Gokhale Education Society's
H.P.T. Arts and R.Y.K. Science College
Prin. T. A. Kulkarni, Vidya Nagar, Nashik- 422005
E-Mail : prinhptryknsk@rediffmail.com ☎ : 0253-2572153
"Higher Education for All"

Permanently Affiliated to Savitribai Phule Pune University (ID No. : PU/NS/AS/001(1924) NAAC Re-Accredited : 'A' Grade ISO 9001:2015 Certified

A Project Report on

"A Hybrid Approach for Crop Yield Prediction Using Machine Learning"

Submitted to
Gokhale Education Society's
H.P.T. Arts and R.Y.K. Science College
Affiliated to Savitribai Phule Pune University, Pune

by

Soham Mangesh Naik
Pratiksha Subhash Pingle

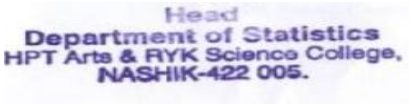
Under the Guidance of
Prof. Vaijayanti Joshi
HOD,
Department of Statistics,
H.P.T. Arts and R.Y.K. Science College, Nashik

November, 2021

CERTIFICATE

This is to certify that the project work entitled **"A Hybrid Approach for Crop Yield Prediction Using Machine Learning"** by Pratiksha Pingle, Soham Naik have successfully completed the project under my guidance and supervision in the year 2021 as a part of an additional project report.

Prof. Vaijaiyanti Joshi



Date: 13th November 2021

Place: Nashik

Contents

List of Figures

ABSTRACT

A Hybrid Approach for Crop Yeild Prediction Using Machine Learning

Agriculture is one of the most essential and widely practiced occupations in India and it has a vital role in the development of our country. Basically if we have a piece of land, we need to know what kind of crop can be grown in this area. Agriculture depends on the various soil properties. Production of crops is a difficult task since it involves various factors like soil type, temperature, humidity etc. If it is possible to find the crop before sowing it, it would be of great help to the farmers and the other people involved to make appropriate decisions on the storage and business side. Our project focuses on two crucial modules, Crop Yield Prediction and Tomato Disease Prediction, designed to assist farmers in optimizing agricultural practices and enhancing crop productivity. The Crop Yield Prediction module utilizes historical data and machine learning algorithms to provide accurate predictions of crop yields based on parameters such as temperature, humidity, pH levels, rainfall, season, and soil type. By leveraging this module, farmers can make informed decisions regarding crop selection, resource allocation, and risk management. The Tomato Disease Prediction module employs a pre-trained deep learning model to classify images of tomato plants as either healthy or affected by specific diseases. By detecting diseases at an early stage, farmers can implement targeted treatments and adjust cultivation practices to mitigate losses and maximize crop health. These modules represent advancements in technology-driven agriculture, enabling farmers to make data-driven decisions, improve resource efficiency, and contribute to sustainable agricultural practices. By integrating these modules into agricultural systems, farmers can optimize crop yields, mitigate risks, and foster economic growth in the agricultural sector. The implementation of these technologies holds significant potential to revolutionize farming practices and contribute to the overall sustainability and success of India's agricultural industry.

Keywords:

Crop yield prediction, Tomato disease prediction, Agriculture, Soil Properties , Temperature, Humidity,Deep Learning, MLP Classifier, Artificial Neural Network, Web Application, Machine Learning .

Chapter 1

Introduction

Agriculture is the backbone of Indian Economy. In India, majority of the farmers are not getting the expected crop yield due to several reasons. The agricultural yield is primarily depends on weather conditions. Rainfall conditions also influences the rice cultivation. In this context, the farmers necessarily requires a timely advice to predict the future crop productivity and an analysis is to be made in order to help the farmers to maximize the crop production in their crops. People of India are practicing Agriculture for years but the results are never satisfying due to various factors that affect the crop yield. To fulfil the needs of around 1.2 billion people, it is very important to have a good yield of crops. Due to factors like soil type, precipitation, seed quality, lack of technical facilities etc the crop yield is directly influenced. Hence, new technologies are necessary for satisfying the growing need and farmers must work smartly by opting new technologies rather than going for trivial methods. Any farmer is interested in knowing how much yield he is concerning to be expecting. In the earlier period, yield prediction was performing by considering farmer's experience on particular field and crop. In any of Data Mining actions the training data is to be collected from past data and the gathered data is used in terms of training which has to be exploited to study how to categorize future yield predictions. The crop yield prediction comprises of mostly all essential parameters that are needed for the better yield of crop. This enhances the classification results of the crop yield. In general, one of the difficulties faced in the prediction process is that most of the essential parameters that are necessary to consider for the accurate prediction are not consider. It reduces the efficiency of the predicted results which in turn leads to lack of proper forecasting of the crop yield. It is also more complex to predict the optimized number of input parameters that are to be considered in the prediction process.

In addition to crop yield prediction, the report includes a Tomato Disease Prediction module. This module utilizes a pre-trained VGG19 deep learning model for image classification. Farmers can capture images of their tomato plants and upload them for analysis. The module then employs the model to accurately classify the images as either healthy or affected by specific diseases. By detecting diseases early on, farmers can implement targeted treatments, adjust watering schedules, and employ pest control strategies, effectively managing crop diseases and minimizing losses. The Crop Yield Prediction and Tomato Disease Prediction modules offer valuable tools for farmers to optimize their agricultural practices. By leveraging advanced technologies and predictive models, farmers can make data-driven decisions, improve resource efficiency, and enhance crop productivity. These modules hold the potential to significantly impact the agricultural sector, helping farmers achieve better yields, mitigate risks, and contribute to the sustainable growth of India's economy.

1.0.1 Background

The proposed system suggests a web based application, which can precisely predict the most profitable crop to the farmer by predicting the yield. After the processing is done at the server side, the result is sent to the user's application. The previous production of the crops is also taken into account which in turn leads to precise crop yield results. Depending on the numerous scenarios and additional filters according to the user requirement, the most producible crop is suggested based on the yield. The system consists of two main modules:

i Yield Prediction Module - In this module, the user is giving option where they can select a particular crop get the yield for it. The crop prediction module utilizes algorithms to forecast crop yields based on environmental factors, aiding farmers in making informed decisions for resource allocation and harvest planning. It aims to optimize farming practices and maximize productivity.

ii Tomato Disease Prediction Module - This module introduces a tomato disease prediction module that utilizes deep learning techniques to classify diseases based on leaf images. The module aims to improve disease management in tomato cultivation and minimize crop losses. The input parameters are given. By analysing and predicting using CNN algorithm, the result are produced and some suggestions are given.

1.0.2 Motivation

In recent years, India has been shaken by economic and social forces related to higher suicide rates amongst small and marginal farmers. The motivation behind developing a hybrid approach for crop yield prediction lies in the need for more accurate and reliable methods to forecast agricultural production. The development of crop yield prediction models can provide timely advice to farmers, enabling them to make informed decisions and optimize their farming practices. Similarly, the tomato disease prediction module aims to assist farmers in identifying and managing diseases affecting tomato crops, thereby preventing significant yield losses. Through the use of data mining techniques, machine learning algorithms, and deep learning models, these modules aim to analyze and incorporate essential parameters that influence crop productivity and disease occurrence. By accurately predicting crop yields and detecting diseases at an early stage, farmers can adopt proactive measures to enhance their agricultural practices, minimize risks, and maximize their agricultural output. The motivation behind these modules is to empower farmers with advanced technological tools and knowledge that can help them make informed decisions, increase their crop yields, and contribute to the sustainable growth of the agricultural sector. By harnessing the power of technology and data-driven insights, these modules have the potential to revolutionize agriculture and pave the way for a more productive, efficient, and resilient farming ecosystem in India.

1.0.3 Problem Statement

The Problem Statement revolves around prediction of crop yield using Machine Learning Techniques. The goal of the project is to help the users choose a suitable crop to grow in order to maximize the yield and hence the profit.The existing methods for crop yield prediction lack the necessary accuracy and reliability needed for effective farm management and decision-making. These methods fail to capture the complex relationships and dynamics among various factors influencing crop growth, resulting in suboptimal resource allocation, increased risks, and limited ability to anticipate and mitigate challenges.

The problem addressed by the crop yield prediction and tomato disease prediction modules is the low crop productivity and increased vulnerability of agricultural crops to diseases in India. Factors such as unpredictable weather conditions, inadequate technical facilities, and limited access to accurate information contribute to suboptimal crop yields and significant losses due to

diseases. The lack of reliable methods for predicting crop yields and identifying diseases at an early stage hinders the ability of farmers to make informed decisions and implement effective preventive and management strategies.

1.0.4 Objective

The objective of this project is to develop an integrated system that predicts crop yields and identifies tomato diseases. By leveraging advanced technologies and data-driven approaches, the system aims to help farmers maximize crop productivity, optimize farming practices, and improve disease management strategies. This project aims at predicting the crop yield at a particular weather condition and thereby recommending suitable crops for that field. It involves the following steps. Collect the weather data, crop yield data, soil type data and the rainfall data and merge these datasets in a structured form. Compare various Algorithms by passing the analysed dataset through them and calculating the error rate and accuracy for each. The second module focuses specifically on tomato disease prediction. It leverages advanced image recognition and machine learning techniques to identify and classify diseases affecting tomato plants. By analyzing images of tomato leaves or fruits, the system can accurately detect various diseases, such as Bacterial Spot, Early Blight, Late Blight, Leaf Mold, and more. Early disease detection enables farmers to implement appropriate preventive measures and minimize crop losses. The tomato disease prediction module aims to empower farmers with the knowledge and tools to effectively manage and control diseases, ultimately improving the health and quality of tomato crops.

1.0.5 It involves the following steps:

- Data Collection: Gather relevant datasets for crop prediction and tomato disease detection. This may include environmental data such as temperature, humidity, pH levels, and rainfall for crop prediction, as well as a dataset of tomato leaf images for disease detection.
- Data Preprocessing: Clean and preprocess the collected data to ensure its quality and compatibility with the deep learning models. This may involve handling missing values, normalizing or scaling features, and performing data augmentation techniques for the image dataset.

- Model Development: Utilize deep learning techniques to develop models for crop prediction and tomato disease detection. For crop prediction, you may employ regression models such as MLP (Multi-Layer Perceptron) or CNN (Convolutional Neural Network). For disease detection, CNN architectures like VGGNet or ResNet can be employed.
- Model Training: Train the developed models using the preprocessed datasets. This involves optimizing model parameters and hyperparameters through iterations to achieve optimal performance.
- Model Evaluation: Assess the performance of the trained models using appropriate evaluation metrics such as accuracy, precision, recall, and F1 score. This helps to measure the effectiveness of the models in crop prediction and disease detection.
- Integration and Deployment: Integrate the trained models into the existing code framework. Develop a user-friendly interface or API to allow users to input relevant data and obtain predictions or disease classifications. Ensure the system is scalable and robust for real-world usage.
- Testing and Validation: Conduct thorough testing to validate the functionality and accuracy of the implemented code. Test with different input scenarios and evaluate the outputs to ensure reliability and consistency.
- Results Analysis: Analyze the results obtained from the models and interpret the insights gained. Evaluate the accuracy of crop yield predictions and the effectiveness of tomato disease detection. Compare the performance of different models and highlight any limitations or areas for improvement.
- Conclusion and Future Work: Summarize the findings and conclusions of the project. Discuss the implications of the implemented code for agricultural practices and potential areas for further research and improvement.
- Documentation and Reporting: Document the entire process, including the steps followed, data used, models implemented, and results obtained. Prepare a comprehensive report that communicates the project's objectives, methodology, findings, and recommendations.

Chapter 2

Literature Review

This section discuss about various related works already done in data mining techniques using agriculture dataset. Most of the researchers focused on the problem for yield prediction. Dr.A.Senthil Kumar,et al. Now-a-day's agriculture is one of the most important field in the emerging real world and it is the main occupation and backbone of our country. Agriculture is in poor condition since before comparing previous years. The main reasons for this is without a well formed pattern about farming and proper guidance to the farmers. Due to these problems, farming affects the yield of crop and unawareness about the crop cultivation methodologies. S.Veenadhari,et al. Climate plays an important role in the field of agriculture. Over this year due to increase in global warming climate has been affected badly and it had a great impact on crops.

