

PLANT LEAF DISEASES FINE-GRAINED CATEGORIZATION USING CONVOLUTIONAL NEURAL NETWORKS

A PROJECT REPORT

Submitted by

MPS GAYATHRI

[REGISTER NO:211419104077]

KV KAVIYAA

[REGISTER NO:211419104132]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2023

PANIMALAR ENGINEERING COLLEGE
(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report “**Plant Leaf Diseases Fine-Grained Categorization Using Convolutional Neural Networks**” is the bonafide work of “ **MPS GAYATHRI (211419104077), KV KAVIYAA (211419104132)** ” who carried out the project work under my supervision.

SIGNATURE

**Dr.L.JABASHEELA,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT**

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

**MRS.R.DEVI, M.E.,
SUPERVISOR
ASSISTANT PROFESSOR
(GRADE – 1)**

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce

Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION BY THE STUDENT

We “**MPS GAYATHRI (211419104077), KV KAVIYAA (211419104132)**” hereby declare that this project report titled “**Plant Leaf Diseases Fine- Grained Categorization Using Convolutional Neural Networks**”, under the guidance of **MRS.R.DEVI**, M.E. is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

MPS.GAYATHRI

KV.KAVIYAA

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt. C.VIJAYARAJESWARI, Dr.C. SAKTHI KUMAR, M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.**, for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr. K.MANI, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L.JABASHEELA, M.E., Ph.D.**, for the support extended throughout the project.

We would like to thank our **Project Guide, MRS.R.DEVI, M.E.**, and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

MPS.GAYATHRI

KV.KAVIYAA

ABSTRACT

The timely identification and early prevention of crop diseases are essential for improving production. In this report, Deep Convolutional Neural Network (CNN) models are implemented to identify and diagnose diseases in plants from their leaves, since CNNs have achieved impressive results in the field of machine vision. Standard CNN models require many parameters and higher computation costs. In this report, we replaced standard convolution with depth = separable convolution, which reduces the parameter number and computation cost. To evaluate the performance of the models, different parameters such as batch size, dropout and different numbers of epochs were incorporated. The implemented models achieved a disease-classification accuracy rate of 98.42%, 99.11%, 97.02% and 99.56% using InceptionV3, InceptionResNetV2, MobileNetV2 and EfficientNetB0 respectively, which were greater than that of traditional handcrafted-feature-based approaches. In comparison with other deep-learning models, the implemented model achieved better performance in terms of accuracy, and it required less training time. Moreover, the MobileNetV2 architecture is compatible with mobile devices using the optimized parameter. The accuracy results in the identification of diseases showed that the deep CNN model is promising and can greatly impact the efficient identification of the diseases which have potential in the detection of diseases in real-time agricultural systems.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	
	1.1 Overview	2
	1.2 Problem Definition	3
2	LITERATURE SURVEY	5
3	SYSTEM ANALYSIS	
	3.1 Existing System	12
	3.2 Proposed System	12
	3.3 Feasibility Study	13
	3.4 Project Requirement	14
	3.5 Environment Requirement	15
4	SYSTEM DESIGN	
	4.1 ER Diagram	17
	4.2 Dataflow Diagram	18
	4.3 UML Diagram	21
	4.4 Class Diagram	22
	4.5 Activity Diagram	23
	4.6 Sequence Diagram	24

5	SYSTEM ARCHITECTURE	
	5.1 Module Description	27
	5.2 Proposed System algorithm	31
6	SYSTEM TESTING	
	6.1. Unit Testing	42
	6.1.2 Integration Testing	43
	6.4 Test Case Report	44
	PERFORMANCE EVALUATION	
7	7.1 Result and discussion	47
	7.2 Performance metrics	47
	CONCLUSION	
8	8.1 Conclusion	52
	8.2 Future works	52
	APPENDICES	
	A Sample Dataset	53
	B Sample Coding	54
	C Sample Screenshot	67
	REFERENCES	70

LIST OF FIGURES

FIGURE NO.	FIGURE TITLE	PAGE NO.
Fig 4.1	ER DIAGRAM	17
Fig 4.2.1	DATAFLOW DIAGRAM LEVEL 0	18
Fig 4.2.2	DATAFLOW DIAGRAM LEVEL 1	19
Fig 4.2.3	DATAFLOW DIAGRAM LEVEL 2	20
Fig 4.3	UML DIAGRAM	21
Fig 4.4	CLASS DIAGRAM	22
Fig 4.5	ACITVITY DIAGRAM	23
Fig 4.6	SEQUENCE DIAGRAM	24
Fig 5.1	SYSTEM ARCHITECTURE	26
Fig 5.5.1	WORKFLOW DIAGRAM FOR PROPOSED SYSTEM	32
Fig 5.3.4	RESISDUAL NETWORK LAYERS 9 LAYERS	39
Fig 8.1	CONFUSION MATRIX FOR CROP DETECTION	48
Fig 8.2	F1 SCORES FOR PREDICTION MODEL	49
Fig 8.3	ACCURACY vs NO. OF EPOCHS	49
Fig 8.4	LOSS vs NO. OF EPOCHS	50
Fig 8.5	LEARNING RATE vs BATCH NUMBER	50
Fig C3.1 – C3.6	SCREENSHOTS OF OUTPUT SCREENS	67-69

LIST OF ABBREVIATIONS

ABBREVIATIONS	EXPLANATION
API	Application Programming Interface
IDE	Integrated Development Environment
CNN	Convolutional Neural Network
KNN	K- Nearest Neighbor
XAI	Explainable Artificial Intelligence
LIME	Local Interpretable Model- Agnostic Explanation
UI	User Interface
HTTP	Hypertext Transfer Protocol
SVM	Support Vector Machines
DB	Database
RAM	Random Access Memory
ResNet	Residual Networks
ANN	Artificial Neural Network
ILSVRC	Image Net Large Scale Visual Recognition
VGG-16	Visual Geometry Group
ReLU	Rectified Linear Activation Unit
CSS	Cascading Style Sheet

CHAPTER 1

INTRODUCTION

1 INTRODUCTION

1.1 OVERVIEW

The agriculture sector has long relied on contemporary science to supply food for 7 billion people. The impact of plant disease, among the numerous risks, is actually significant because it not only results in significant losses of plants intended for human use, but also negatively impacts the lives of farmers whose primary source of income is the cultivation of healthy crops. Human experts laboriously inspect mature plants during the plant harvesting process to ensure they are free of disease and fit for human consumption before removing them. However, this conventional visual method of diagnosing the illness that a certain plant has costs money and takes a long time. Furthermore, it only makes sense that this process be mechanized in order to meet the rising demands of the populace given the apparent daily growth in population around the globe.

In comparison to the conventional visual identification of plant illnesses, the early detection of plant diseases is now considerably simpler, quicker, and less expensive via the advent of machine learning models. Since there has been a lot of study in this area recently, machine learning models are gradually taking the place of traditional plant disease identification in the business. The objective of this thesis is to implement two different machine learning models—the Convolutional Neural Network (CNN) and the K-Nearest Neighbor (KNN) model—on the plant village dataset and to assess the performance of the aforementioned models using the metrics Accuracy, Precision, Recall, and F1-Score.

The originality of this study comes in the use of the Explainable Artificial Intelligence (XAI) approach and Local Interpretable Model-Agnostic Explanations to provide transparency and explainability for the judgements made by the aforementioned models (LIME). The specialty of this report lies in the fact that it

aims to not only implement but also explain and provide transparency to the users on the predictions made by the aforementioned machine learning models. The use of XAI in order to explain the predictions made by the machine learning models is very uncommon and not to be found in many research papers in this particular domain.

1.2 PROBLEM STATEMENT

India is known to be the world's largest producer of pulses, rice, wheat, spices and spice products. Farmer's economic growth depends on the quality of the products that they produce, which relies on the plant's growth and the yield they get. Therefore, in the field of agriculture, detection of disease in plants plays an instrumental role. Plants are highly prone to diseases that affect the growth of the plant which in turn affects the ecology of the farmer. Utilizing the right methods to distinguish between healthy and diseased leaves reduces crop loss and boosts productivity. This section includes many machine-learning methods now in use for identifying plant diseases. The majority of farmers lack any interest in expanding, which could result in sufferings that are more severe than half because of pests and plant diseases. Additionally, it is applicable in other disciplines of agricultural disease identification with comparable application scenarios. In order to detect a plant disease at the very initial stage, use of automatic disease detection technique is advantageous. The symptoms of plant diseases are conspicuous in different parts of a plant such as leaves, etc. Manual detection of plant disease using leaf images is a tedious job. Hence, it is required to develop computational methods which will make the process of disease detection and classification using leaf images automatic.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE REVIEW

The purpose of literature review is to gain an understanding of the existing research and debates relevant to a particular topic or area of study, and to present that knowledge in the form of a written report. Conducting a literature review helps to build knowledge in a particular field.

PAPER 1

TITLE: ESTIMATION OF LEAF ANGLE DISTRIBUTION BASED ON STATISTICAL PROPERTIES OF LEAF SHADING DISTRIBUTION

AUTHOR: Kuniaki Uto , Mauro Dalla Mura , Yuka Sasaki and Koichi Shinoda

YEAR: 2020

ABSTRACT:

Leaf angle distribution is an important phenotype parameter that is related to photosynthesis. Thanks to the recent advent of drones and high-resolution imaging devices, leaf-scale aerial images with high spectral and spatial resolution are available. This work is the first attempt to utilize a single leafscale image to differentiate plants with different leaf angle distribution. First, assuming that a rice leaf surface resembles a section of a hemiellipsoid surface, a collection of rice leaf surfaces is approximated by a hemiellipsoid surface. Timeseries of shading distributions on the hemiellipsoids with different structural parameters under different direct sunlight directions are generated. By investigating the statistical properties, i.e., skewness, kurtosis and the most probable intensity, of the frequencies of the simulated shading intensity that welldifferentiate hemiellipsoids

with different structural parameters, we identified an appropriate time slot, i.e., 11:00-12:30, for image acquisitions. Then, time-series leaf-scale images and depth maps of rice plants with/without silicate fertilizer under sunlight were collected. Based on the depth maps, it was confirmed that silicate fertilizer dosed leaves are more upright than leaves from non treated plants. It was demonstrated that 89% and 100% of kurtosis and the most probable intensity of the leaf-scale images during the appropriate time slot showed consistent relations with the simulations, which indicates that the proposed method is useful to distinguish different leaf angle distributions based on the frequency of shading intensity of rice leaf images.

PAPER 2

TITLE: Plant Species Identification from Occluded Leaf Images

AUTHOR: Ayan Chaudhury, and John L. Barron

YEAR: 2020

ABSTRACT:

We present an approach to identify the plant species from the contour information from occluded leaf image using a database of full plant leaves. Although contour based 2D shape matching has been studied extensively in the last couple of decades, matching occluded leaves with full leaf databases is an open and little worked on problem. Classifying occluded plant leaves is even more challenging than full leaf matching because of large variations and complexity of leaf structures. Matching an occluded contour with all the full contours in a database is an NP-hard problem, so our algorithm is necessarily suboptimal. First, we represent the 2D contour points as a β -Spline curve. Then we extract interest points on these curves via the Discrete Contour Evolution (DCE) algorithm. We use subgraph matching using the DCE points as graph nodes, which produces a

number of open curves for each closed leaf contour. Next, we compute the similarity transformation parameters (translation, rotation and uniform scaling) for each open curve. We then “overlay” each open curve with the inverse similarity transformed occluded curve and use the Frechet distance metric to measure the quality of the match, retaining the best η matched curves. Since the Frechet metric is cheap to compute but not perfectly correlated with the quality of the match, we formulate an energy functional that is well correlated with the quality of the match, but is considerably more expensive to compute. The functional uses local and global curvature, Shape Context descriptors and String Cut features. We minimize this energy functional using a convex-concave relaxation framework. The curve among these best η curves, that has the minimum energy, is considered to be the best overall match with the occluded leaf. Experiments on three publicly available leaf image database shows that our method is both effective and efficient, outperforming other current state-of-the-art methods. Occlusion is measured as the percentage of the overall contour (and not leaf area) that is missing. We show our algorithm can, even for leaves with a high amounts of occlusion (say 50% occlusion), still identify the best full leaf match from the databases.

PAPER 3

TITLE: Tomato Septoria Leaf Spot Necrotic and Chlorotic Regions Computational Assessment Using Artificial Bee Colony-Optimized Leaf Disease Index

AUTHOR: Ronnie Concepcion, Sandy Lauguico, Elmer Dadios, Argel Bandala, Edwin Sybingco, Jonnel Alejandrino

YEAR: 2020

ABSTRACT:

Visual inspection of plant health status and disease severity may yield subjective assessments due to errorprone sphere of colors and textures as affected by angular photosynthetic light source and the complexity of chlorosis. Quantification of damages on leaves due to destructive diseases is paramount for plant and pathogen interactions. To address this challenge, the proposed solution is the integration of computer vision and computational intelligence for tomato Septoria leaf spot necrotic and chlorotic region computational assessment. Dataset contains healthy and diseased tomato leaves that were captured individually. Non-vegetation pixels removal was done using CIELab color space. RGB color components and five Haralick texture features were extracted from the segmented leaf. Hybrid neighborhood component analysis and ReliefF algorithm were employed to select the important predictors resulting to RGB-entropy vector. A new tomato leaf disease index (tomLDI) optimized using artificial bee colony (ABC) was developed by normalizing visible red reflectance, and introducing red-green and red-blue reflectance ratios to enhance Septoria leaf spots pixels and reducing sensitivity to healthy green pixels. KNN bested classification tree, linear discriminant analysis and Naïve Bayes in detecting Septoria leaf disease with accuracy of 97.46%. Deep transfer image regression was tested using raw infected leaf images and the tomLDI transformed colored channels through MobileNetV2, ResNet101 and InceptionV3. Using tomLDI channel, MobileNetV2 and ResNet101 bested other networks in estimating leaf diseased region percentage and number of Septoria spots with R2 values of 0.9930 and 0.9484 respectively. tomLDI channel proved to be more accurate than using raw images for regression.

