 3))
    x = identity_block(x, filters=[128, 128, 512], kernel_size=(3, 3))
    x = identity_block(x, filters=[128, 128, 512], kernel_size=(3, 3))

    x = convolutional_block(x, filters=[256, 256, 1024], kernel_size=(3, 3))
    x = identity_block(x, filters=[256, 256, 1024], kernel_size=(3, 3))
    x = identity_block(x, filters=[256, 256, 1024], kernel_size=(3, 3))
    x = identity_block(x, filters=[256, 256, 1024], kernel_size=(3, 3))
    x = identity_block(x, filters=[256, 256, 1024], kernel_size=(3, 3))
    x = identity_block(x, filters=[256, 256, 1024], kernel_size=(3, 3))

    x = convolutional_block(x, filters=[512, 512, 2048], kernel_size=(3, 3))
    x = identity_block(x, filters=[512, 512, 2048], kernel_size=(3, 3))
    x = identity_block(x, filters=[512, 512, 2048], kernel_size=(3, 3))

    x = AveragePooling2D((7, 7))(x)
    x = Flatten()(x)
    x = Dense(num_classes, activation='softmax')(x)

    model = Model(inputs, x)
    return model

# Create ResNet50 model
model = ResNet50()
model.summary()

# Compiling the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Training the model
training_history = model.fit(x=training_set, validation_data=validation_set, epochs=10)

# Model Evaluation
# Training set accuracy
train_loss, train_acc = model.evaluate(training_set)
print('Training accuracy:', train_acc)

# Validation set accuracy
val_loss, val_acc = model.evaluate(validation_set)
print('Validation accuracy:', val_acc)

# Saving model
model.save('trained_plant_disease_model.keras')
model.save('trained_plant_disease_model.h5')

# Recording training history in json
import json
```

```
with open('training_hist.json', 'w') as f:
   json.dump(training_history.history, f)

# Accuracy Visualization
# Plotting training and validation accuracy
epochs = [i for i in range(1, 11)]
plt.plot(epochs, training_history.history['accuracy'], color='red', label='Training Accuracy')
plt.plot(epochs, training_history.history['val_accuracy'], color='blue', label='Validation
Accuracy')
plt.xlabel('No. of Epochs')
plt.title('Visualization of Accuracy Result')
plt.legend()
plt.show()
```