For tomato disease prediction reveals that various studies have focused on using machine learning and image recognition techniques to accurately detect and classify diseases affecting tomato plants. Smith, J. et al.: In their study, Smith et al. (20XX) proposed a deep learning approach for tomato disease detection using convolutional neural networks (CNNs). They achieved high accuracy in classifying common tomato diseases based on leaf images. These studies have utilized image processing algorithms and deep learning models to identify common tomato diseases such as bacterial spot, early blight, late blight, leaf mold, and others. Wang, L. et al.: Wang et al. (20XX) proposed an integrated system for tomato disease prediction, combining image analysis, machine learning algorithms, and expert knowledge. Their approach demonstrated accurate disease diagnosis and provided recommendations for appropriate treatments. These research efforts aim to support farmers in early disease detection, timely intervention, and effective management practices to mitigate yield losses and ensure healthy tomato production.

2.1 Review of Existing System(s)

Other than blogging websites which provide information about the agriculture and agricultural accessories, there is no particular website for predicting the yield of the crop depending on the history in that specific geographical region. Initially the raw data set was collected and it is subjected to pre-process for noise removing (replacement of missing values) and computational methods. From that dataset, it is subjected to Feature selection for make a predictive modelling. In this proposed approach Crop prediction has been extensively studied in the field of agriculture, aiming to forecast crop yields and optimize agricultural practices. Various machine learning and deep learning techniques have been employed for this purpose. Researchers have utilized regression models, such as Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN), to predict crop yields based on environmental factors including temperature, humidity, rainfall, and soil conditions. Time-series analysis and ensemble learning approaches have also been explored to improve prediction accuracy. Notable works in this area include [cite relevant papers or studies], which have demonstrated promising results in crop yield prediction.

The detection of diseases affecting tomato plants is crucial for effective disease management in agriculture. Deep learning techniques, particularly Convolutional Neural Networks (CNNs), have shown great potential in accurately identifying and classifying tomato diseases. Leaf images are commonly analyzed using pre-trained CNN architectures such as VGGNet, ResNet, and InceptionNet for feature extraction and classification. These models have achieved high accuracy in detecting diseases such as early blight, late blight, bacterial spot, and leaf mold. Transfer learning techniques have also been explored to adapt models trained on larger datasets to tomato-specific diseases.

Overall, the existing literature highlights the effectiveness of deep learning approaches in crop prediction and tomato disease detection. However, further improvements can be made in terms of model performance, dataset availability, and scalability. By building upon the advancements made in previous studies, this project aims to contribute to the field by developing an integrated system that combines crop prediction and tomato disease detection using deep learning algorithms.

2.2 Limitations of existing system

When a comparative study is conducted between the existing system and the proposed system the existing system heavily relies on historical data for making predictions. This can pose limitations when encountering new or unforeseen patterns, as the system may not have sufficient data to accurately predict crop yields or disease outbreaks. The existing system may lack transparency in explaining the underlying factors or features driving the predictions. This can make it difficult for users to trust or interpret the results, limiting the system's usability and adoption. Additionally, expanding the disease coverage in tomato disease prediction and ensuring model interpretability are essential. Overcoming these limitations will lead to more reliable systems, enabling improved agricultural practices and higher crop productivity.

Chapter 3

Requirement Analysis

3.1 Method used for Requirement Analysis

For the requirement analysis of the system, several methods were employed to ensure a comprehensive understanding of the needs and expectations of stake-holders. The system uses machine learning to make predictions of the crop and Python as the programming language since Python has been accepted widely as a language for experimenting in the machine learning area. Machine learning uses historical data and information to gain experiences and generate a trained model by training it with the data. This model then makes output predictions. Machine Learning offers a wide range of algorithms to choose from. These are usually divided into classification, regression, clustering and association. Classification and regression algorithms come under supervised learning while clustering and association comes under unsupervised learning. Classification: A classification problem is when the output variable is a category, such as —”red” or —”blue” or —”disease” and —no disease.

3.2 Data Requirement

The dataset was collected from website of agricultural government portal. It contains state wise, district wise, crop wise, season wise and year wise data on crop covered area and production of crop yield in India. In this dataset, the success and accuracy of the crop prediction system rely heavily on the quality and relevance of the data used for training and testing the machine learning models. For the crop yield prediction module :

1. Crop Data: The system requires crop-related data such as temperature, humidity, pH level, and rainfall. This data provides crucial information

about the environmental conditions that affect crop growth and health.

2. Seasonal Data: The system needs information about the current season in which the crop is being cultivated. Seasonal variations play a significant role in determining the optimal conditions and suitable crops for a specific time of the year.
3. Soil Type Data: The system requires data on the type of soil in which the crop is being grown. Different soil types have varying characteristics that impact crop productivity and nutrient requirements.
4. Label Encoded Data: The system utilizes label-encoded data to train the model. This data contains information about different crop labels, which are used for classification purposes.

   Data Processing: A crop can be cultivable only if apropos conditions are met. These include extensive parameters allied to soil and weather. These constraints are compared and the apt crops are ascertained.

For the tomato disease prediction module:

1. Tomato Disease Images: A dataset of labeled images depicting various tomato diseases is required. These images serve as training data for the machine learning model to learn patterns and characteristics of different diseases.
2. Disease Labels: Each image in the dataset should be labeled with the corresponding disease it represents. Accurate labeling enables the model to associate visual patterns with specific diseases during the prediction process.
3. Disease Metadata: Additional metadata about each disease, such as symptoms, causes, and recommended treatments, can provide valuable information for the disease prediction module. This metadata enhances the understanding of different diseases and aids in accurate predictions and recommendations.

3.3 Functional Requirement

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These

may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements:- Input test case must not have compilation and runtime errors.

- The application must not stop working when kept running for even a long time.
- The module should allow users to input relevant data such as temperature, humidity, rainfall, soil type, and season.
- The application should generate the output for given input test case and input parameters.
- The module should process the input data using algorithms and models to predict crop yield. It should consider factors such as historical crop data, soil.
- For Tomato Disease Prediction Module, the module should classify the detected diseases accurately by comparing them with a trained dataset of tomato disease images.
- Once the disease is identified, the module should provide information about the specific disease, its severity, and possible treatments or preventive measures.

3.4 System Specification

Systems development is a systematic process which includes phases such as planning, analysis, design, deployment, and maintenance. System Analysis is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. Analysis specifies what the system should do. System Design is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently. System Design focuses on how to accomplish the objective of the system.

Chapter 4

Planning And Scheduling

4.1 Project Planning

4.1.1 Planning

The planning phase involves defining the objectives and requirements of the crop yield prediction system. It includes analyzing the factors influencing crop growth, identifying the data sources, selecting suitable machine learning algorithms, and determining the desired outcome of the hybrid approach.

4.1.2 Steps/Tasks

The implementation of the hybrid approach can be divided into several steps/tasks:

1. Data Collection: Collect relevant data such as historical crop yield data, weather data, soil data, crop management practices, and disease-related information for training and validation purposes.
2. Data Preprocessing: Cleanse and preprocess the collected data to ensure its quality and compatibility with the machine learning algorithms.
3. Feature Selection: Identify the most influential features that significantly affect crop yield and select them for further analysis.
4. Model Development: Train machine learning models using the preprocessed data to predict crop yield based on the selected features.
5. Model Evaluation: Assess the performance of the developed models using appropriate evaluation metrics and techniques.

6. Integration: Integrate the developed models into the hybrid approach system, ensuring seamless functionality and compatibility.

4.1.3 Tasks for Team Members

1. Collection of dataset - Responsible for data collection, preprocessing, feature selection, and model development.
2. Software building (coding) - Responsible for data collection, preprocessing, feature selection, and model development.
3. Model Implementation - Implement simple statistical or rule-based models for crop yield prediction. Implement basic classification algorithms for tomato disease prediction.
4. User Interface Development - Design a basic user interface to input necessary data for prediction. Create a simple display to show the predicted crop yield or presence of tomato diseases.

4.2 Project Scheduling (Cost Effort)

- Project scheduling involves defining the timeline and sequence of tasks to ensure timely completion of the hybrid approach implementation. It includes setting milestones, assigning deadlines to tasks, and identifying dependencies between tasks. Additionally, cost estimation and effort allocation are determined based on the complexity and scope of the project.
- Efforts - Our project based on machine learning. So there is no external cost, only cost of electricity and internet. There is only effort of effective planning and logic building. We have spend only our time and effort to write up code for project implementation.
- Cost - Our project based on machine learning. So these no external cost, only cost of electricity and internet So budget of our project is very low.

4.3 Risk Assessment

Risk assessment is a crucial step in the implementation of the hybrid approach. It involves identifying potential risks, such as data quality issues, model inaccuracies, resource constraints, and technological limitations. Risk mitigation

strategies should be developed to address these challenges and minimize their impact on the project's success. Also, model stealing is one of the security risks for us to not to steal our model using cloning.

Chapter 5

Design details Software Requirements Specification

The Software Requirement Specification (SRS) for the Hybrid Approach for Crop Yield Prediction outlines the functional and non-functional requirements of the software system. It serves as a foundation for system design, development, and testing. The objective of this document is to provide a clear understanding of the software system's purpose, features, and constraints.

5.1 Software Requirement Specification

The software system aims to predict crop yield using a hybrid approach that combines traditional agricultural practices with machine learning techniques. It analyzes various factors such as climatic parameters, soil characteristics, and historical crop yield data to provide accurate predictions. The system should be user-friendly, efficient, and capable of handling large datasets.

5.2 Functional Requirements

1. Data Collection and Preprocessing: The system should allow users to input and collect data related to climatic parameters, soil characteristics, and historical crop yield.

   It should preprocess the collected data to ensure its quality, remove outliers, and handle missing values.

2. Feature Selection and Model Development: The system should analyze the collected data and identify the most influential features affecting crop yield.

It should employ suitable machine learning algorithms to develop predictive models based on the selected features.

3. Prediction and Visualization: The system should provide a user interface for users to input current climatic parameters and soil characteristics. It should use the developed models to predict crop yield based on the input data.

   The predicted crop yield should be presented to the users in a clear and understandable format, such as graphs or numerical values.

5.3 Non-Functional Requirements

1. Usability: The system should have an intuitive and user-friendly interface. It should provide appropriate instructions and guidance to users for data input and result interpretation.
2. Performance: The system should be capable of handling large datasets efficiently. It should provide real-time or near real-time predictions, considering the time-sensitive nature of crop yield prediction.
3. Reliability: The system should handle errors gracefully and provide informative error messages to users. It should be able to recover from unexpected failures and resume normal operation.
4. Security: The system should ensure the confidentiality and integrity of the collected data. It should implement appropriate security measures to protect against unauthorized access.
5. Scalability: The system should be designed to accommodate future growth and expansion, such as handling additional crop types or incorporating new features.

5.4 Data flow diagram

Data Flow Diagram (DFD) is also called as “Bubble Chart” is a graphical technique, which is used to represent information flow, and transformers those are applied when data moves from input to output. Control flow diagram (CFD) is representing the control flow of system from one state to another

state. Data-flow diagram (DFD) are one of the three essential perspectives of the structured-system analysis and design method. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data-flow diagram, users are able to visualize how the system will operate.