PAPER 4

TITLE: Tomato Leaf Disease Identification by Restructured Deep Residual Dense Network

AUTHOR: CHANGJIAN ZHOU, SIHAN ZHOU, JINGE XING, AND JIA SONG

YEAR: 2021

ABSTRACT:

As COVID-19 spread worldwide, many major grain-producing countries have adopted measures to restrict their grain exports; food security has aroused great concern from various parties. How to improve grain production has become one of the most important issues facing all countries. However, crop diseases are a difficult problem for many farmers so it is important to master the severity of crop diseases timely and accurately to help staff take further intervention measures to minimize plants being further infected. In this paper, a restructured residual dense network was proposed for tomato leaf disease identification; this hybrid deep learning model combines the advantages of deep residual networks and dense networks, which can reduce the number of training process parameters to improve calculation accuracy as well as enhance the flow of information and gradients. The original RDN model was first used in image super resolution, so we need to restructure the network architecture for classification tasks through adjusted input image features and hyper parameters. Experimental results show that this model can achieve a top-1 average identification accuracy of 95% on the Tomato test dataset in AI Challenger 2018 datasets, which verifies its satisfactory performance. The restructured residual dense network model can obtain significant

Improvements over most of the state-of-the-art models in crop leaf identification, as well as requiring less computation to achieve high performance.

PAPER5

TITLE: Classification of Watermelon Leaf Diseases Using Neural Network Analysis

AUTHOR: Suhaili Beeran Kutty, Noor Ezan Abdullah, Dr. Hadzli Hashim

YEAR: 2020

ABSTRACT:

This paper mainly discussed the process to classify Anthracnose and Downey Mildew, watermelon leaf diseases using neural network analysis. A few of infected leaf samples were collected and they were captured using a digital camera with specific calibration procedure under controlled environment. The classification on the watermelon's leaf diseases is based on color feature extraction from RGB color model where the RGB pixel color indices have been extracted from the identified Regions of Interest (ROI). The proposed automated classification model involved the process of diseases classification using Statistical Package for the Social Sciences (SPSS) and Neural Network Pattern Recognition Toolbox in MATLAB. Determinations in this work have shown that the type of leaf diseases achieved 75.9% of accuracy based on its RGB mean color component

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Traditional reasoning based on logical rules promotes the automation process of reasoning for pest diagnosis and control measures to a certain extent, but it experiences obvious defects, such as insufficient learning ability, low data utilization rate and inaccurate rate to be improved, which do not meet the requirements of practical application. Objects in fine-grained tasks typically share small inter-class variances and large intra-class variances, as well as multiple object scopes and complex backgrounds, which makes the problem more complicated. Considering the increasing data on diseases and insect pests, problems are inevitable, such as missing information and prolonged time. This cannot increase the complexity of the model. Compared with the traditional classification network, the method of generalization ability enhancement further enhances the identification accuracy. And the method needs less memory but can achieve low performance terminal real-time identification of various leaf diseases. And it can be applied in other crop disease identification fields with the similar application scenarios.

3.1.1 DISADVANTAGES

- It requires a large training data.
- So, this method is not effective for knowledge map with many relationships and sparse knowledge.

3.2 PROPOSED SYSTEM

Plants are susceptible to various disease-related disorders and seizures. There are various causes which can be characterized by their effect on plants, disturbances due to environmental conditions such as temperature, humidity, excessive or insufficient food, light and the most common diseases such as bacterial, viral and

fungal diseases. In the proposed system, we use the CNN algorithm to detect disease in plant leaves because with the help of CNN the maximum accuracy can be achieved if the data is good. To avoid over fitting, we phase wise divided the dataset into different training and testing ratios. In the case 80% of training and 20% of testing image data. System Proposed in the report can detect the different types of disease efficiently and have the ability to deal with complex scenarios. Validation results show the accuracy ,which depicts the feasibility of Convolution Neural Network and present the path for AI based Deep Learning Solution to this Complex Problem.

3.2.1 ADVANTAGE

- It is considered as the best DL technique for image classification due to high accuracy.
- In fine-grained disease categorization method based on attention network to solve the problem.
- With the growth of knowledge and the complexity of relationships, these methods are no longer the main knowledge representation methods because of their limited expressive ability and lack of flexibility.

3.3 FEASIBILITY STUDY

3.3.1 DATA WRANGLING

The data wrangling is checking for cleanliness, and then trimming and cleaning given dataset for analysis.

3.3.2 DATA COLLECTION

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The data

model which 14 was created using Random Forest, Linear Support Vector Machine and Decision Tree Classifier are applied on the Training set and based on the test result accuracy, Test set prediction is done. In this project, the train and test dataset ratio are 8:2.

3.3.3 PREPROCESSING

The dataset collected might contain missing values that may lead to inconsistency. To gain better results data need to be pre-processed to improve the efficiency of the algorithm. The outliers must be removed and also variable conversion need to be done.

3.4 PROJECT REQUIREMENTS

3.4.1 FUNCTIONAL REQUIREMENTS

INPUT: The major inputs for Integration of Web based Accommodation Upholding Maintenance System can be categorized module -wise. Basically all the information is managed by the software and in order to access the information one has to produce one's identity by entering the user-id and password. Every user has their own domain of access beyond which the access is dynamically refrained rather denied.

OUTPUT: The major outputs of the system are tables and reports. Tables are created dynamically to meet the requirements on demand. Reports, as it is obvious, carry the gist of the whole information that flows across the institution. This application must be able to produce output at different modules for different inputs.

3.4.3 GENERAL REQUIREMENTS

- The new system should be cost effective.
- To improve productivity and service and service.

- To enhance user interface.
- To improve information presentation and durability.
- To upgrade systems reliability, availability and flexibility.
- To address human factors for better and uses acceptance.

3.4.4 ENVIRONMENTAL REQUIREMENTS

HARDWARE REQUIREMENTS

- Processor – I3, I5, I7
- RAM - 6 Gb
- Hard Disk - 500 GB

SOFTWARE REQUIREMENTS

- Operating System - Windows 11
- Front End - Html, Cascading Style Sheets
- Scripts – python language
- Tool – Python IDE

CHAPTER 4

SYSTEM DESIGN

4.1 ER DIAGRAM

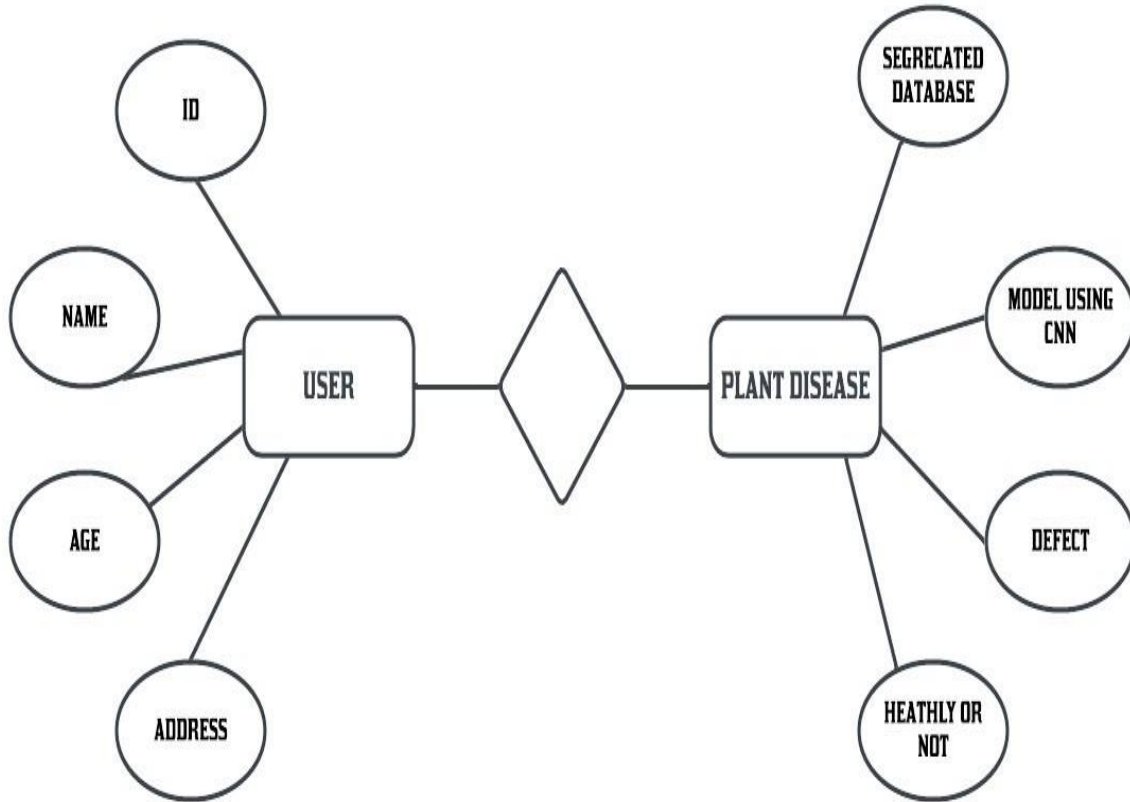


Fig 4.1 ER DIAGRAM

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

4.2 DATA FLOW DIAGRAM

LEVEL – 0:

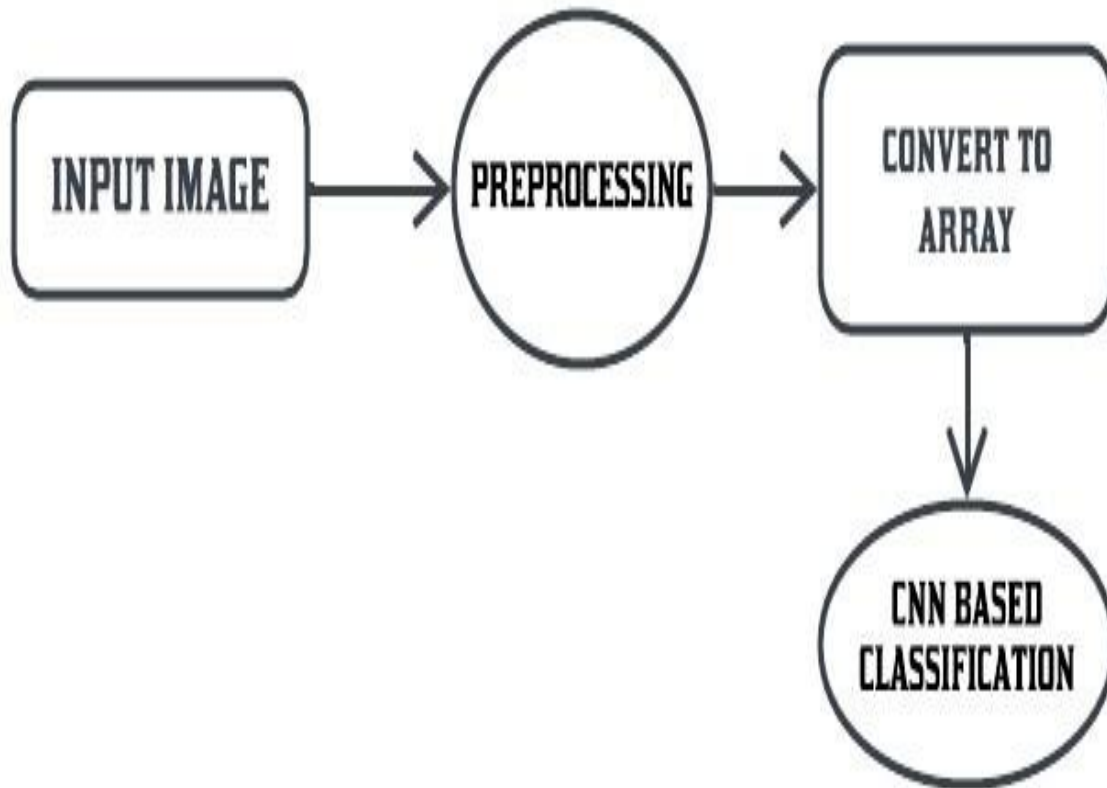


Fig 4.2.1 DFD LEVEL - 0 DIAGRAM

The Level 0 DFD diagram shows that the input will be given as values and images by the end user and the outputs are classified.

LEVEL - 1:

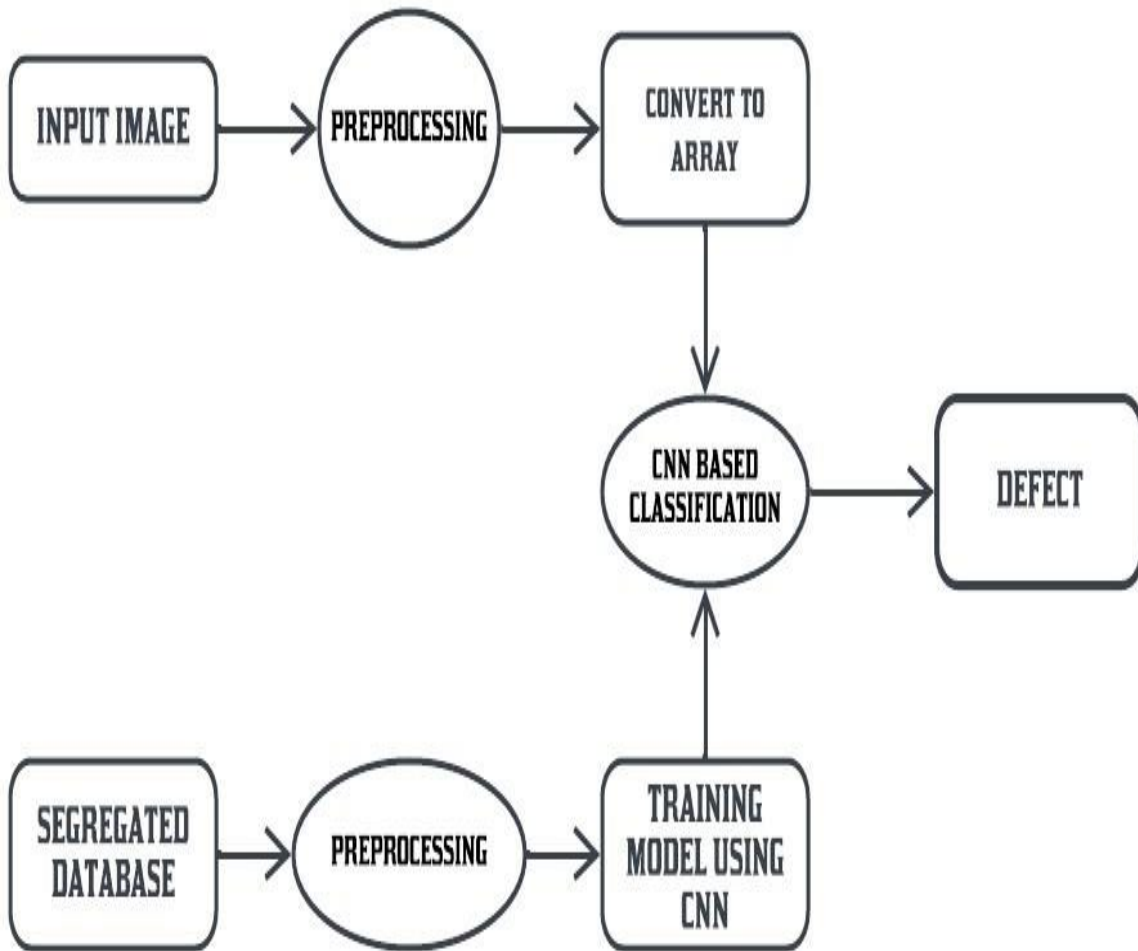


Fig 4.2.2 DFD LEVEL - 1 DIAGRAM

The below diagram shows the flow of data in this project starting from collection of data to deployment containing data preprocessing, cleaning and the building model for the project

LEVEL - 2:

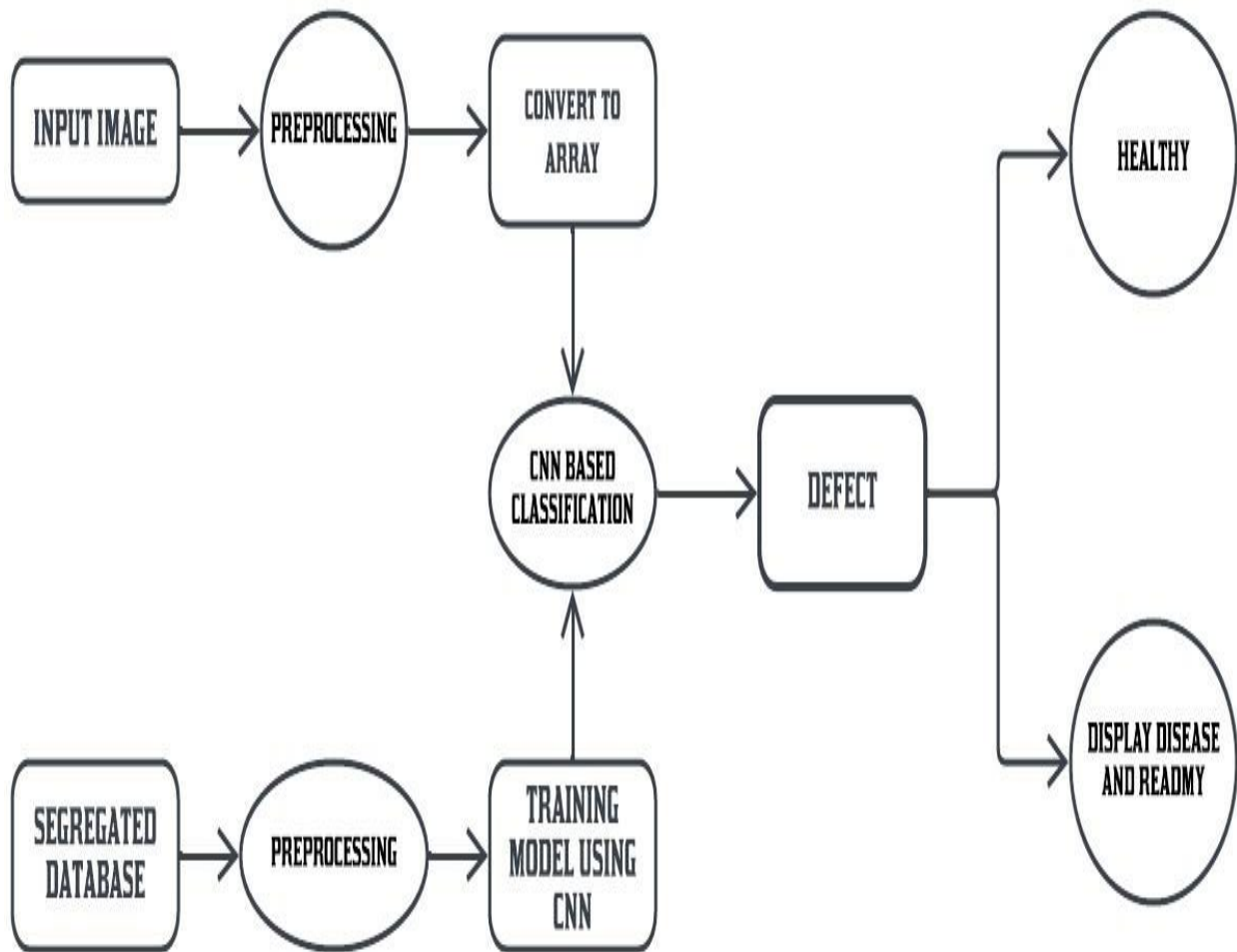


Fig 4.2.3 DFD LEVEL - 2 DIAGRAM

4.3 UML DIAGRAM

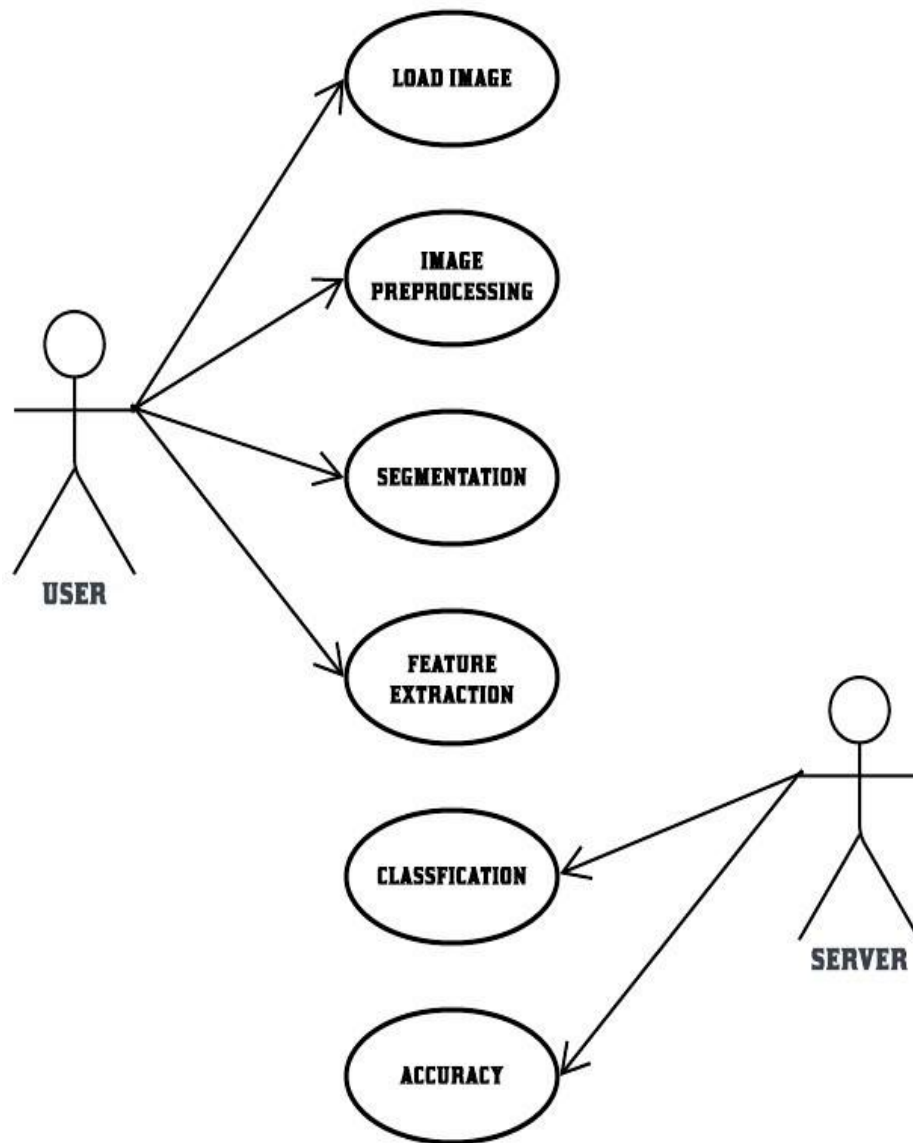


Fig 4.3 USE CASE DIAGRAM

The above use case diagram contains users like people who give the input as images, values and developer who fixes the bug. The developer also develops the software and trains and test the model using the algorithms.

4.4 CLASS DIAGRAM

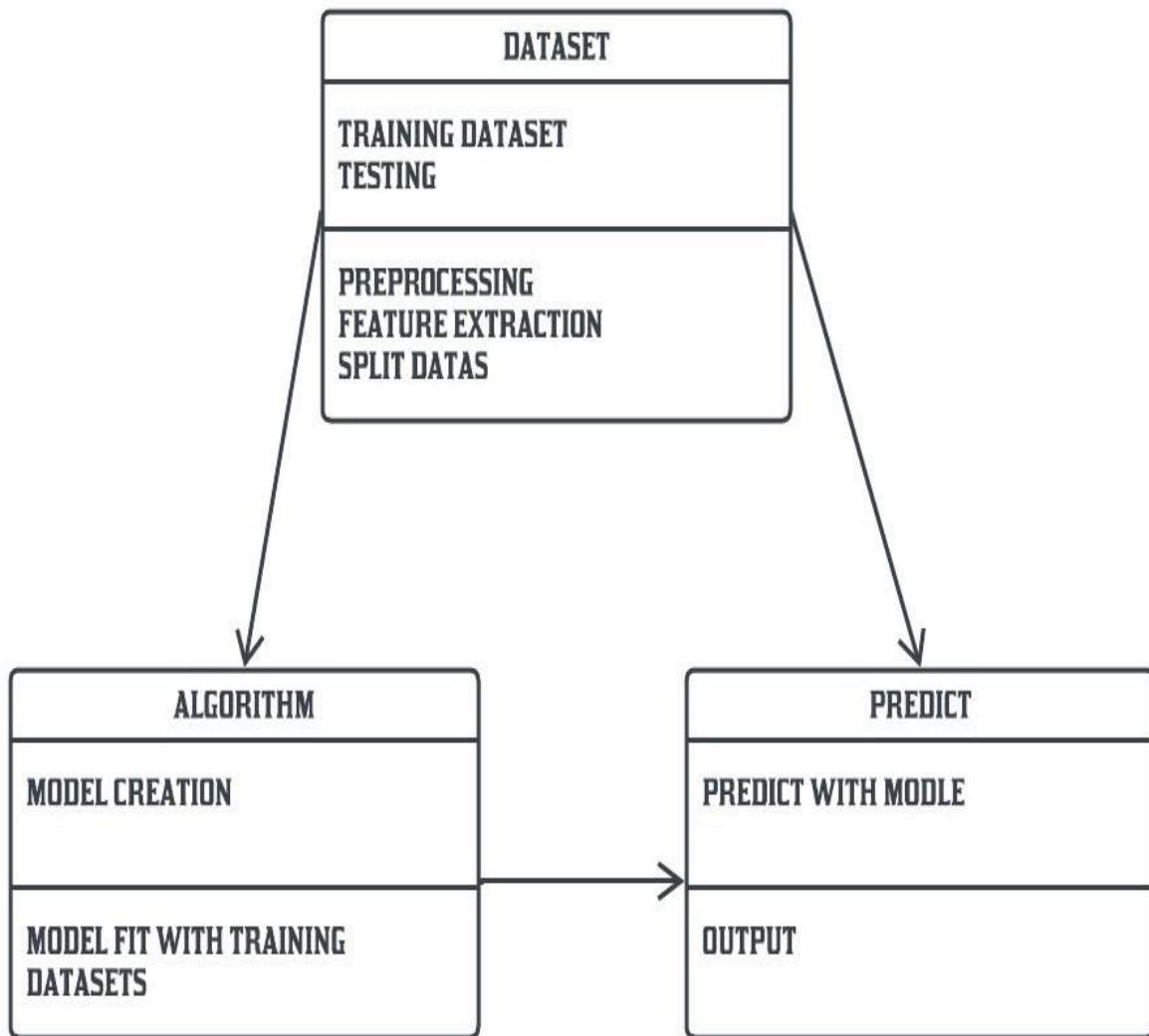


Fig 4.4 CLASS DIAGRAM

The above mentioned class diagram represents the detailed diagrammatic representation of the classes used. The dataset class is used for data preprocessing and data visualization.