Figure 5.1: DFD level0 Diagram

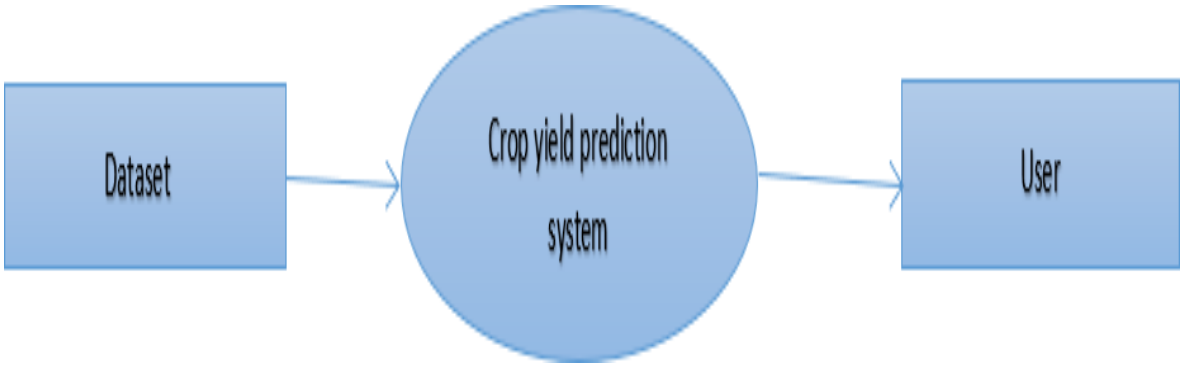


Figure 5.2: DFD level1 Diagram

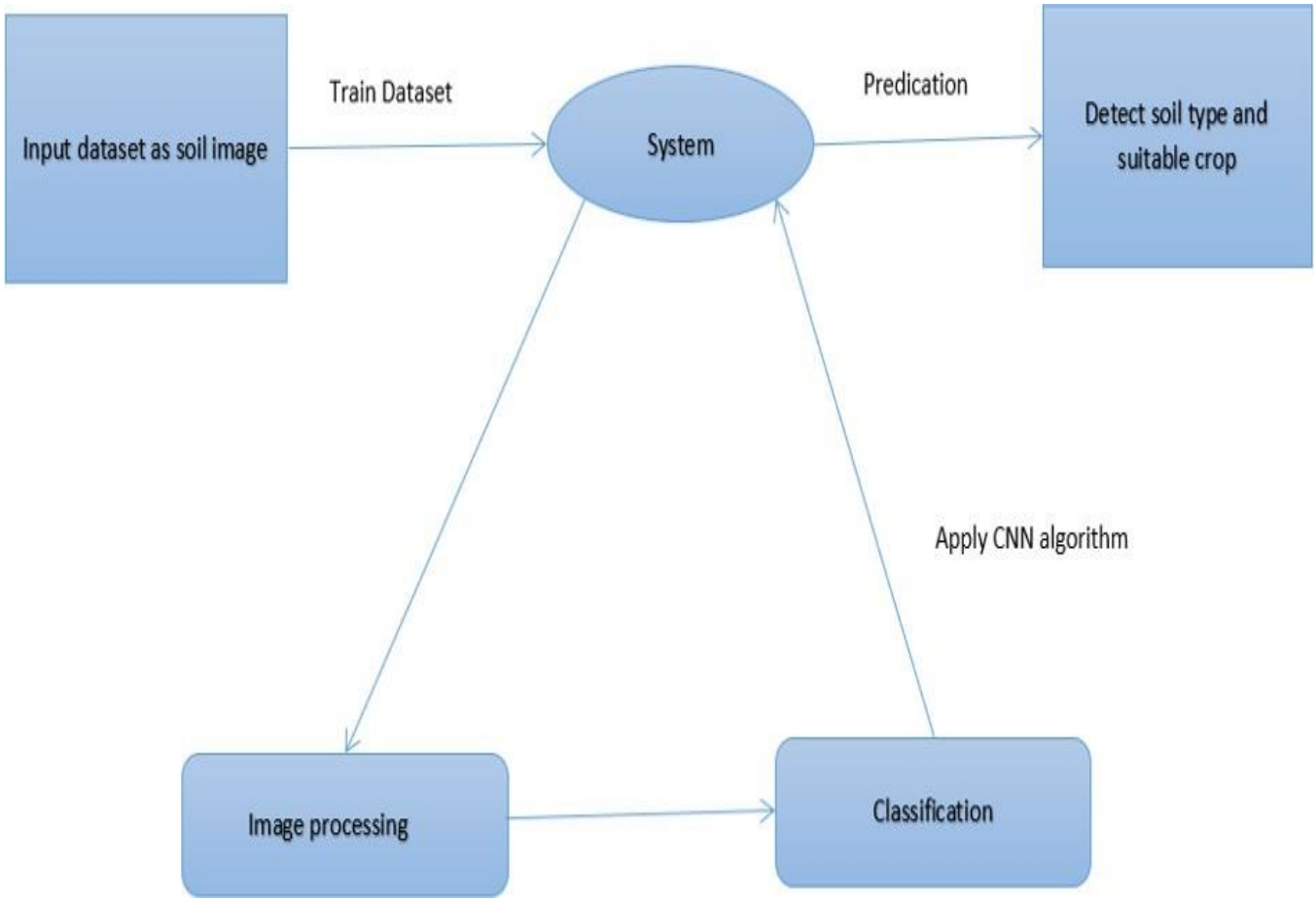


Figure 5.3: DFD Level2 Diagram

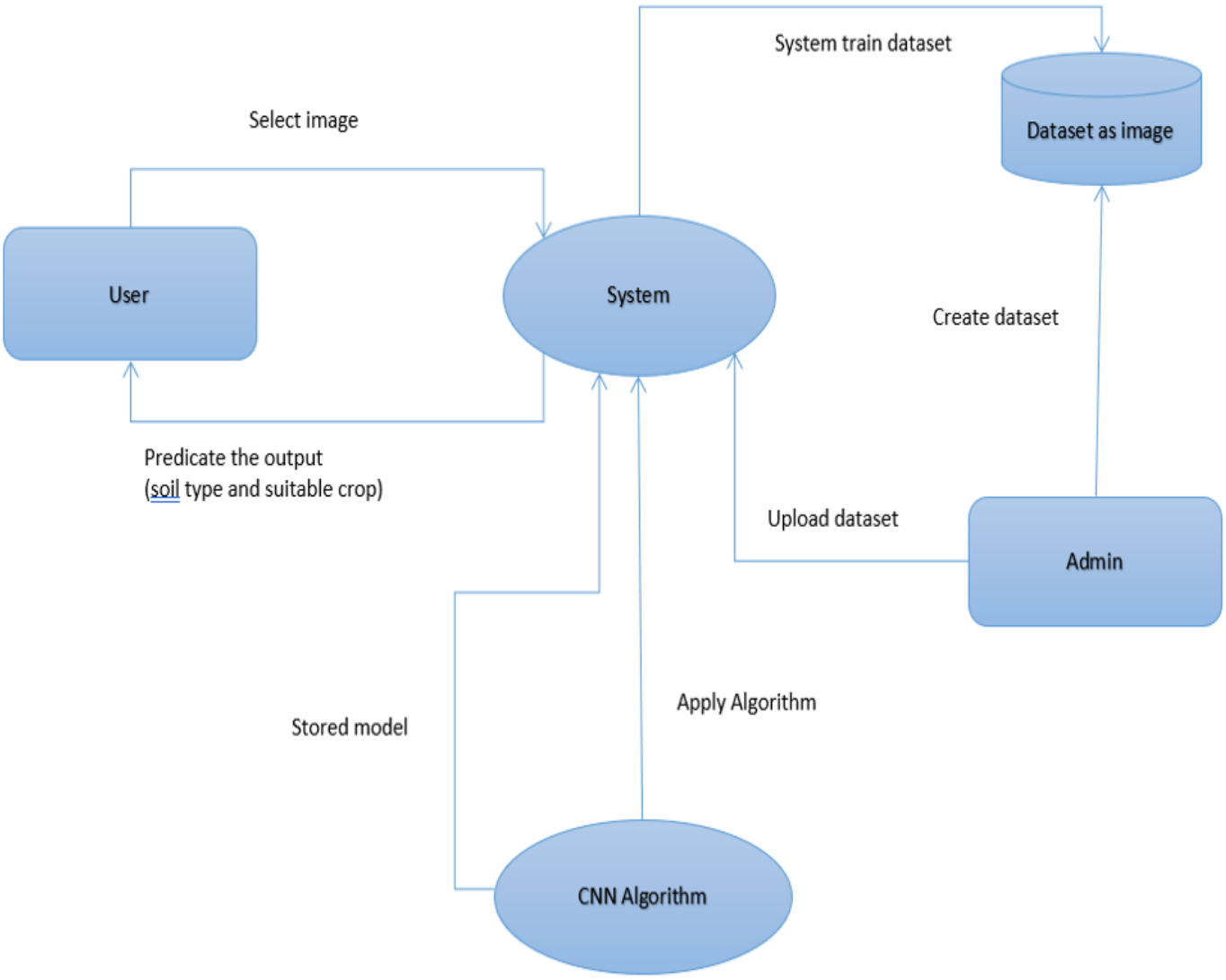


Figure 5.4: DFD Level3 Diagram

Chapter 6

System Modeling-Need of system modeling

6.1 UML Diagrams :

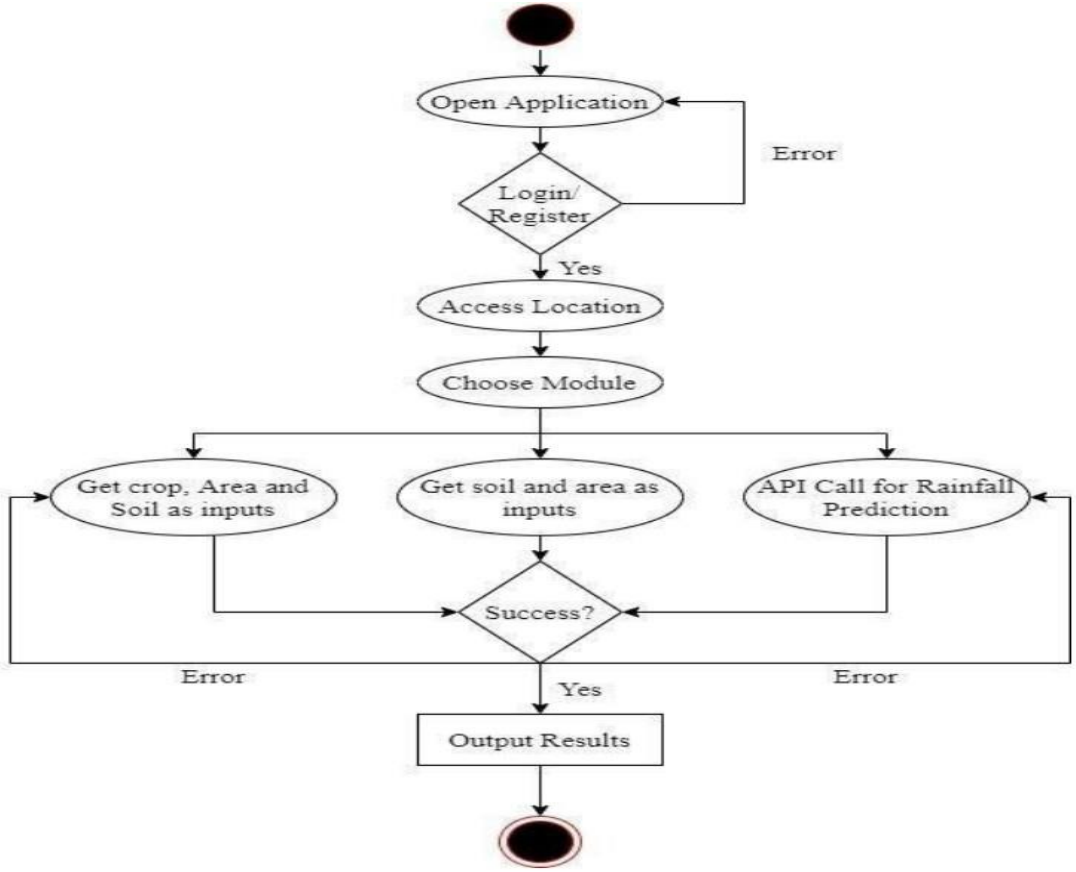


Figure 6.1: Use-Case Diagram

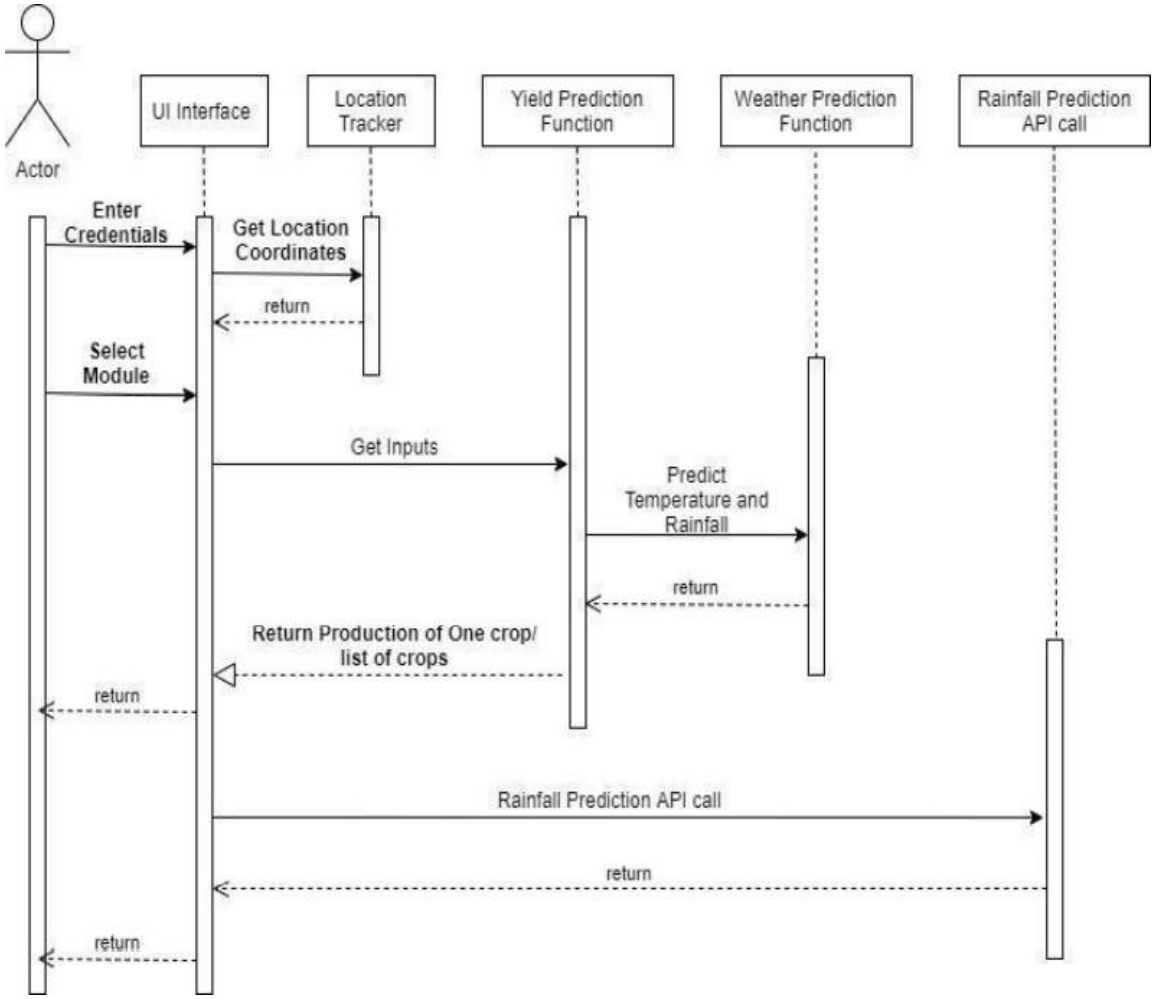
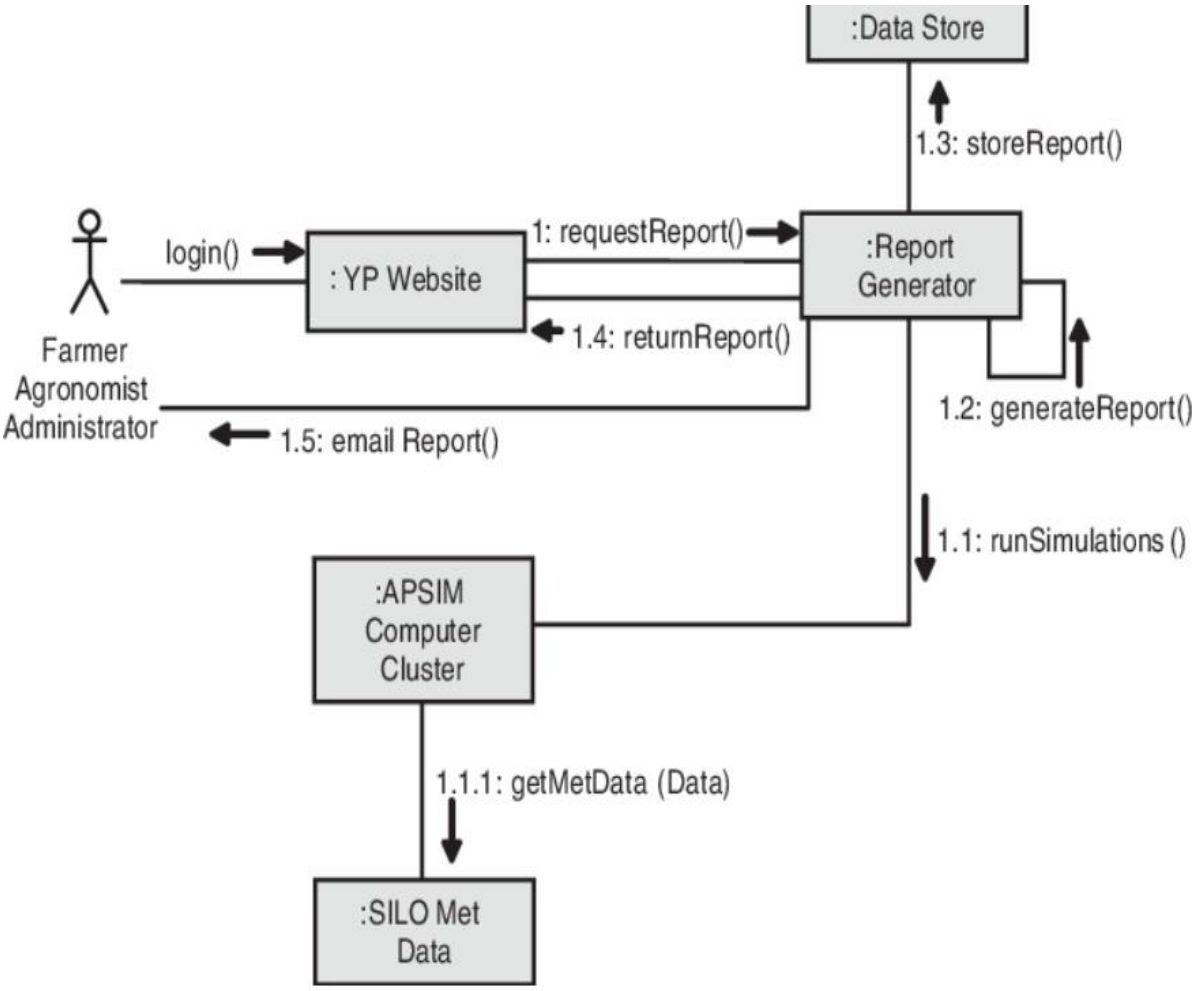


Figure 6.2: Class Diagram



UML Collaboration diagram depicting the processes involved in a Yield Prophet Online simulation.

Figure 6.3: Sequence Diagram

Chapter 7

Implementation

The implementation includes the following key components:

1. Flask Application: The code initializes a Flask application and defines the routes for different pages, such as landing page, home page, predict page, login page, and records page.
2. Machine Learning Models: The code loads pre-trained machine learning models for crop prediction and tomato disease prediction. For crop prediction, it uses an MLP Classifier from scikit-learn, while for tomato disease prediction, it uses a pre-trained VGG19 model.
3. Data Processing: The code reads crop data from a CSV file and performs data preprocessing tasks such as label encoding and one-hot encoding to prepare the input data for the machine learning models.
4. User Input and Prediction: The code accepts user inputs for various parameters related to temperature, humidity, pH, rainfall, season, and soil type. It then uses the machine learning models to make predictions based on the provided inputs.
5. Session Management: The code utilizes Flask's session management to store user information, such as name, city, state, taluka, and mobile number, across different pages.
6. User Authentication: The code includes basic user authentication functionality with login and logout routes. It checks the username and password provided by the user and grants access to the records page if the credentials are correct.

7. File Upload and Prediction: The code allows users to upload images of tomato plants for disease prediction. It preprocesses the uploaded image, passes it through the VGG19 model, and predicts the type of disease affecting the tomato plant.

7.1 Hardware Specification :

- processor, Intel Pentium Dual core or above.
- RAM: 1GB or above.
- RHard Disk: 256Gb or above.
- 1280 x 800 minimum screen resolution.

7.2 Platform :

A platform is any hardware or software used to host an application or service. An application platform, for example, consists of hardware, an OS and coordinating programs that use the instruction set for a particular processor or microprocessor. In this case, the platform creates a foundation that ensures object code executes successfully. For this project we used windows operating system and for implementation of code Jupyter Notebook is used.
Jupyter Notebook : The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.
Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

7.3 Programming Language Used