4.5 ACTIVITY DIAGRAM

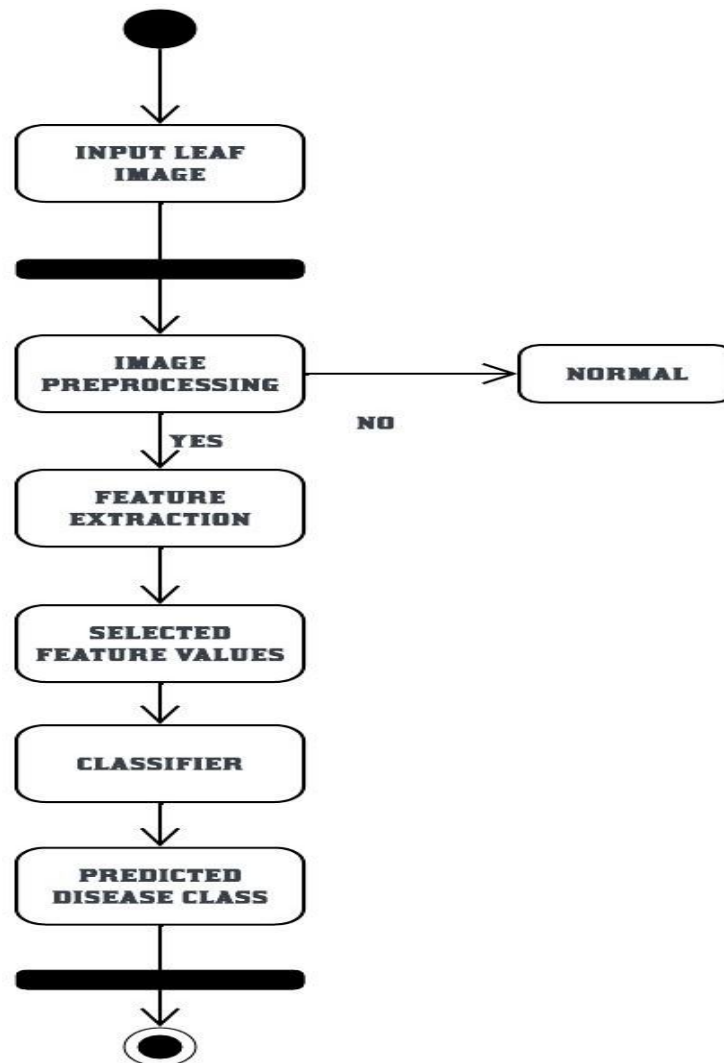


Fig 4.5 ACTIVITY DIAGRAM

The activity diagram represents the entire activities that are covered from the user giving the input to the software detecting the values, images and representing it as a output to the user. Initially the user gives the input as values to the software which is then used to detect the result based on the software developed by the developer and represented as the output to the user.

4.6 SEQUENCE DIAGRAM

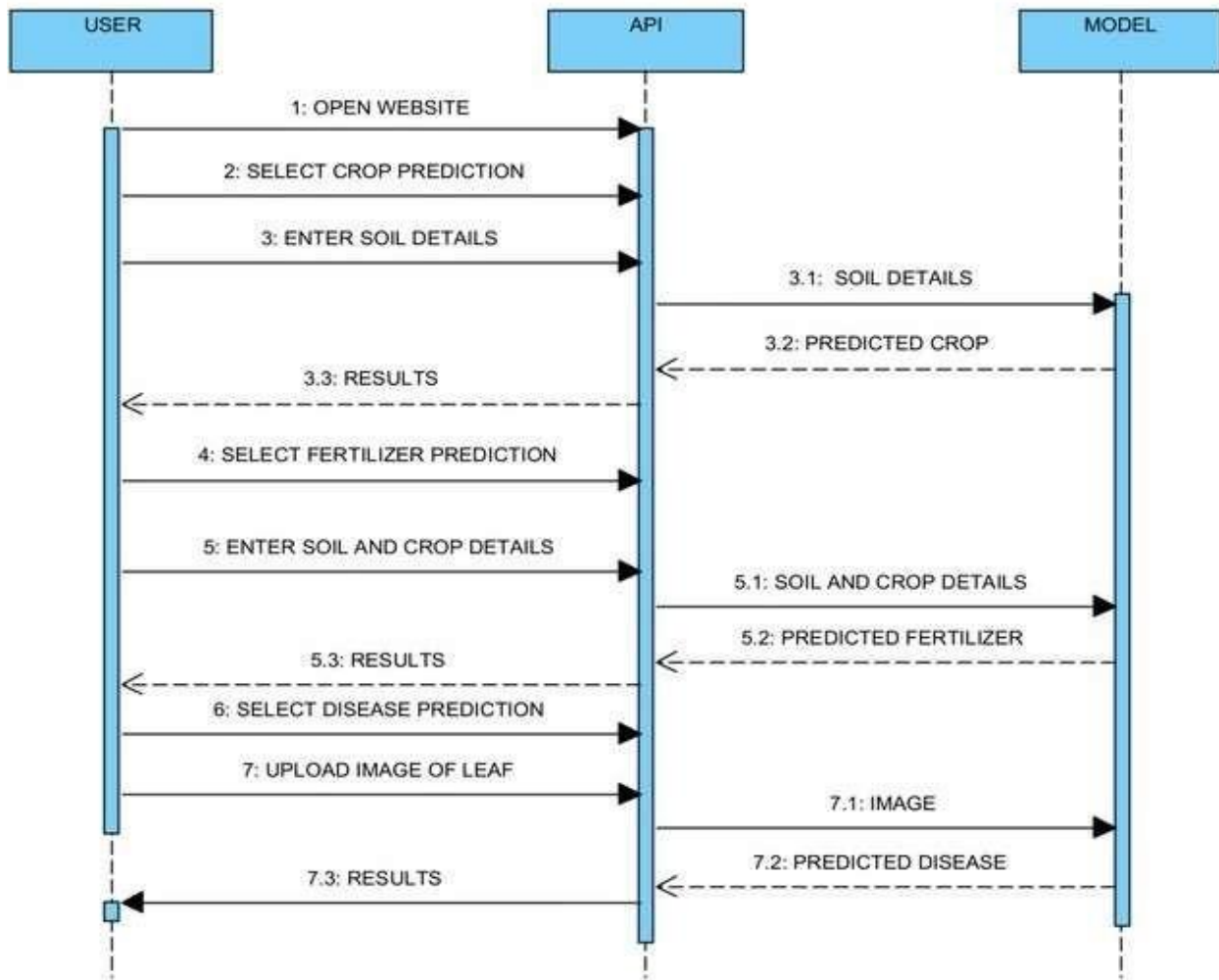


Fig 4.6 SEQUENCE DIAGRAM

The above diagram depicts all the sequence of activities in detecting the values and images from the input given by the user. Initially the data preprocessing on the dataset is done to get the required data and data visualization to represent the data in a understanding format. Then the developer uses this preprocessed data to train and test the model and finally finds the best model and deploys the software. It is then used by the users to give the input and detect the required result.

CHAPTER 5

SYSTEM

ARCHITECTURE

5 MODULE DESCRIPTIONS

5.1 SYSTEM ARCHITECTURE

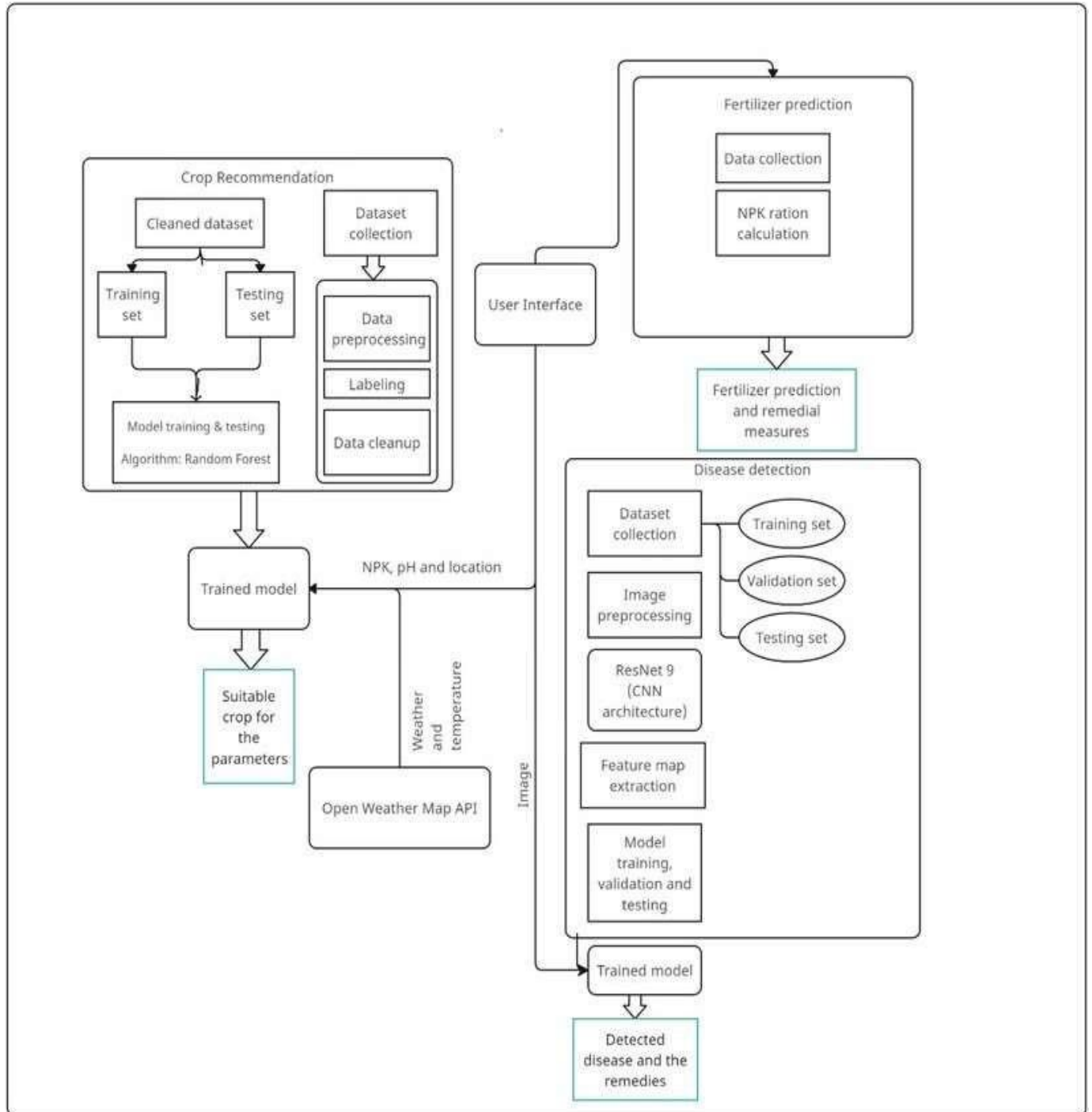


Fig 5.1 SYSTEM ARCHITECTURE DIAGRAM

5.1.1 ARCHITECTURE DESCRIPTION

The architecture diagram of “AGRI-COMPANION USING MACHINE LEARNING” includes the Prediction of Crop and Fertilizer along with disease detection and prevention measures prediction. The first step involves the collection of data from various soil and labeling them into classes for the prediction process. After labeling, preprocessing of data has to be done in order to remove noisy and irrelevant data. Doing this step will reduce overfitting of the model and hence after training the model, the results will be accurate for both the test data and the unseen user data. The temperature and weather values for the user’s location will be fetched by an open-source API called Open Weather Map API which has an immense amount of the daily weather forecast data. For the fertilizer prediction module, the NPK values are fetched from the user and the ratios are calculated and the remedial measures are suggested. To find the disease in plants, a photo has to be uploaded. On uploading the image the data will be processed and the image classification is done by the trained model. And the prediction of the right Leaf disease is displayed to the farmer. ResNet9 was used for disease detection. If the disease predicted was harmful then the subsequent preventive methods for eradicating are also suggested. The UI for each of the functionality can be used by the user to provide inputs and using those inputs the model will predict the results.

5.2 LIST OF MODULES

- Dataset Collection
- Data Preprocessing
- Feature Extraction
- Model Selection and Metrics
- Configuration of the Classification Model
- Crop recommendation and Detection of Leaf Disease

5.2.1 DATASET COLLECTION

Data is a crucial part of any Machine Learning System. Datasets from various government websites and Kaggle were used to predict the crops. The dataset for the crops recommendation system consists of 22 crops grown across India. Parameters considered for the crop recommendation model are nitrogen, phosphorus, potassium, rainfall, temperature, humidity; ph. Dataset for crop disease classification consists of images of leaves of 14 plants while excluding healthy leaves, 26 types of images that show a particular disease in a plant. For each plant disease type, there are 1800 images.

5.2.2 DATA PREPROCESSING

- Labels of each plant images are then also mapped to a unique the pre-processing step for any machine learning model is of great importance and ideally shapes the performance and results of the models chosen. In this report, the following were the steps that were carried out in order to make sure that the models produced optimal results:
- After reading and resizing the images, we then convert the images into an array form using `np.array()`
- The value using `Label Binarizer()`
- Finally, the plant village dataset is split into two different sets, namely, train and test set with a 75:25 ratio respectively.

5.2.3 FEATURE EXTRACTION

Feature extraction is a process in machine learning where relevant information is extracted from raw data to create a more meaningful and simplified representation of the data that can be easily processed by a learning algorithm. For the plant leaf disease image dataset, feature extraction involves extracting features such as color,

texture and shape of the leaves. This can be done using image processing techniques such as edge detection, color histograms and CNN to identify and extract these features. These features are then fed into a machine learning algorithm for classification or prediction of the type of plant disease present. On the other hand, for the crop recommendation CSV dataset, feature extraction involves extracting features such as Nitrogen, Potassium, phosphorus values, climate, pH level and geographical location. These features are then used to make recommendations on the best crops to grow in a particular region, considering the environmental factors that affect crop growth.

5.2.4 MODEL SELECTION AND METRICS

It involves selecting the appropriate algorithm and architecture to use in the model, as well as tuning the hyper parameters to achieve the best performance. For crop recommendation using CSV dataset, one approach could be to use supervised learning algorithms such as Decision trees, Random forests, Naive Bayes, Logistic regression and Support Vector Machines (SVMs). These algorithms can learn from the input data and generate a decision tree or rule-based model that can be used to make recommendations based on the input features. We can find the accuracy metric of each model. For plant leaf disease detection using image dataset, deep learning architectures such as CNN are commonly used. CNN are particularly suited to image data as they can automatically extract relevant features from images without the need for manual feature engineering. Transfer learning, where a pre-trained CNN model is fine-tuned for the specific task of plant leaf disease detection, can also be used to improve performance with limited data.

5.2.5 CONFIGURATION OF THE CLASSIFICATION MODEL

In this model the first block contains a Convolutional layer with 32 filters of size 3 x 3 and the activation function used was the ReLU activation function. We then

follow the operation by performing batch normalization and choosing the Max Pooling layer with a pool size of and adding a dropout layer with 25% drop-out.

Batch normalization was performed in order to speed up the convergence of the neural network. It is generally applied after each individual layer so that the output of the previous layer can be normalized allowing for each individual layer present in the network to perform learning independent. Dropout layer is a technique used to prevent the model from over fitting by randomly switching off some sections of the neurons. When some sections of the neurons are switched off the incoming as well as the outgoing connections from the neurons are also switched off and this results in the betterment of the model in learning and allows for the model to not generalize to the test dataset. We used the pre trained model for leaf disease prediction.

5.2.6 CROP RECOMMENDATION AND DETECTION OF LEAF DISEASE

Pre-trained CNN models can be used to make predictions and detect plant leaf diseases by extracting relevant features from the images and classifying the type of disease present. This approach can improve the accuracy of the predictions and reduce the need for manual feature engineering. However, it requires a large amount of data to fine-tune the pre-trained model for the specific task. Finally, classifiers are used for the training and testing of the datasets. These classifiers may be Random Forest, Support Vector Machine (SVM), Naïve Bayes, Neural Network, Logistic Regression, Decision Tree and Artificial Neural Network based etc. These methods are used to classify and detect the leaf Diseases, Crop, Fertilizer. The user must give an input image of the infected plant to the ML model and the model classifies the image explaining why the disease has occurred to the plant. It also suggests remedies to cure the plant.

5.3 PROPOSED SYSTEM

5.3.1 RANDOM FOREST ALGORITHM

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.

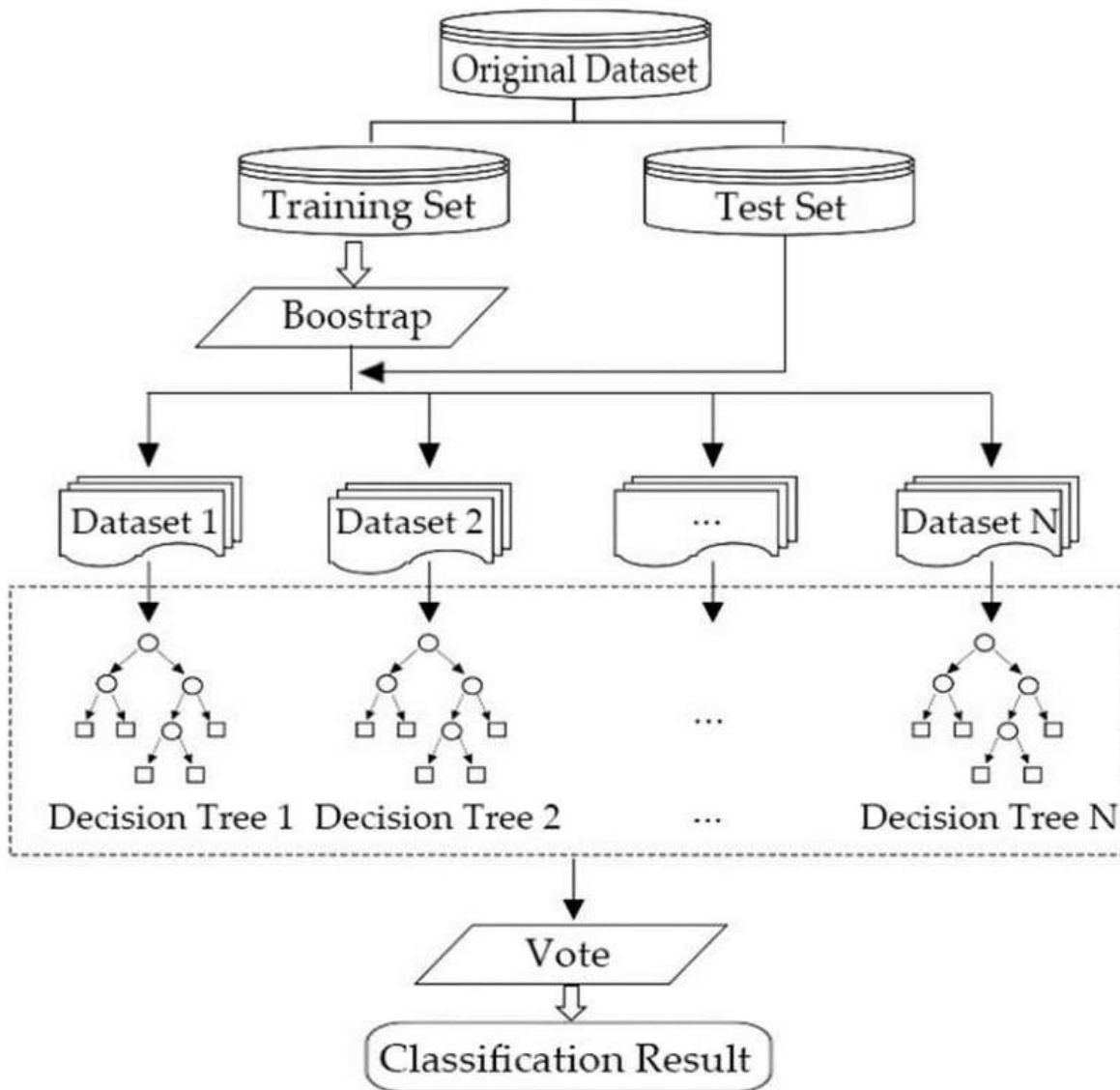


Fig 5.3.1 WORK FLOW DIAGRAM FOR PROPOSED SYSTEM

ASSUMPTIONS:

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

WORKING:

Random Forest works in two-phase first is to create the random forest by combining N decision tree and second is to make predictions for each tree created in the first phase. The working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

PSEUDOCODE:

1. Split the Dementia dataset into training and testing sets.
2. Initialize an empty list to store the decision trees.
3. For $i = 1$ to $n_estimators$ (number of trees):
 - a. Sample a random subset of the training data (with replacement).
 - b. Grow a decision tree using the sampled data.
 - c. Add the decision tree to the list of decision trees.

4. For each test instance:

a. Make a prediction by aggregating the predictions of all decision trees in the forest.

5. Evaluate the performance of the model on the testing set using a performance metric such as accuracy, AUC, etc.

Overall, Random Forest algorithm combines multiple decision trees to improve the predictive performance and reduce over fitting. By randomly sampling the training data and features, Random Forest creates a diverse set of decision trees that capture different aspects of the data. By aggregating the predictions of all decision trees, Random Forest provides a robust and accurate prediction.

ADVANTAGES:

- Random Forest can perform both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the over fitting issue.

5.3.2 CONVOLUTIONAL NEURAL NETWORKS (CNN)

Image classification involves the extraction of features from the image to observe some patterns in the dataset. Using an ANN for the purpose of image classification would end up being very costly in terms of computation since the trainable parameters become extremely large.

EXAMPLE: If we have a 50 X 50 image of a cat and we want to train our traditional ANN on that image to classify it into a dog or a cat the trainable parameters become —
 $(50 \times 50) \times 100$ image pixels multiplied by hidden layer + 100 bias + 2×100 output neurons + 2 bias = 2,50,30

STEP 1: Choose a Dataset.

Choose a dataset of your interest or you can also create your own image dataset for solving your own image classification problem. An easy place to choose a dataset is on kaggle.com. The dataset I'm going with can be found [here](#).

This dataset contains 12,500 augmented images of blood cells (JPEG) with accompanying cell type labels (CSV). There are approximately 3,000 images for each of 4 different cell types grouped into 4 different folders (according to cell type). The cell types are Eosinophil, Lymphocyte, Monocyte, and Neutrophil. Here are all the libraries that we would require and the code for importing them.

STEP 2: Prepare Dataset for Training.

Preparing our dataset for training will involve assigning paths and creating categories (labels), resizing our images. Resizing images into 200 X 200.

STEP 3: Create Training Data.

Training is an array that will contain image pixel values and the index at which the image in the categories list.

STEP 4: Shuffle the Dataset.

STEP 5: Assigning Labels and Features.

This shape of both the lists will be used in classification using the neural networks.

STEP 6: Normalizing X and converting labels to categorical data.

STEP 7: Split X and Y for use in CNN.

STEP 8: Define, compile and train the CNN Model.

STEP 9: Accuracy and Score of models.

```

function XCOMPRESSCU(*pCurCU) {CALCULATION}

M  $\leftarrow$  FastCUMope(PO,QP)

if M  $\geq$  4 SPLIT then

C2n  $\leftarrow$  CHECKINTRA(pCurCU)

else

C2n  $\leftarrow \infty$ 

end if

if M  $\neq$  HOMO and Dcur < Dmax then

Cn  $\leftarrow$  0

for i = 0 to 3 do

pSubCUi  $\leftarrow$  pointer to SubCUi

CN  $\leftarrow$  CN+ XCompressCU(pSubCUi)

end for

else

CN  $\leftarrow \infty$ 

end if

CHECKBESTMODE(C2N, CN)

end function

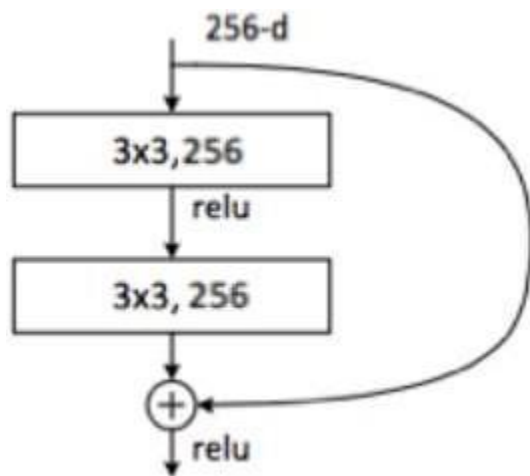
```

5.3.3 RESIDUAL NETWORKS (ResNet) IN KERAS

Very deep neural networks are hard to train as they are more prone to vanishing or exploding gradients. To solve this problem, the activation unit from a layer could be fed directly to a deeper layer of the network, which is termed as a **Skip Connection**. This forms the basis of **Residual Networks** or **ResNets**. This post will introduce the basics the residual networks before implementing one in Keras.

RESIDUAL BLOCK:

A building block of a ResNet is called a **Residual Block** or **Identity Block**. A residual block is simply when the activation of a layer is fast-forwarded to a deeper layer in the neural network.



As you can see in the image above, the activation from a previous layer is being added to the activation of a deeper layer in the network. This simple tweak allows training much deeper neural networks.

In theory, the training error should monotonically decrease as more layers are added to a neural network. In practice however, for a traditional neural network, it will reach a point where the training error will start increasing. ResNets do not suffer

from this problem. The training error will keep decreasing as more layers are added to the network. In fact, ResNets have made it possible to train networks with more than 100 layers, even reaching 1000 layers.

BUILDING A ResNet FOR IMAGE CLASSIFICATION:

Keras is a high-level neural networks API, written in Python and capable of running on top of [TensorFlow](#), [CNTK](#), or [Theano](#). It was developed with a focus on enabling fast experimentation. Now, let's build a ResNet with 50 layers for image classification using [Keras](#). In this case, we will use TensorFlow as the backend. Of course, feel free to grab the entire [notebook](#) and make all the necessary imports before starting.

STEP 1: Define the identity block.

STEP 2: Convolution block.

STEP 3: Build the model.

STEP 4: Training.

STEP 5: Print the model summary.

Algorithm 2. Pseudo code of the used preprocessing method.

Input: The raw 1D sensor signal (S) with size of 5625

Output: Graylevel image (Im) with size of 125 x 45

```
1: count = 1;
2: for  $i=1$  to 125 do
3:   for  $j=1$  to 45 do
4:      $Im(i, j) = S(count)$ ;
5:      $count = count + 1$ ;
6:   end for  $j$ 
7: end for  $i$ 
8: Normalize  $Im$  by using min-max normalization.
```

5.3.4 VGG16 IMPLEMENTATION IN KERAS

VGG16 is a CNN architecture which was used to win ILSVR (Image net) competition in 2014. It is considered to be one of the excellent vision model architecture till date. Most unique thing about VGG16 is that instead of having a large number of hyper-parameter they focused on having convolution layers of 3x3 filters with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC (fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx) parameters.