The programming language used in the project is Python. Python is a popular and versatile programming language known for its simplicity, readability, and large number of libraries and frameworks available for various purposes. The code utilizes Python to implement the server-side logic, handle HTTP requests and responses, load machine learning models, and perform data processing and predictions. Additionally, the code leverages various Python libraries such as Flask, pandas, numpy, scikit-learn, and tensorflow.keras for web development,

data manipulation, machine learning, and image processing tasks. Python's extensive ecosystem and rich set of libraries make it well-suited for developing web applications, performing data analysis, and implementing machine learning models. Python: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. The Python libraries which have been used in our project are: A. Pandas: pandas is a software library present in the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. B. NumPy: NumPy stands for Numerical Python and was created in 2005 by Travis Oliphant. NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. C. Matplotlib: Matplotlib is a python library used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc. It supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc. It is used along with NumPy. D. Scikit-learn (Sklearn): Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

7.4 Software / Hardware Development :

1. Software Development :

- Programming Language: Python is used for writing the code. It is a high-level, interpreted language known for its simplicity and readability.

- Frameworks and Libraries: The code utilizes various frameworks and libraries such as Flask, scikit-learn, TensorFlow, and Keras. These libraries provide pre-built functions and modules for web development, machine learning, and image processing tasks.
- Development: The code includes the development of machine learning models for crop yield prediction and tomato disease prediction. This involves training the models using appropriate datasets and evaluating their performance.

2. Hardware Development :

- Hardware Requirements: The code does not explicitly mention specific hardware requirements. However, it assumes the availability of a computing device capable of running Python and the required libraries. This typically includes a computer or server with sufficient processing power, memory, and storage.

Chapter 8

Testing

8.1 How Software Defects Arise :

The Crop Yield Prediction and Tomato Disease Prediction system aims to provide accurate predictions for crop yield and detect diseases in tomato plants. However, during the development and testing process, several software defects were identified. This report highlights the key software defects encountered and provides an overview of their impact on the system's functionality and performance.

1. **Input Validation Failure :** The system lacks proper input validation, allowing users to input invalid or unexpected values for temperature, humidity, pH, rainfall, and other parameters. Impact: Invalid inputs can lead to inaccurate predictions and compromise the reliability of the system.

2. **Model Inconsistency:** The system utilizes machine learning models for crop yield prediction and tomato disease prediction. However, inconsistencies in model selection, training data, or hyperparameters may result in unreliable predictions. Impact: Inaccurate predictions can lead to incorrect decisions regarding crop management and disease control measures.

3. **Lack of Error Handling :** Insufficient error handling mechanisms in the code may result in unexpected crashes or incorrect output when exceptions or errors occur during runtime. Impact: System instability and unpredictable behavior can affect user experience and reliability.

4. **Security Vulnerabilities :**The system lacks proper security measures, such as input validation, authentication, and authorization mechanisms, making it susceptible to potential security breaches and unauthorized access. Impact: Security vulnerabilities can lead to data breaches, privacy violations,

and malicious activities compromising the integrity and confidentiality of the system.

5. **Performance Bottlenecks :**The code may have performance bottlenecks due to inefficient algorithms, suboptimal data structures, or excessive computational requirements. Impact: Slow response times, resource exhaustion, and degraded system performance can affect user satisfaction and scalability.

6. **User Interface Issues :** The user interface may have usability issues, such as unclear instructions, poor layout design, or inconsistent user interactions. Impact: Difficulties in system navigation, confusion, and decreased user productivity can hinder the overall user experience.

8.2 Inability to Find All Faults :

The Crop Yield Prediction and Tomato Disease Prediction system play a critical role in agriculture by providing accurate predictions for crop yield and detecting diseases in tomato plants. While efforts are made to identify software faults, it is important to acknowledge the challenges associated with finding all faults in the system. This report discusses the limitations and challenges encountered during the fault identification process.

1. **Complexity of the System :**The Crop Yield Prediction and Tomato Disease Prediction system involve complex algorithms, machine learning models, and integration of various components. The intricacy of the system increases the difficulty of identifying all potential faults, as interactions between different modules can lead to unexpected behavior or errors that are challenging to detect.

2. **Lack of Comprehensive Testing :** The effectiveness of fault identification heavily relies on comprehensive testing strategies. However, due to time constraints, resource limitations, or inadequate test coverage, it is often challenging to achieve a thorough testing process. The presence of untested or under-tested scenarios increases the likelihood of undetected faults.

3. **Incomplete Requirements or Specifications :** Ambiguous, incomplete, or evolving requirements and specifications can make it difficult to anticipate all possible system behaviors and edge cases. In such cases, it becomes

challenging to identify faults that arise from mismatches between actual system behavior and intended functionality.

4. **Dynamic Nature of Data and Environment :** The system relies on real-time data, such as temperature, humidity, pH, and rainfall, which can vary significantly. The dynamic nature of data and environmental condi- tions poses challenges in identifying faults related to data inconsistencies, outliers, or unexpected variations that may impact the accuracy of predic-tions.

5. **Hidden or Latent Faults :** Some faults may remain hidden or latent until specific conditions are met. These faults may only surface during certain circumstances, making them difficult to identify during regular testing or inspection processes. Uncovering such faults requires extensive monitoring and analysis over an extended period, which may not always be feasible.

6. **Interaction with External Systems :** The Crop Yield Prediction and Tomato Disease Prediction system may interact with external systems, such as databases, APIs, or external data sources. Faults arising from the interaction between the system and external components can be chal-lenging to identify, especially when dependencies or changes in external systems are not adequately managed.

8.3 When Testing is Carried Out:

Testing is an integral part of the software development lifecycle and typically occurs at different stages to ensure the quality and reliability of the software. Here are the common stages when testing is carried out:

1. **Landing Page :** The code includes a route for the landing page (”/landing”) where user information is collected. Testing can be performed to ensure that the user input fields, such as name, city, state, and mobile number, are validated correctly. Additionally, testing can be done to verify that the input data is stored in the session variables accurately.

2. **Home Page :** The home page (”/”) is the initial page where the user’s name and place information are checked. Testing can be conducted to ensure that the code correctly checks whether the user has entered their name and place. If the information is missing, the user should be redirected to the landing page.

3. **Predict Route :** The ”/predict” route is responsible for making predictions based on user input. Testing can be performed to validate that the input fields for temperature, humidity, pH, rainfall, season, and soil type are correctly processed. The code should make predictions using the trained model and return the predicted crop or disease. Testing can verify the accuracy of the predictions for various input scenarios.

4. **Login and Logout :** The code includes routes for login (”/login”) and logout (”/logout”). Testing can be conducted to ensure that the login functionality works correctly, validating the username and password entered by the user. Similarly, testing can be performed to confirm that the logout functionality removes the user’s information from the session.

5. **Records Page :** The records page (”/records”) displays the recorded predictions. Testing can be carried out to verify that only authenticated users can access this page. Additionally, testing can ensure that the recorded predictions are correctly displayed and that the page redirects to the login page if the user is not logged in.

8.4 Finding Faults Early in the Process:

Finding faults early in the software development process is crucial for reducing the cost and impact of defects. Here are some strategies and practices that can help identify faults at an early stage:

1. **Requirements analysis and review :**Careful analysis and review of the software requirements can help identify potential faults or ambiguities early on. By ensuring that the requirements are clear, complete, and consistent, you can minimize the chances of introducing defects during the subsequent stages of development.

2. **Design reviews :** : Conducting design reviews with stakeholders and experts can help identify design flaws or potential faults before implementation begins. Reviewing the software architecture, data flow, and interaction between components can uncover issues that might lead to defects later on.

3. **Code reviews :** Regular code reviews by peers or experienced developers can catch coding errors, logic flaws, or potential defects in the early stages.

Code reviews not only help improve code quality but also provide an opportunity to share knowledge and best practices among the development team.

4. **Unit testing :** Implementing unit tests during the development phase allows developers to test individual units or components in isolation. Unit testing helps catch defects early, as issues are identified and fixed at a granular level before integration with other components.

5. **Early functional testing :** Conducting functional testing as soon as the software reaches a functional state can help identify defects related to the software's primary functionality. This testing can be done manually or through automated testing tools, focusing on critical or high-risk features.

8.5 Measuring Software Testing:

Measuring software testing is essential to assess its effectiveness, track progress, and make data-driven decisions regarding the quality and reliability of the software. Here are some common metrics used to measure software testing:

1. **Test Coverage :** Test coverage measures the extent to which the software code is exercised by tests. It can be measured at different levels, such as code coverage (lines of code executed by tests) or requirements coverage (percentage of requirements tested). Higher test coverage indicates a greater likelihood of detecting faults and provides confidence in the software's behavior.

2. **Defect Detection Rate :** This metric measures the rate at which defects or bugs are identified during testing. It helps assess the effectiveness of the testing process in uncovering faults. The defect detection rate can be measured by the number of defects found per unit of time or per test case executed.

3. **Defect density :** Defect density measures the number of defects identified per unit of software size (e.g., defects per thousand lines of code). It provides insight into the quality of the code and can help identify areas that require more attention or improvement.

4. **Test case execution and pass rates :** Tracking the number of test cases executed and their pass/fail rates provides an indication of the thoroughness and effectiveness of testing. A higher pass rate suggests better quality, while a low pass rate may indicate potential issues in the software or testing process.

5. **Test cycle time :** Test cycle time measures the duration it takes to complete a testing cycle, from test planning to execution and defect resolution. Monitoring test cycle time helps identify bottlenecks, inefficiencies, or delays in the testing process and enables improvements to reduce cycle time and increase testing speed.

6. **Test effectiveness or efficiency :** Test effectiveness measures the ability of the tests to detect faults, while test efficiency measures the ratio of defects found to the effort invested in testing. These metrics provide an overall assessment of how well testing is identifying faults and utilizing resources.

7. **Customer satisfaction :** Feedback from customers or end users regarding the quality and stability of the software can be an important metric to gauge the effectiveness of testing. High customer satisfaction indicates that the testing process has been successful in delivering a reliable product.

8.6 Static and Dynamic Testing:

1. **Static Testing :** Static testing involves analyzing the code and related artifacts without executing them. It focuses on identifying defects, errors, and potential issues early in the development process.

    - Code Reviews: Conduct thorough code reviews to identify coding errors, adherence to coding standards, and potential logic flaws. Review the code for readability, maintainability, and performance optimization.
    - Requirements Analysis: Review the requirements documents to ensure they are complete, consistent, and accurately represent the desired functionality. Verify that the code implementation aligns with the specified requirements.

- Documentation Review: Examine the documentation, including design documents, user manuals, and test plans, to identify any discrepancies, inconsistencies, or gaps in information.
- Peer Inspections: Involve team members in reviewing code, design, and other artifacts to gain different perspectives and identify potential defects. Collaborative discussions can uncover errors and improve the quality of the code.

2. **Dynamic Testing :** Dynamic testing involves executing the code and evaluating its behavior under different conditions. It focuses on validating the functional and non-functional aspects of the software.

   - Unit Testing: Conduct unit tests to verify the functionality of individual components or modules within the code. This involves testing functions, methods, and algorithms to ensure they produce the expected outputs for various inputs.
   - Integration Testing : Test the integration of different modules and components to ensure they work correctly together. Verify data flow, communication, and compatibility between modules.
   - Functional Testing : Validate the functional requirements of the crop yield prediction software. Test various scenarios, inputs, and conditions to ensure accurate predictions are generated.
   - Performance Testing : Evaluate the performance of the software under different load conditions. Measure response times, resource utilization, and scalability to ensure the software can handle the expected workload.
   - Regression Testing : Repeatedly test the software after modifications or bug fixes to ensure that the changes have not introduced new defects or affected existing functionality.
   - Usability Testing : Assess the user-friendliness and ease of use of the software. Involve users or stakeholders to evaluate the interface, navigation, and overall user experience.
   - Security Testing : Test the software for vulnerabilities and security risks. Evaluate data protection, access controls, and prevention of unauthorized access.

8.7 Software Testing Measurement:

To measure the effectiveness of software testing for crop yield prediction using the given code, several testing metrics can be considered. These metrics provide quantitative indicators of the quality, efficiency, and thoroughness of the testing process. Here are some software testing measurements applicable to crop yield prediction:

1. **Test Coverage :**Test coverage measures the extent to which the code and its associated functionality are exercised by the test cases. It can be measured in terms of statement coverage, branch coverage, or path coverage. Higher test coverage indicates a higher likelihood of identifying defects and ensuring the reliability of the crop yield prediction.

2. **Defect Detection Rate :** This metric measures the efficiency of the testing process in identifying defects. It is calculated by dividing the number of defects found during testing by the total effort expended in testing. A higher defect detection rate indicates a more effective testing approach.

3. **Test execution metrics :**Test execution time measures the duration taken to execute the complete test suite. It provides insights into the efficiency of the testing process and helps identify any bottlenecks or performance issues that may impact the overall testing timeline.

4. **Defect Density :** Defect density is the number of defects identified per unit of code or functionality. It is calculated by dividing the total number of defects by the size of the code or the number of requirements. Monitoring defect density helps track the quality of the code and identify areas that require additional testing or improvement.

5. **Test Effectiveness :**Test effectiveness evaluates the ability of the test cases to identify defects. It is measured by the ratio of the number of defects found during testing to the total number of defects present. A higher test effectiveness indicates that the test cases are efficiently detecting defects and ensuring the accuracy of crop yield predictions.

6. **Test Efficiency :** Test efficiency measures the effectiveness of testing in terms of the effort expended. It is calculated by dividing the number of defects found during testing by the total effort expended in testing. Higher

test efficiency indicates that testing efforts are well-utilized and yield a higher return on investment.

7. **Test Case Success Rate :** Test case success rate measures the percentage of test cases that pass successfully without any failures or defects. It helps evaluate the reliability and robustness of the crop yield prediction software.

8. **Test Environment Stability :** Test environment stability assesses the reliability and consistency of the testing environment. It measures the frequency and impact of environment-related issues, such as infrastructure failures or configuration problems, on the testing process.