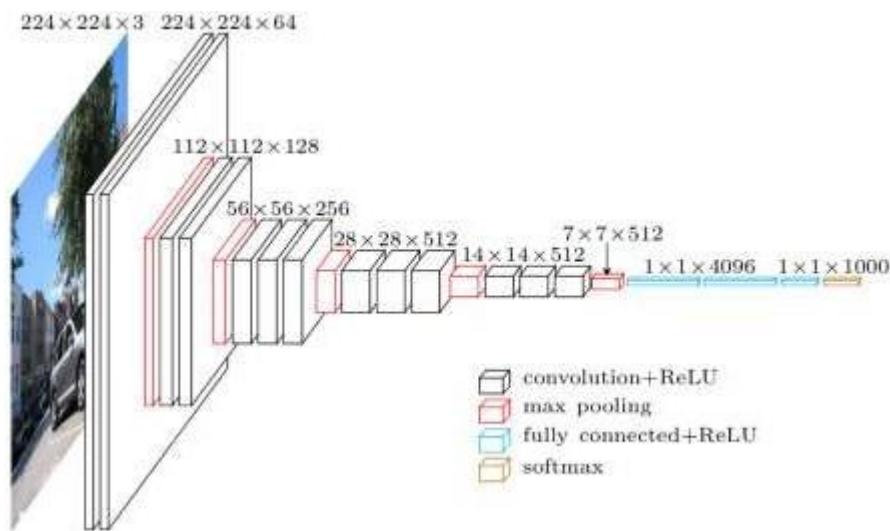


Fig 5.3.4 RESIDUAL NETWORKS 9 LAYERS

Here I first import all the libraries which i will need to implement VGG16. I will be using Sequential method as I am creating a sequential model (Sequential model means that all the layers of the model will be arranged in sequence). Here I have imported Image Data Generator from keras preprocessing. The objective of Image Data Generator is to import data with labels easily into the model. It is a very useful

class as it has many functions to rescale, rotate, zoom, flip etc. The most useful thing about this class is that it doesn't affect the data stored on the disk. This class alters the data on the go while passing it to the model.

```

function CONVOP( $i, IBuf_x$ )
   $C_o \leftarrow 0$ 
  while  $C_o < O$  do
    if  $C_o = 0$  and  $i = 0$  then
       $c \leftarrow 0$ 
      READBIAS()
      READKERNEL( $C_o \sim C_o + 31, K_c$ )
    end if
    || PPMACCONV( $K_c, IBuf_x$ )
    || PREFETCHKERNEL( $C_o + 32 \sim C_o + 63, K_c$ )
     $c \leftarrow \bar{c}$ 
  end while
end function

function CONV( $T_x, T_y, C_i$ )
   $T_x \leftarrow 0, T_y \leftarrow 0, C_i \leftarrow 0$ 
  while  $T_y < Y$  do
    while  $T_x < X$  do
      while  $C_i < I$  do
        if  $T_x = 0$  and  $T_y = 0$  then
           $c \leftarrow 0$ 
          READTILE( $IBuf_c, T_x, T_y, C_i$ )
        end if
        || CONVOP( $C_i, IBuf_c$ )
        || PREFETCHTILE( $IBuf_c, T_x, T_y, C_i + 1$ )
         $c \leftarrow \bar{c}$ 
      end while
    end while
  end while
end function

```

CHAPTER 6

SYSTEM TESTING

6.1 INTRODUCTION

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive. A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

6.2 UNIT TESTING

The various modules of the system were developed and tested individually after the development of each unit. Each form was designed and each API in the flask app was tested after the integration with the UI. Each of the models was trained separately.

- i. The landing page is tested first and it loads without any errors while clicking the link.
- ii. The crop recommendation module was developed, both front end and back end and the unit test cases were written and tested for the module.
- iii. When the user enters the values in the form in the crop recommendation form, the API is hit, the prediction is made by the model and the results are displayed successfully. If in case of any error, the try again page is displayed.
- iv. For the fertililzer recommendation, given the values in the form for the fertilzer prediction api, the ratios are calculated and the result is displayed in the appropriate API.
- v. For the disease detection, the image is uploaded and the results are displayed which was verified and tested.

6.2.1 BASIC PATH TESTING

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

- Use the design of the code and draw correspondent flow graph.
- Determine the Cyclomatic complexity of resultant flow graph, using formula:

$$V(G) = E - N + 2 \text{ or } V(G) = P + 1 \text{ or } V(G) = \text{Number of Regions}$$

Where $V(G)$ is Cyclomatic complexity,

E is the number of edges,

N is the number of flow graph nodes,

P is the number of predicate nodes.

- Determine the basis of set of linearly independent paths.

6.3 INTEGRATION TESTING

After the development and the testing phase of each of the modules has been completed, all the units were integrated into a single module. After the training and the testing of each mode in the unit testing phase, the pth and the pickle files were used in the flask app file to integrate all the functionalities. The flask app was developed and all the functionalities were tested.

6.4 TEST CASES AND REPORTS

TEST CASE ID	TEST CASE / ACTION TO BE PERFORMED	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1	Display the Home page by clicking on the Website Link	Display the Information and Features of the website	Display the various services of Agri-Companion.	Pass
2	Display the various services cards in the home page	Display the various services and the explanation when hovered	Displays the card for each service when user hovers over them	Pass
3	Selecting Crop button in home page	Display the crop prediction form	Scrolls down and displays the form for crop prediction	Pass
4	Display the various input fields for user to enter values	Show various fields for each input value	Gets the input value from the user	Pass
5	View the various states drop down menu	View the various state list in the form	Lists the various states for the user to choose from	Pass

6	Selecting predict button in crop recommendation	Display the crop that is recommended for that soil	Displays the crop name	Pass
7	Selecting the Fertilizer button in home page	Display the Fertilizer that is recommended for that soil	Displays the Fertilizer page	Pass
8	Selecting predict button in fertilizer recommendation	Display the information on how to improve the soil	Display the information on how to improve the soil	Pass
9	Selecting the Disease button in home page	Display the page for uploading the image	Displays the form to upload the image	Pass
10	Drag and drop functionality to upload the image	Allow user to select an image and drag and drop it in the form	The user is able to upload the images via drag and drop option	Pass
11	Clicking the Predict button in the Disease page	Display information about the disease, cause of disease and the preventive measures	Scrolls down and shows the information about the Diseases.	Pass

TABLE 6.1 TEST CASES AND REPORTS

CHAPTER 7

PERFORMANCE

EVALUATION

7.1 RESULT AND DISCUSSION

This project aimed at improving crop yield and productivity. The system uses data from various sources, such as weather conditions, soil properties and historical crop data, to predict crop growth, disease occurrence and recommend the best fertilizer and pesticides to use. The results of this project can be significant, as it provides farmers with accurate and reliable information to optimize their crops and increase their yields. The use of machine learning technology in agriculture can also help reduce the environmental impact of farming, by enabling farmers to use fewer resources while producing more food.

7.2 PERFORMANCE METRICS

- Performance is measured in terms of reports generated weekly and monthly.
- For the crop prediction model using Random Forest, the calculated accuracy is 98.409%.
- The confusion matrix for the crop detection model has been given below.

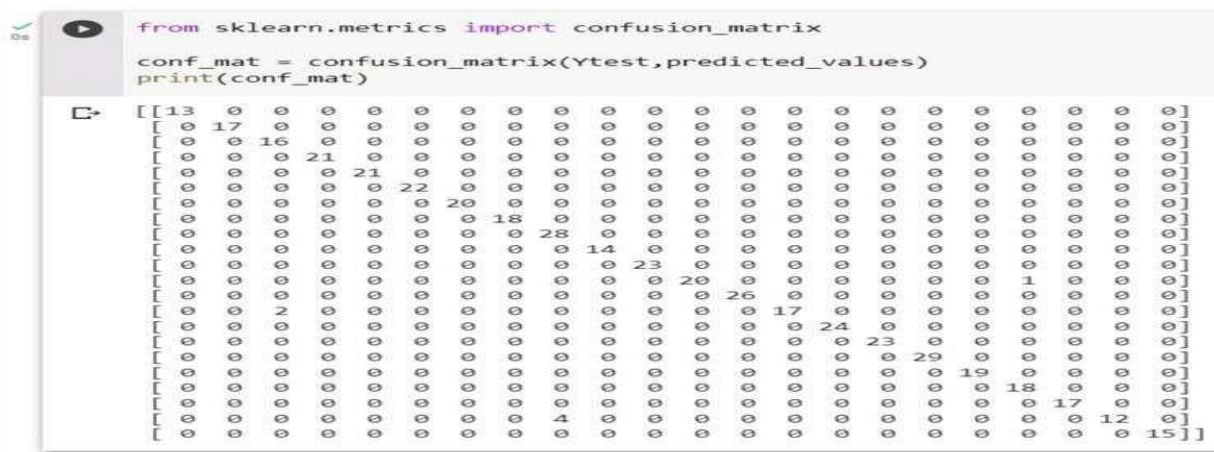


Fig. 8.1 Confusion Matrix for crop detection model

Accuracy = $\frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{False Positives} + \text{True Negatives} + \text{False Negatives}}$

$$\text{Accuracy} = \frac{13+17+16+21+21+22+20+18+28+14+23+20+26+17+24+23+29+19+18+17+12+15}{13+17+16+21+21+22+20+18+28+14+23+20+26+17+24+23+29+19+18+17+12+15+1+2+4}$$

$$= 433/440$$

$$= 0.98409090909$$

- Therefore, the accuracy achieved is 98.4% for the crop prediction model.
- The f1 scores of each of the classes has also been given below.

```
print(classification_report(Ytest,predicted_values))
```

RF's Accuracy is: 0.9840909090909091				
	precision	recall	f1-score	support
apple	1.00	1.00	1.00	13
banana	1.00	1.00	1.00	17
blackgram	0.89	1.00	0.94	16
chickpea	1.00	1.00	1.00	21
coconut	1.00	1.00	1.00	21
coffee	1.00	1.00	1.00	22
cotton	1.00	1.00	1.00	20
grapes	1.00	1.00	1.00	18
jute	0.88	1.00	0.93	28
kidneybeans	1.00	1.00	1.00	14
lentil	1.00	1.00	1.00	23
maize	1.00	0.95	0.98	21
mango	1.00	1.00	1.00	26
mothbeans	1.00	0.89	0.94	19
mungbean	1.00	1.00	1.00	24
muskmelon	1.00	1.00	1.00	23
orange	1.00	1.00	1.00	29
papaya	1.00	1.00	1.00	19
pigeonpeas	0.95	1.00	0.97	18
pomegranate	1.00	1.00	1.00	17
rice	1.00	0.75	0.86	16
watermelon	1.00	1.00	1.00	15
accuracy			0.98	440
macro avg	0.99	0.98	0.98	440
weighted avg	0.99	0.98	0.98	440

Fig.8.2 f1 scores for crop prediction model

- For the disease detection model the model's accuracy has been validated using the validation set for each class. An accuracy of 99.2% has been achieved in the last epoch.

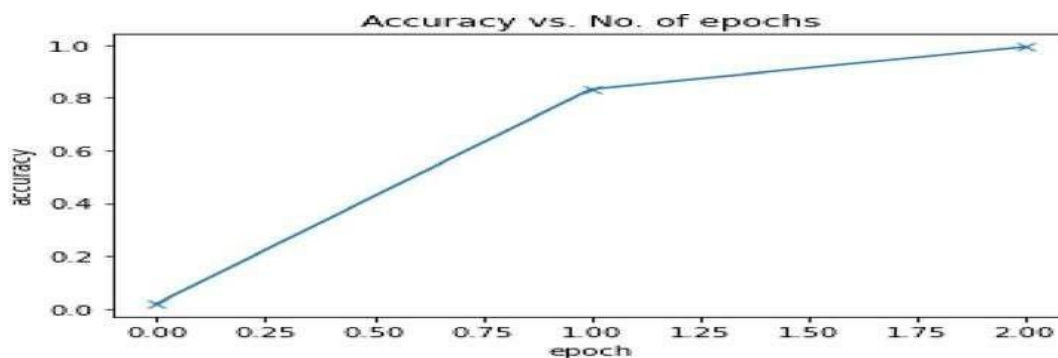


Fig. 8.3 Accuracy vs. No. of epochs

Epoch [0],
last_lr: 0.00812,
train_loss: 0.7466,
val_loss: 0.5865,
val_acc: 0.8319

Epoch [1],
last_lr: 0.00000,
train_loss: 0.1248,
val_loss: 0.0269,
val_acc: 0.9923

- A loss of 2.69 has been reported for the validation set.

- The learning rate has also been visualized below.

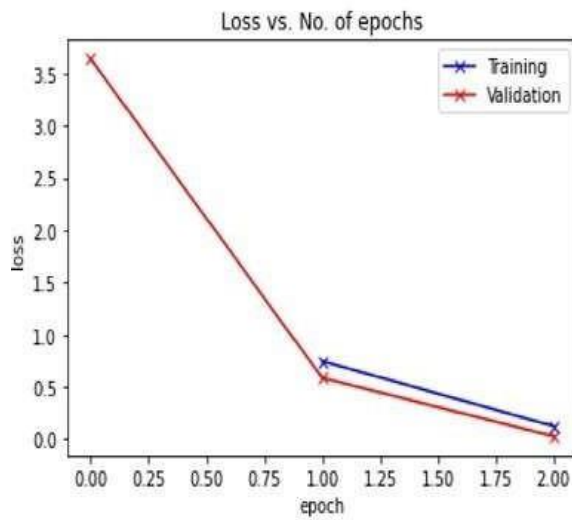


Fig. 8.4 Loss vs. No. of epochs

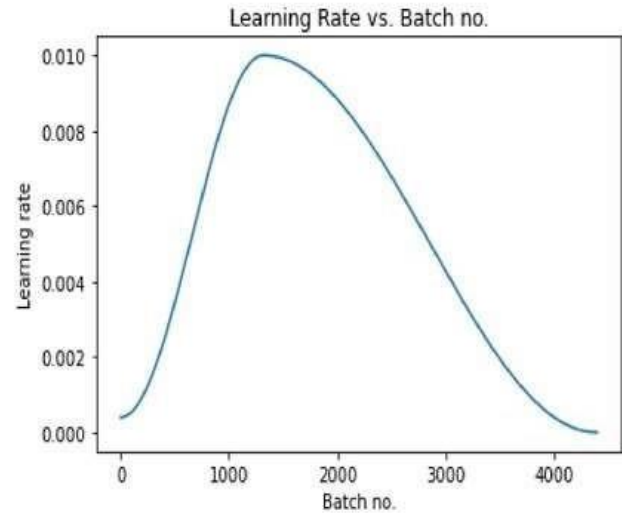


Fig. 8.5 Learning Rate vs. Batch No.

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

Protecting crops in organic farming is not an easy task. This depends on a thorough knowledge of the crop being grown and possible pests, pathogens and weeds. In our system, a special deep learning model has been developed based on a special architectural convolution network to detect plant diseases through images of healthy or diseased plant leaves. The system described above can be upgraded to a real-time video entry system that allows unattended plant care. Another aspect that can be added to certain systems is an intelligent system that cures identified ailments. Studies show that managing plant diseases can help increase yields by about 50%.

8.2 FUTURE WORKS

Due to the lack of enough Random Access Memory (RAM) storage the study had to be limited to only 10,000 images. In the future, it would be great to test the implementation of both the CNN and KNN model and also use LIME on the whole plant village dataset containing multiple different plants in order to bring detection and explain ability to a wide variety of plants.

Another work that this study would like to pursue in the future is to provide a comparative study on different XAI techniques and implement a user study in order to find out which XAI technique provides the best explain ability, transparency and interpretability. With the addition of data on Volatile organic compounds, soil types, environmental conditions and time of the month as mentioned by farmers through feedback from the user study, the user trust of the detection tool is expected to grow a little higher. As discussed earlier in the use case of this study, a working application that is capable of taking pictures of plants and detecting plant diseases in real-time is the ideal goal and will prove to be of great use to the farmers and botany enthusiasts.

APPENDICES

A1. SAMPLE DATASET

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	N	P	K		temperat	humidity	ph		rainfall	label																			
2	91	37	39	20.87974	82.00274	6.502905	202.9355	rice																					
3	98	36	44	21.77046	80.32964	7.038096	226.6355	rice																					
4	63	57	39	23.00446	82.32076	7.840207	263.9642	rice																					
5	89	53	44	26.4911	80.13836	6.900401	242.864	rice																					
6	89	49	37	20.13017	81.60487	7.628479	262.7179	rice																					
7	86	40	41	23.05805	83.37012	7.073404	251.055	rice																					
8	60	42	39	22.78884	82.67941	5.790806	271.3249	rice																					
9	97	54	43	20.27774	82.89409	5.718627	241.9742	rice																					
10	76	56	37	24.31388	83.53522	6.685346	230.4462	rice																					
11	95	55	35	23.22397	83.03323	6.336254	221.2092	rice																					
12	68	57	36	26.52724	81.41754	5.386168	264.6349	rice																					
13	66	58	36	23.97898	81.45062	7.502834	250.0832	rice																					
14	73	48	44	26.8008	80.8885	5.108882	284.4365	rice																					
15	66	44	35	24.01498	82.05687	6.984354	185.2779	rice																					
16	65	40	42	25.60385	80.66385	6.94802	209.587	rice																					
17	61	37	38	24.28309	80.90326	7.042299	231.0883	rice																					
18	75	37	39	23.58712	82.78837	6.349051	276.6352	rice																					
19	86	43	43	23.79392	80.43818	6.97086	206.2612	rice																					
20	61	56	40	21.86325	80.1823	5.933933	234.335	rice																					
21	87	53	39	23.57944	83.38376	5.853932	291.2967	rice																					
22	81	40	37	23.32504	80.47476	6.443475	185.4875	rice																					
23	100	35	44	25.15786	83.11713	5.070176	231.3843	rice																					
24	93	36	38	21.94767	80.97384	6.013633	213.3561	rice																					
25	87	54	43	21.05254	82.6784	6.254028	233.1076	rice																					
26	89	36	39	23.48381	81.33265	7.375483	224.0581	rice																					
27	97	42	36	25.07564	80.52389	7.778915	257.0039	rice																					
28	61	35	39	26.33927	84.04404	6.2885	271.3586	rice																					
29	82	37	42	24.52923	80.54899	7.07096	260.2634	rice																					
30	83	55	38	20.77576	84.49774	6.244841	240.0811	rice																					
31	90	41	40	22.30157	80.66436	6.043305	197.9791	rice																					
32	86	60	41	21.44854	84.94376	5.624709	272.2017	rice																					
33	78	46	41	22.17912	80.33127	6.357389	200.0883	rice																					
34	96	60	44	24.52784	82.79886	6.364135	224.6757	rice																					
35	95	39	43	20.26708	81.63895	5.014507	270.4417	rice																					
36	64	39	45	25.70543	83.88266	6.149411	233.1321	rice																					
37	100	48	43	26.79534	82.34809	5.950601	193.3474	rice																					
38	80	53	36	26.75754	81.17734	5.96037	272.2999	rice																					
39	80	36	44	23.8633	83.15251	5.561399	205.2494	rice																					
40	83	50	43	21.01945	82.95222	7.418245	298.4018	rice																					

B2. SAMPLE CODING

CLIENT SIDE CODING

```
{% extends 'layout.html' %}
{% block body %}
<section class="banner_w3lspvt" id="home">
    <div class="csslider infinity" id="slider1">
        <div class="banner-top">
            <div class="overlay">
                <div class="container">
                    <div class="w3layouts-banner-info text-center">
                        <h3 class="text-wh">Agri Companion</h3>
                        <h4 class="text-wh mx-auto my-4"><b>Get informed
decisions about your farming strategy.</b></h4>
                        <br>
                        <h4 class="text-wh mx-auto my-4"><strong> Here are
some queries we answer</strong></h4>
                        <p class="text-li mx-auto mt-2">
                            1. What crop to plant here? <br>
                            2. What fertilizer to use? <br>
                            3. Which disease do your crop have?<br>
                            4. How to cure the disease?</p>
                    </div> </div></div></div></div>
                </div>
            </div>
        </div>
    </div>
</section>
<section class="blog py-5", style="height:700px">
    <div class="container py-md-5">
        <h3 class="heading mb-sm-5 mb-4 text-center"> Our Services</h3>
```

```
<div class="row blog-grids">
```

```
<div class="col-lg-4 col-md-6 blog-left mb-lg-0 mb-sm-5 pb-lg-0 pb-5">
```

```
<div class="cardmain">
```

```
<div class="align">
```

```
<span class="red"></span>
```

```
<span class="yellow"></span>
```

```
<span class="green"></span>
```

```
</div>
```

```
<h1 style="color:#d9dedb">CROP</h1>
```

```
<p class="mt-2", style="color:#d9dedb">
```

Recommendation of crops that can be cultivated which is best suited for the respective conditions. </p>

```
</div></div>
```

```
<div class="col-lg-4 col-md-6 blog-left mb-lg-0 mb-sm-5 pb-lg-0 pb-5">
```

```
<div class="cardmain">
```

```
<div class="align">
```

```
<span class="red"></span>
```

```
<span class="yellow"></span>
```

```
<span class="green"></span>
```

```
</div>
```

```
<h1 style="color:#d9dedb">FERTILIZER</h1>
```

```
<p class="mt-2", style="color:#d9dedb">
```

Recommendation about the type of fertilizer best suited for the particular soil and the recommended crop. </p> </div></div>

```
<div class="col-lg-4 col-md-6 blog-left mb-lg-0 mb-sm-5 pb-lg-0 pb-5">
```

```
<div class="cardmain">
```

```
<div class="align">
```

```
<span class="red"></span>
```

```
<span class="yellow"></span>
```

```
<span class="green"></span>
```

```
</div>
```



```

        <h1 style="color:#d9dedb">DISEASE</h1>
        <p class="mt-2", style="color:#d9dedb">Predicting the name
and causes of crop disease and suggestions to cure it.</p>
    </div></div></div></div>

</section>
<!-- //Products & Services -->
<!-- Creating custom grid and hover effect
<section>
<div class="col-lg-3 col-md-4 col-sm-6 col-xs-12">
    <div class="hovereffect">
        
        <div class="overlay">
            <h2>Hover effect 1</h2>
            <a class="info" href="#">link here</a>
        </div>
    </div>
</div>
</div> -->
</html>
{ % endblock % }

```

SERVER SIDE CODING

```

from flask import Flask, render_template, request, Markup
import numpy as np
import pandas as pd
from utils.disease import disease_dic
from utils.fertilizer import fertilizer_dic
import requests
import config
import pickle

```

```

import io
import torch
from torchvision import transforms
from PIL import Image
from utils.model import ResNet9
import os

disease_model_path = 'D:/templates_for_project/new_models/plant-disease-model.pth'
disease_model = ResNet9(3, len(disease_classes))
disease_model.load_state_dict(torch.load(
    disease_model_path, map_location=torch.device('cpu')))
disease_model.eval()

crop_recommendation_model_path = 'D:/templates_for_project/models/RandomForest.pkl'
crop_recommendation_model = pickle.load(
    open(crop_recommendation_model_path, 'rb'))

def weather_fetch(city_name):

    api_key = config.weather_api_key
    base_url = "http://api.openweathermap.org/data/2.5/weather?"

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

app = Flask(__name__)
@ app.route('/')
def home():
    title = 'Agri Companion - Home'
    return render_template('index.html', title=title)
@ app.route('/crop-recommend')
def crop_recommend():

```

```

    title = 'Agri Companion - Crop Recommendation'
    return render_template('crop.html', title=title)
@app.route('/fertilizer')
def fertilizer_recommendation():
    title = 'Agri Companion - Fertilizer Suggestion'
    return render_template('fertilizer.html', title=title)

@app.route('/crop-predict', methods=['POST'])
def crop_prediction():
    title = 'Agri Companion - Crop Recommendation'
    if request.method == 'POST':
        N = int(request.form['nitrogen'])
        P = int(request.form['phosphorous'])
        K = int(request.form['pottasium'])
        ph = float(request.form['ph'])
        rainfall = float(request.form['rainfall'])

        city = request.form.get("city")
        if weather_fetch(city) != None:
            temperature, humidity = weather_fetch(city)
            data = np.array([[N, P, K, temperature, humidity, ph, rainfall]])
            my_prediction = crop_recommendation_model.predict(data)
            final_prediction = my_prediction[0]
            return render_template('crop-result.html', prediction=final_prediction, title=title)

        else:
            return render_template('try_again.html', title=title)

if (__name__ == '__main__'):
    port=int(os.environ.get("port",500))
    app.run(host='0.0.0.0',port=port)

```

MODELS CODING

```
from flask import Flask, render_template, request, Markup
```

```
import numpy as np
```

```
import pandas as pd
```

```
from utils.disease import disease_dic
```

```
from utils.fertilizer import fertilizer_dic
```

```
import requests
```

```
import config
```

```
import pickle
```

```
import io
```

```
import torch
```

```
from torchvision import transforms
```

```
from PIL import Image
```

```
from utils.model import ResNet9
```

```
import os
```

LOADING THE TRAINED MODELS

```
disease_classes=
```

```
['Apple___Apple_scab','Apple___Black_rot','Apple___Cedar_apple_rust',  
'Apple___healthy','Blueberry___healthy','Cherry_(including_sour)___Powdery_mi  
ldew','Cherry_(including_sour)___healthy','Corn_(maize)___Cercospora_leaf_spot  
Gray_leaf_spot','Corn_(maize)___Common_rust_', 'Corn_(maize)___Northern_Lea
```

```
f_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
Twospotted_spider_mite','Tomato___Target_Spot','Tomato___Tomato_Yellow_L
eaf_Curl_Virus','Tomato___Tomato_mosaic_virus','Tomato___healthy']
```

```
disease_model_path = 'models/plant_disease_model.pth'
```

```
disease_model = ResNet9(3, len(disease_classes))
```

```
disease_model.load_state_dict(torch.load(
```

```
disease_model_path, map_location=torch.device('cpu')))
```

```
disease_model.eval()
```

LOADING CROP RECOMMENDATION MODEL

```
crop_recommendation_model_path = 'models/RandomForest.pkl'
```

```
crop_recommendation_model=pickle.load
```

```
(open(crop_recommendation_model_path, 'rb'))
```

```
def weather_fetch(city_name):"""
```

```
return (temperature, humidity):"""
```

```
api_key = config.weather_api_key
```

```

base_url = "http://api.openweathermap.org/data/2.5/weather?"

complete_url = base_url + "appid=" + api_key + "&q=" + city_name

response = requests.get(complete_url)

x = response.json()

    if x["cod"] != "404":

        y = x["main"]

        temperature = round((y["temp"] - 273.15), 2)

        humidity = y["humidity"]

        return temperature, humidity

    else:

        return None

def predict_image(img, model=disease_model):"""

return prediction (string):"""

transform =

transforms.Compose([transforms.Resize(256),transforms.ToTensor(),])

image = Image.open(io.BytesIO(img))

img_t = transform(image)

img_u = torch.unsqueeze(img_t, 0)

```

GET PREDICTIONS FROM MODEL

```

yb = model(img_u)_, preds = torch.max(yb, dim=1)

```

```
prediction = disease_classes[preds[0].item()]
```

```
return prediction
```

FLASK APP

```
app = Flask(__name__)@ app.route('/')
```

```
def home()
```

```
title = 'Harvestify - Home'
```

```
return render_template('index.html', title=title)
```

RENDER CROP RECOMMEDATION FORM PAGE

```
@ app.route('/crop-recommend')
```

```
def crop_recommend()
```

```
title = 'Harvestify - Crop Recommendation'
```

```
return render_template('crop.html', title=title)
```

RENDER FERTILIZER RECOMMEDATION FORM PAGE

```
@ app.route('/fertilizer')
```

```
def fertilizer_recommendation()
```

```
title = 'Harvestify - Fertilizer Suggestion'
```

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

RENDER CROP RECOMMEDATION RESULT PAGE

```
@ app.route('/crop-predict', methods=['POST'])
```

```

def crop_prediction()

title = 'Harvestify - Crop Recommendation'

if request.method == 'POST':

    N = int(request.form['nitrogen'])

    P = int(request.form['phosphorous'])

    K = int(request.form['pottasium'])

    ph = float(request.form['ph'])

    rainfall = float(request.form['rainfall'])

    state = request.form.get("stt")

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

if weather_fetch(city) != None:

    temperature, humidity = weather_fetch(city)

    data = np.array([[N, P, K, temperature, humidity, ph, rainfall]])

    my_prediction = crop_recommendation_model.predict(data)

    final_prediction = my_prediction[0]

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

else:

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

RENDER FERTILIZER RECOMMEDATION RESULT PAGE

@app.route('/fertilizer-predict', methods=['POST'])