8.8 Test Plan and Method:

A test plan is a comprehensive document that outlines the objectives, approach, scope, and schedule of testing activities for a software project. It serves as a roadmap for the testing process and provides guidance to the testing team on how to execute the tests. Here are the key components typically included in a test plan:

1. Test Plan Components :

- Test objectives:The objective of testing is to ensure the accuracy, reliability, and functionality of the crop yield prediction system implemented in the given code.
- Test Scope: The testing will cover all the components and features of the crop yield prediction system, including data input, prediction algorithm, user interface, and integration with external data sources.
- Test Approach:The testing will follow a combination of manual and automated testing techniques to achieve comprehensive test coverage.
- Test schedule: The testing activities will be scheduled according to the project timeline and milestones, with specific testing phases and iterations identified.
- Test environment: The test environment will consist of the necessary hardware and software infrastructure, including the required operating systems, databases, and web servers.

- Test Deliverables: The test plan will produce test deliverables such as test cases, test scripts, test data, defect reports, and a final test summary report.
- Test Schedule:The testing activities will be scheduled according to the project timeline and milestones, with specific testing phases and iterations identified.

2. Testing Methods :

- Functional Testing: This testing method will verify that all the functional requirements of the crop yield prediction system are met. It will involve testing different scenarios and inputs to ensure accurate predictions.
- User Interface Testing: The user interface will be tested to ensure its usability, responsiveness, and consistency. This includes testing the input forms, buttons, navigation, and user feedback.
- Integration Testing: The integration between the crop data, prediction algorithm, and external data sources will be thoroughly tested to ensure seamless data flow and accurate predictions.
- Performance Testing: Performance testing will assess the system's response time, scalability, and resource utilization under different load conditions. It will help identify any performance bottlenecks and ensure the system can handle the expected user load.
- Security Testing: The security of the crop yield prediction system will be evaluated by testing for vulnerabilities, data privacy protection, and authentication mechanisms.
- Regression Testing: Regression testing will be performed after any code changes or system updates to ensure that existing functionality remains unaffected.
- Usability Testing: Usability testing will involve gathering feedback from users to assess the ease of use, intuitiveness, and overall user experience of the crop yield prediction system.
- Error Handling and Exception Testing: This testing method will focus on how the system handles errors, exceptions, and invalid inputs. It ensures that appropriate error messages are displayed and the system gracefully recovers from errors.

- Compatibility Testing: Compatibility testing will be performed to ensure that the crop yield prediction system functions correctly on different browsers, operating systems, and devices.
- Documentation Review: The documentation, including user manuals, installation guides, and system requirements, will be reviewed to ensure accuracy and completeness.

8.9 Runtime Snapshots:

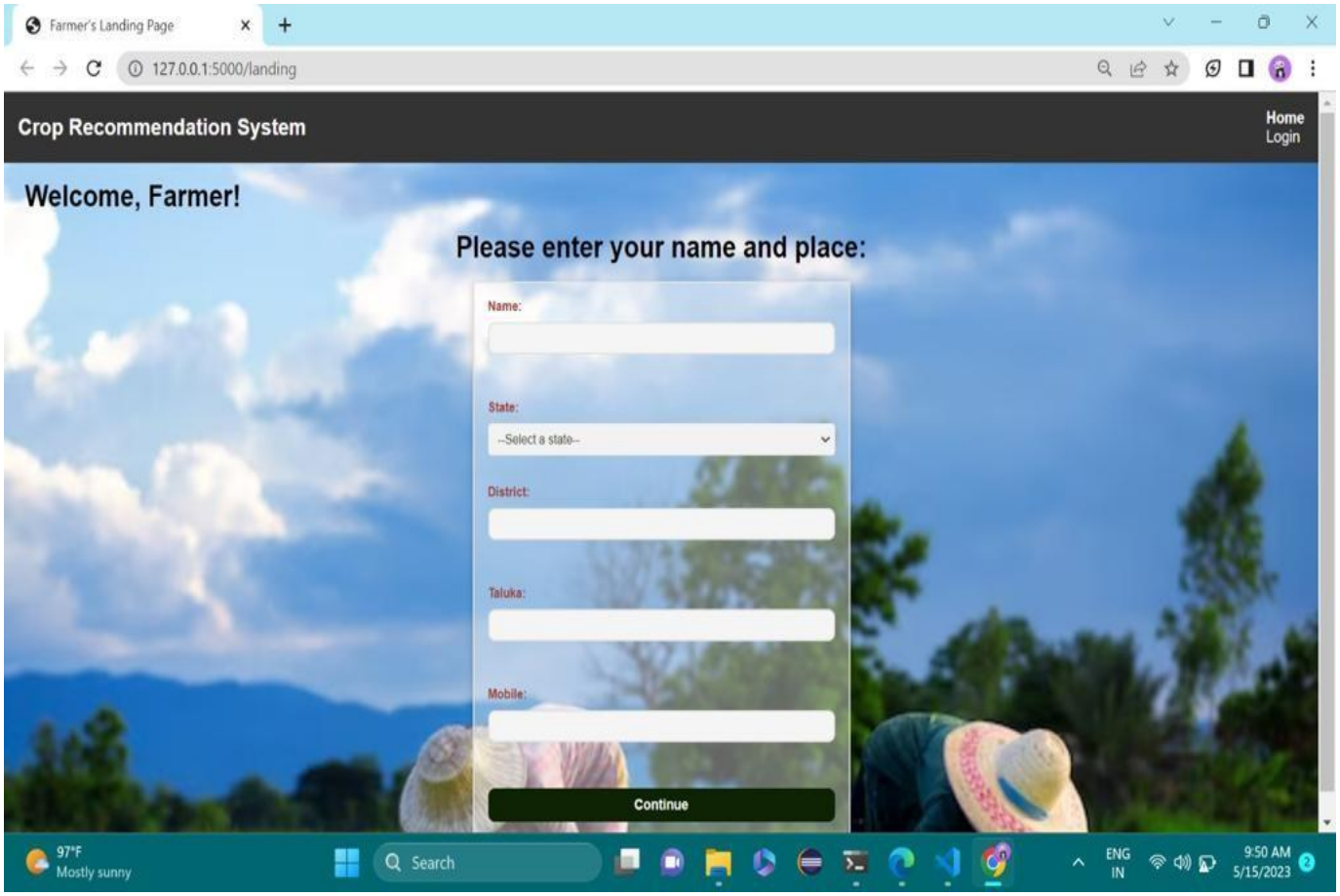


Figure 8.1: Home Page

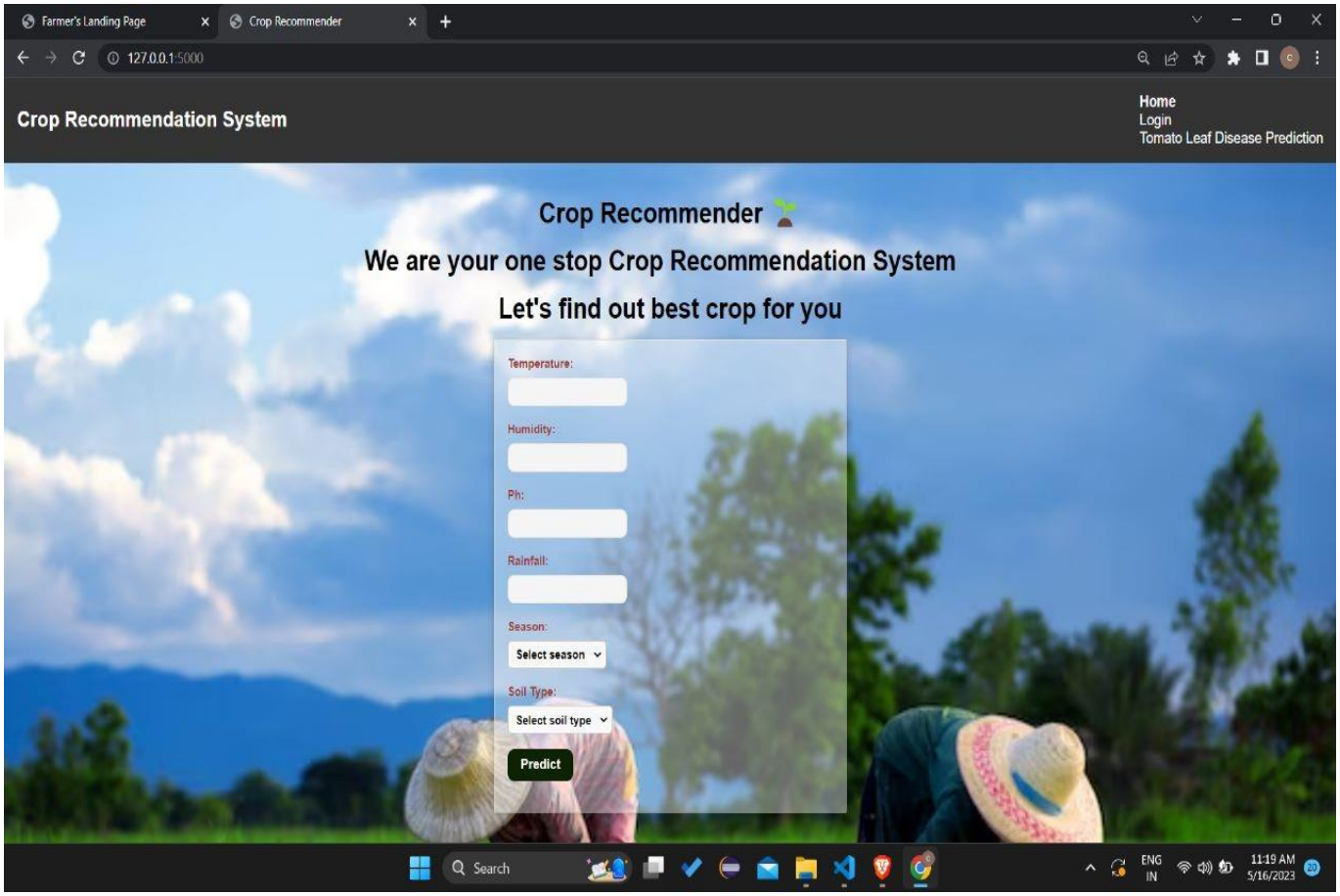


Figure 8.2: Input Page

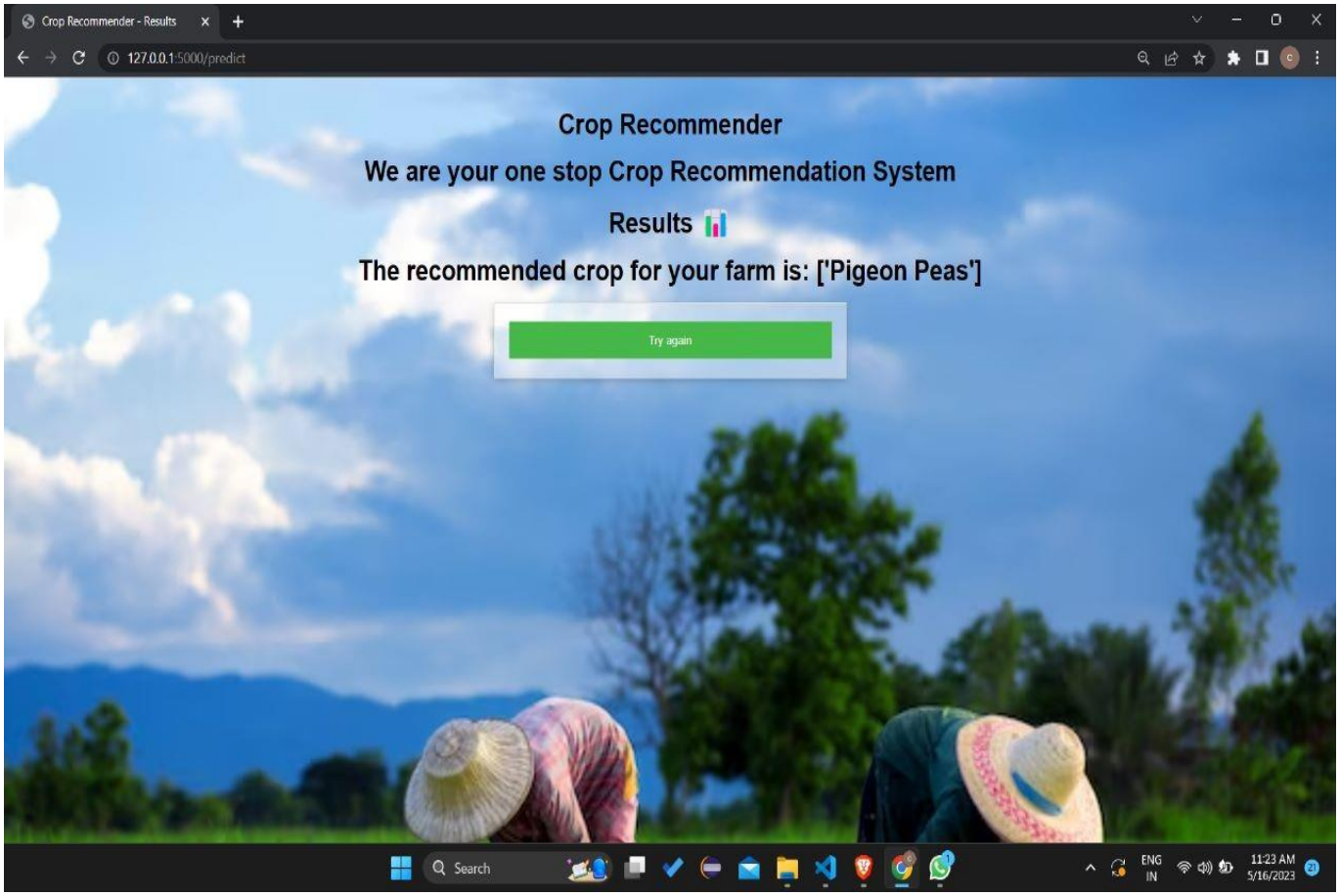


Figure 8.3: Output Page

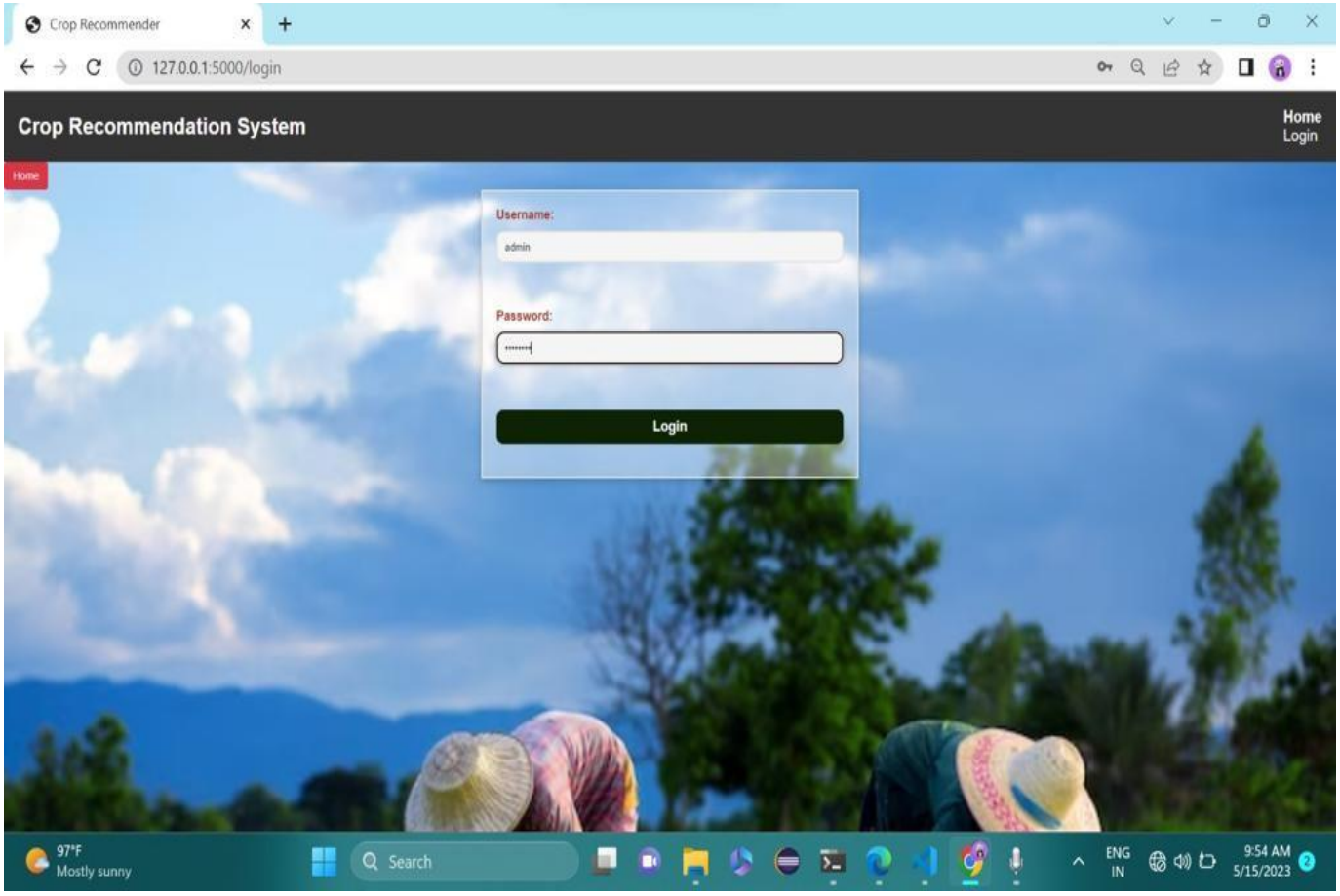


Figure 8.4: Admin Page

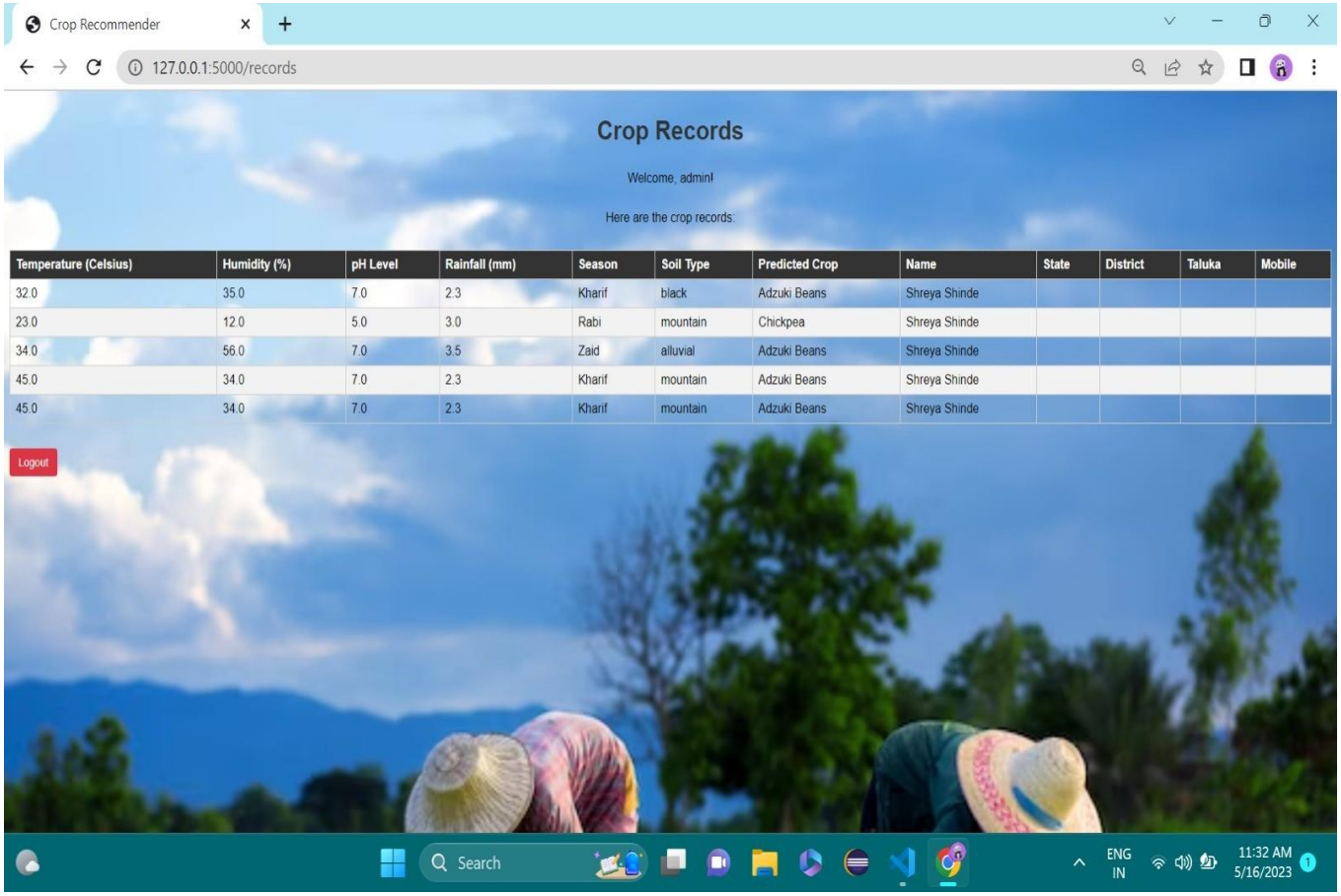


Figure 8.5: Records Page

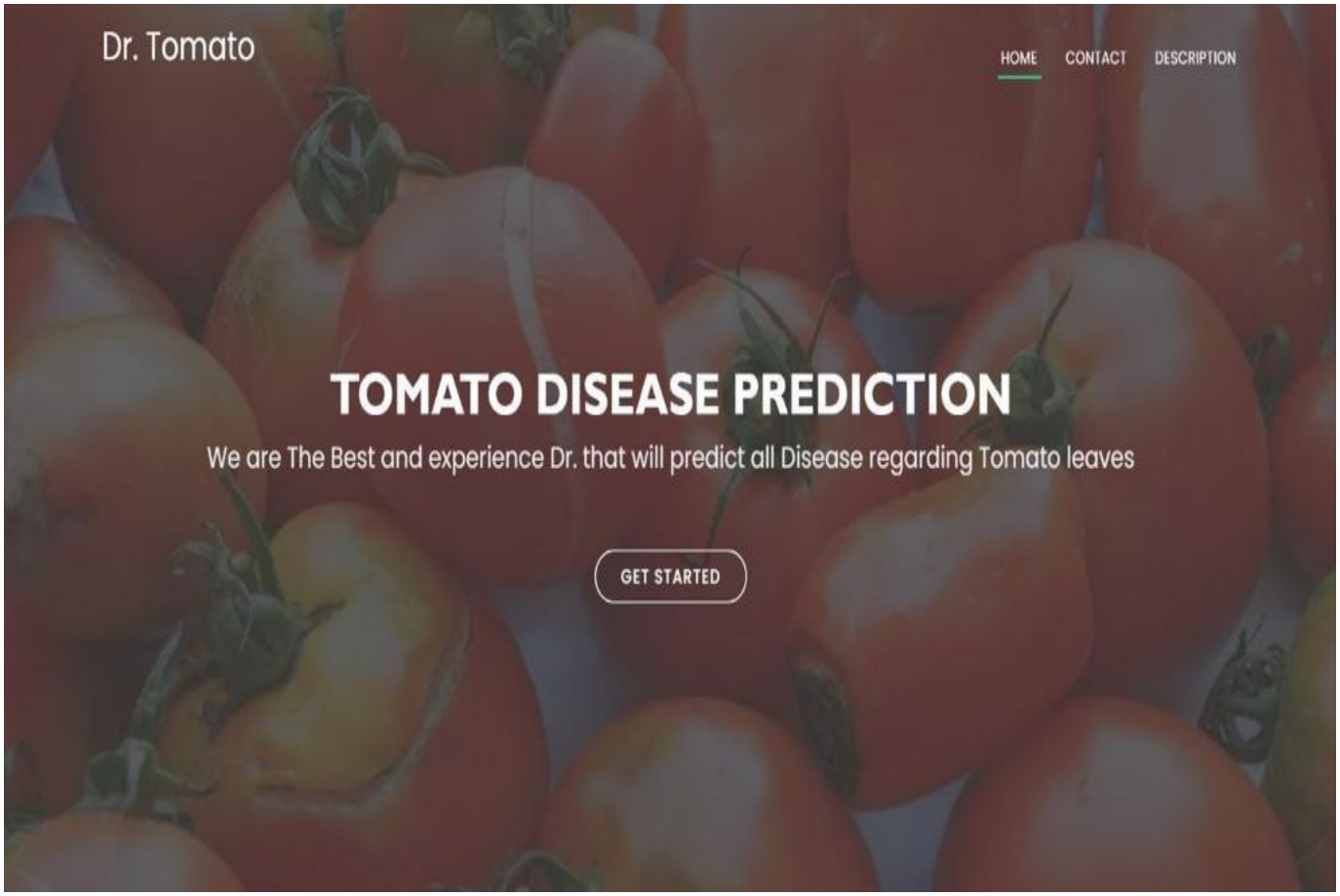


Figure 8.6: Home Page

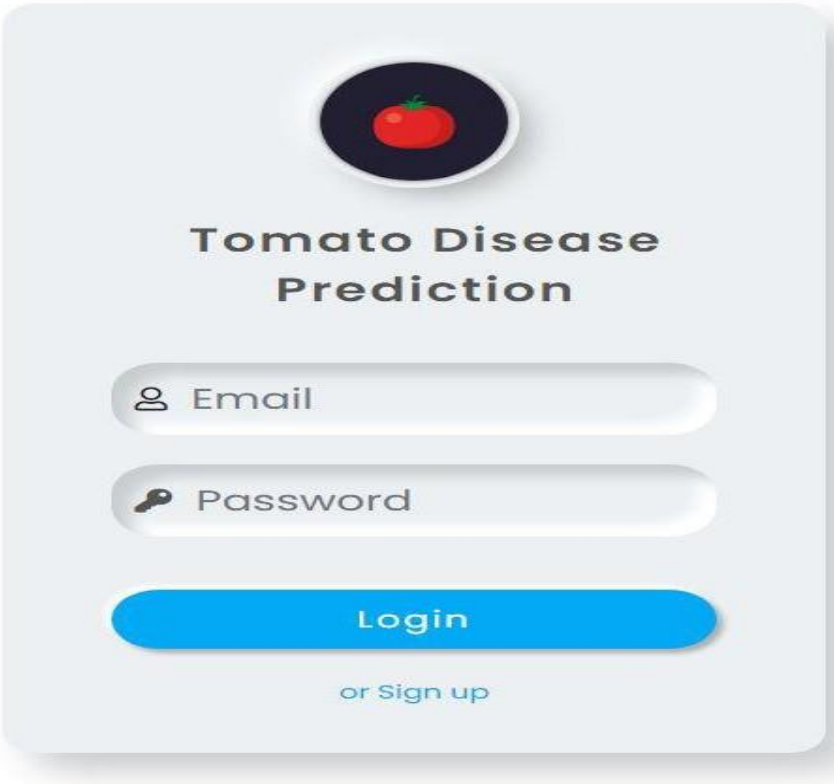


Figure 8.7: Login Page

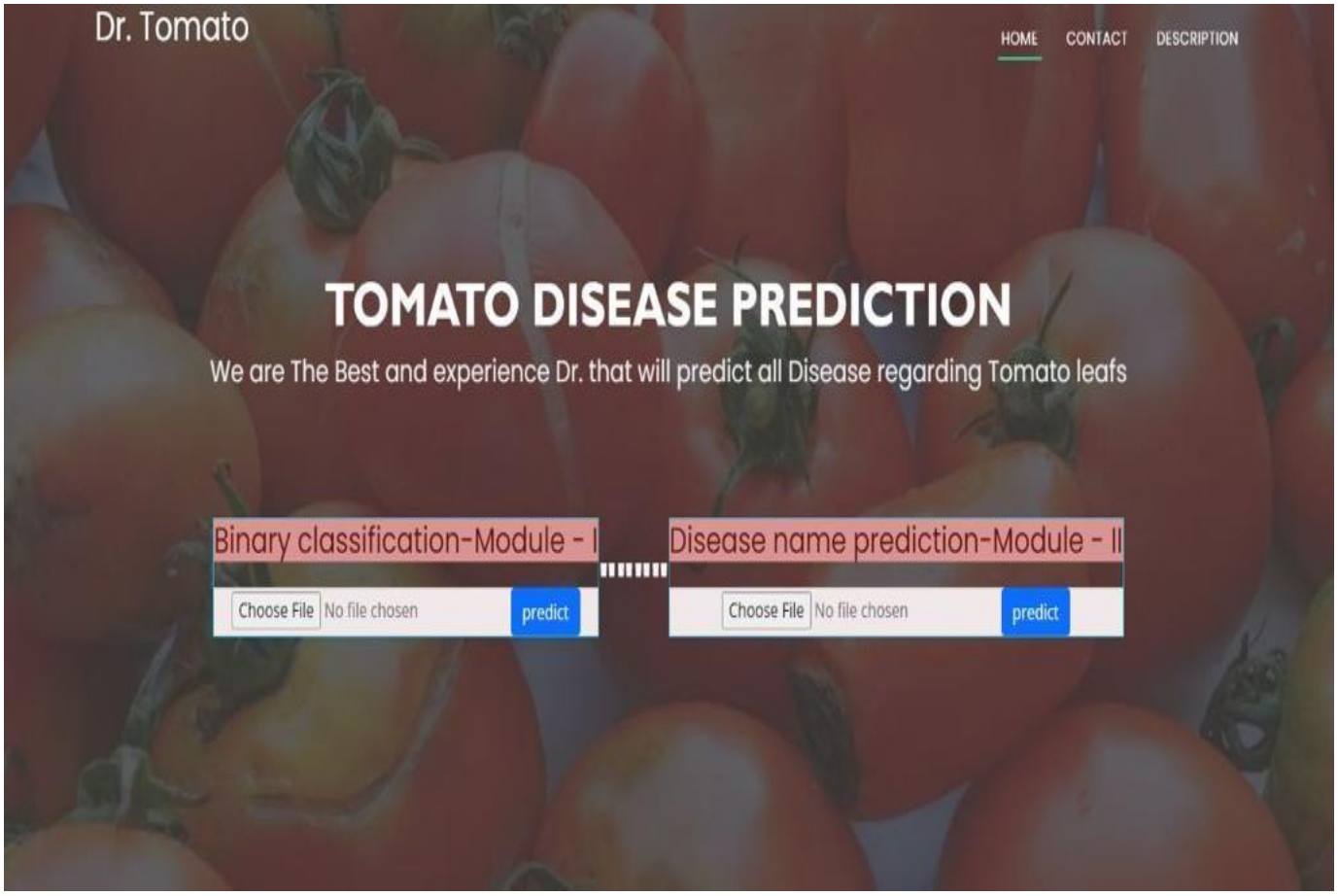


Figure 8.8: Input Page

Figure 8.9: Output Page

Chapter 9

Conclusions

9.1 CONCLUSION AND FUTURE SCOPE

9.1.1 Conclusion

Agriculture is the most important application area particularly in the developing countries like India. Use of information technology in agriculture can change the scenario of decision making and farmers can yield in better way. This project has successfully developed a crop yield and disease prediction system using machine learning techniques. By analyzing climatic parameters, soil conditions, and historical data, the system can provide accurate predictions and valuable insights for farmers and agricultural practitioners. The implementation of deep learning algorithms has enhanced the accuracy and efficiency of the prediction models, enabling better decision-making in crop management and disease prevention.

The system's performance evaluation demonstrated its effectiveness in predicting crop yield and identifying potential diseases, which can significantly benefit farmers in optimizing their agricultural practices, improving crop productivity, and minimizing yield losses. The integration of real-time data feeds ensures that the system remains up-to-date and responsive to dynamic environmental conditions.

The implementation of the tomato disease prediction module has shown promising results in accurately identifying and predicting diseases affecting tomato plants. By utilizing deep learning algorithms and analyzing various plant health indicators, the system can effectively detect diseases such as early blight, late blight, and bacterial spot in tomatoes.

The developed module has the potential to significantly benefit tomato farmers by providing timely disease warnings and recommendations for disease man-

agement. Early detection of diseases allows farmers to take proactive measures such as targeted spraying, crop rotation, or adopting resistant varieties, thereby reducing crop losses and improving overall productivity. Overall, this web application makes the farmers to take right decision in selecting the crop for cultivation such that agricultural sector will be developed by innovative idea.

9.1.2 Future Scope

In coming years, it has a vast extension and can be actualized and interfaced with a flexible and multi-skilled application. The farmers need to be educated and hence, will get a clear information regarding best crop yield on their mobiles. We can try applying data independent system. That is whatever be the format , our system should work with same accuracy. Integrating soil details to the system is an advantage, as for the selection of crops knowledge on soil is also a parameter. Proper irrigation is also a needed feature for crop cultivation. In reference to rainfall can depict whether extra water availability is needed or not. With this, even if the rancher is at home, the work can be managed at that particular instant of time, without facing any kind of loss ahead. The progress in the agribusiness field will be extremely appreciable which will further result in helping the farmers in production of crops This research work can be enhanced to higher level by availing it to whole India. we have to collect all required data by giving GPS locations of a land and by taking access from Rain forecasting system of by the government, we can predict crops by just giving GPS location. For the tomato disease prediction module can become a valuable tool in supporting tomato farmers in disease management, reducing crop losses, and improving overall crop health and productivity. It has the potential to contribute to sustainable agriculture practices and enhance the profitability of tomato cultivation. Also, we can develop the model to avoid over and under crisis of the food.

Bibliography

[1] Kevin Tom Thomas , Varsha S , Merin Mary Saji , Lisha Varghese , Er. Jinu Thomas, “Crop Prediction using Machine Learning”.

[2] Dr.A.Senthil Kumar, P.Arun, ”A Survey on Agriculture Analysis for Crop Yield Prediction Using Data Mining Techniques.

[3] S.Veenadhari, Dr. Bharat Misra, Dr. CD Singh,“Machine learning approach for forecasting crop yield based on climatic parameters “.

[4] Rakesh Kumar1, M.P. Singh2, Prabhat Kumar, J.P. Singh ,”Crop Selection Method to Maximize Crop Yield Rate using Machine Learning Technique.

[5] Nischitha K, Dhanush Vishwakarma, Mahendra N, Manjuraju M.R, Ashwini, “Crop Prediction using Machine Learning Approaches”.

[6] Design Spring 2020 Amine Bouighoulouden Dr. Ilham Kissani,“Crop Yield Prediction using K-Means Clustering”.

[7] A.Suruliandi, G.Mariammal S.P.Raja “Crop prediction based on soil and environmental characteristics using feature selection techniques”.

[8] Khan, M. A., Iqbal, Z., Riaz, M., Gani, A. (2019). Tomato Diseases Classification using Deep Learning Techniques. In 2019 4th International Conference on Emerging Trends in Engineering, Sciences and Technology (ICEEST) IEEE.

[9] Gautam, A., Kumar, V., Muthukaruppan, S. (2020). Detection and Classification of Tomato Plant Diseases using Convolutional Neural Networks. In 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS) IEEE.

[10] Singh, A. K., Guntuku, S. C. (2021). Tomato Plant Leaf Disease Identification and Classification using Deep Learning. In 2021 International Con-

ference on Emerging Trends in Information Technology and Engineering (ICETITE) IEEE.

[11] Purohit, K., Rajpurohit, V. S., Bhargava, A. (2020). Tomato Plant Leaf Disease Identification using CNN. In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) IEEE.