```



```

def fert_recommend()

title = 'Harvestify - Fertilizer Suggestion'

crop_name = str(request.form['cropname'])

    N = int(request.form['nitrogen'])

    P = int(request.form['phosphorous'])

    K = int(request.form['pottasium'])

    ph = float(request.form['ph'])

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

nr = df[df['Crop'] == crop_name]['N'].iloc[0]

pr = df[df['Crop'] == crop_name]['P'].iloc[0]

kr = df[df['Crop'] == crop_name]['K'].iloc[0]

n = nr - N

p = pr - P

k = kr - K

temp = {abs(n): "N", abs(p): "P", abs(k): "K"}

max_value = temp[max(temp.keys())]

if max_value == "N":

    if n < 0:

        key = 'NHigh'

    else:

```

```

        key = "Nlow"

    elif max_value == "P":

    if p < 0:

        key = 'PHigh'

    else:

        key = "Plow"

    else:

        if k < 0:

            key = 'KHigh'

        else:

            key = "Klow"

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

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

```

RENDER FERTILIZER RECOMMEDATION RESULT PAGE

```

@app.route('/disease-predict', methods=['GET', 'POST'])

def disease_prediction()

    title = 'Harvestify - Disease Detection'

    if request.method == 'POST':

        if 'file' not in request.files:

```

```

return redirect(request.url)

file = request.files.get('file')

if not file:

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

try:

    img = file.read()

    prediction = predict_image(img)

    prediction = Markup(str(disease_dic[prediction]))

    return render_template('disease-result.html', prediction=prediction, title=title)

except:

    pass

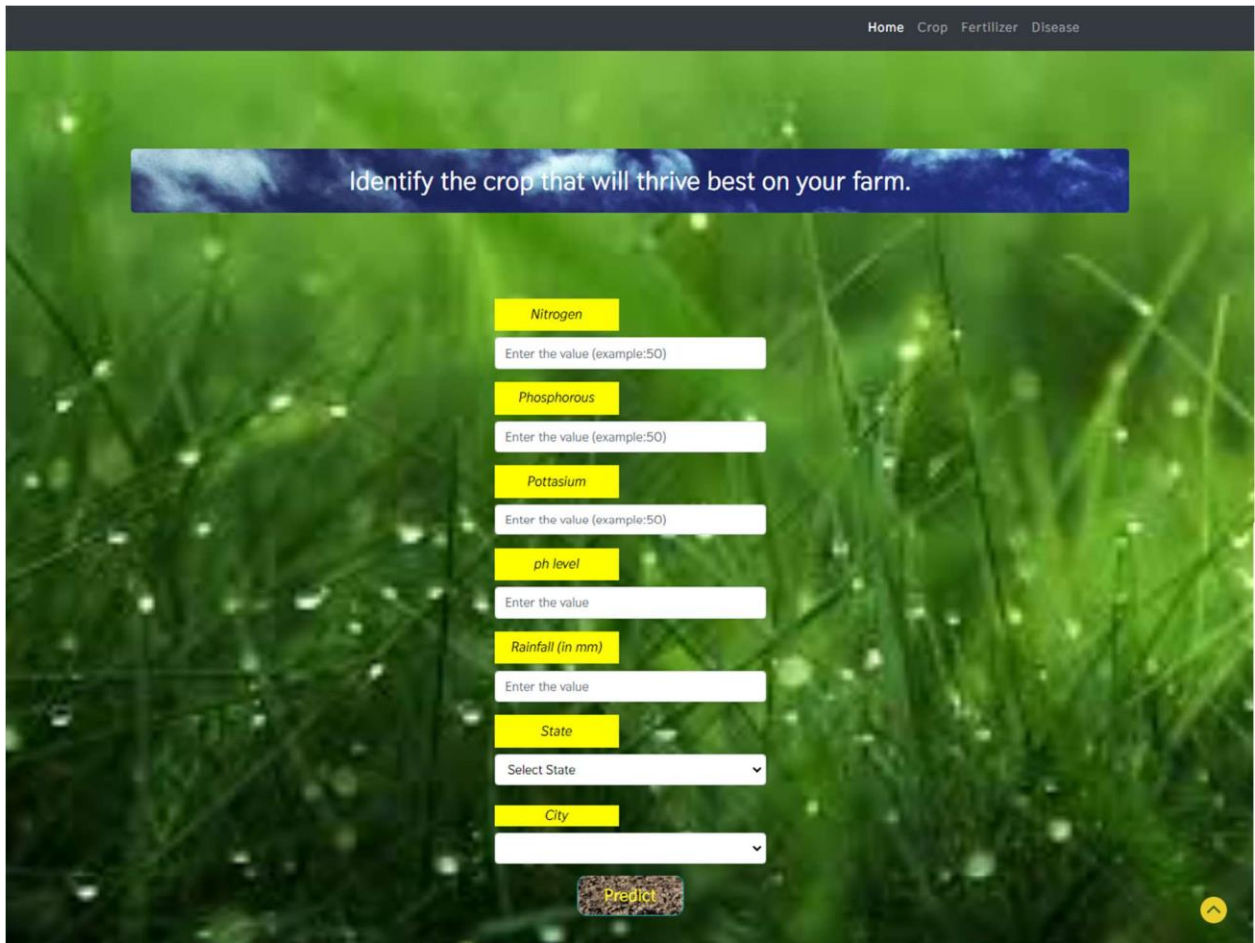
    return render_template('disease.html', title=title)

if (__name__ == '__main__'):port=int(os.environ.get("port",500))

app.run(host='0.0.0.0',port=port)

```

C3. SAMPLE SCREENS



Home Crop Fertilizer Disease

Identify the crop that will thrive best on your farm.

Nitrogen
Enter the value (example:50)

Phosphorous
Enter the value (example:50)

Pottasium
Enter the value (example:50)

ph level
Enter the value

Rainfall (in mm)
Enter the value

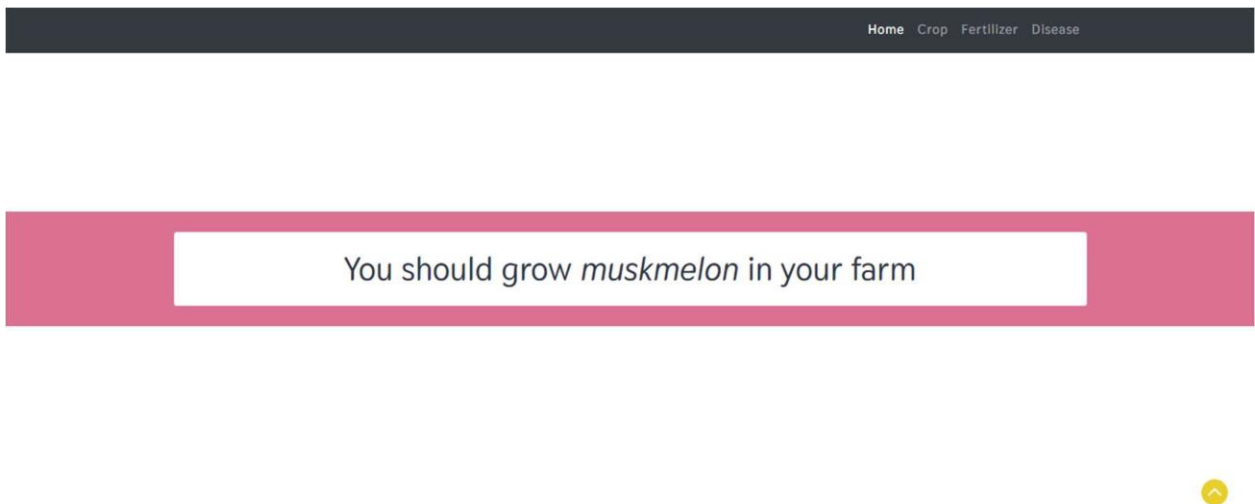
State
Select State

City

Predict

Up arrow icon

Fig C3.1 SCREENSHOT OF CROP MODULE



Home Crop Fertilizer Disease

You should grow *muskmelon* in your farm

Up arrow icon

Fig C3.2 RESULT OF CROP MODULE

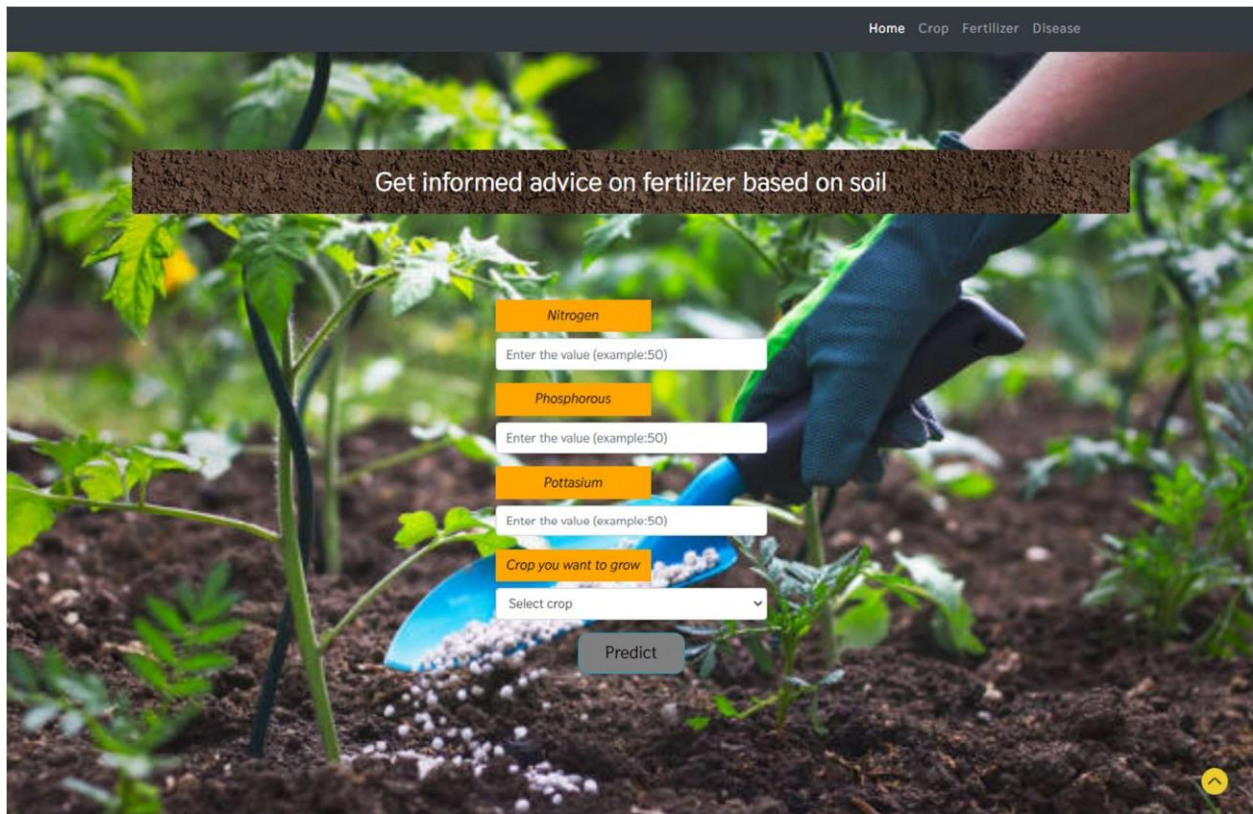


Fig C3.3 SCREENSHOT OF FERTILIZER MODULE

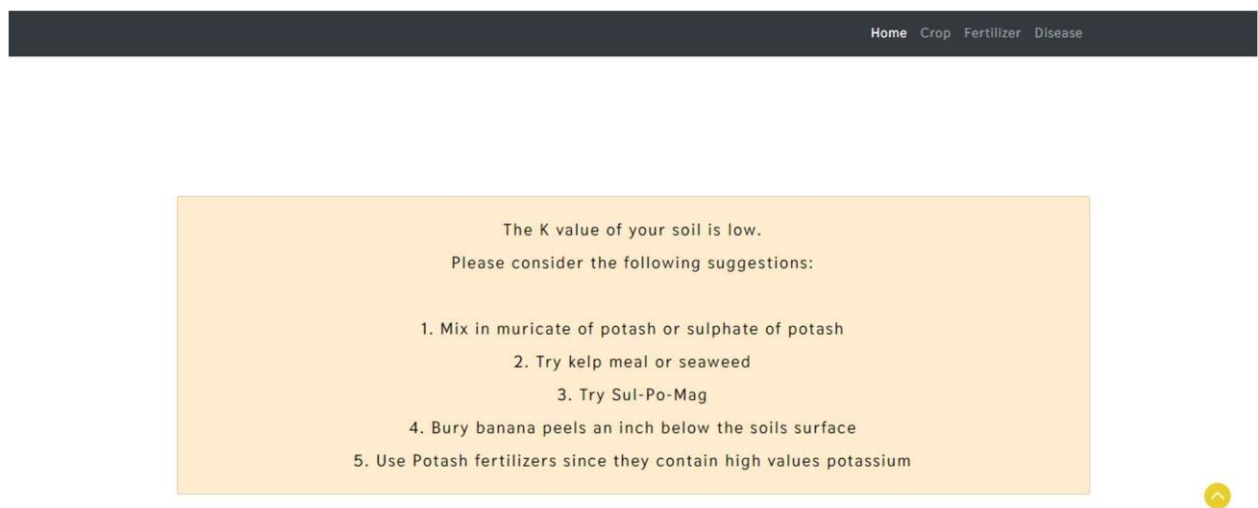


Fig C3.4 RESULT OF FERTILIZER MODULE

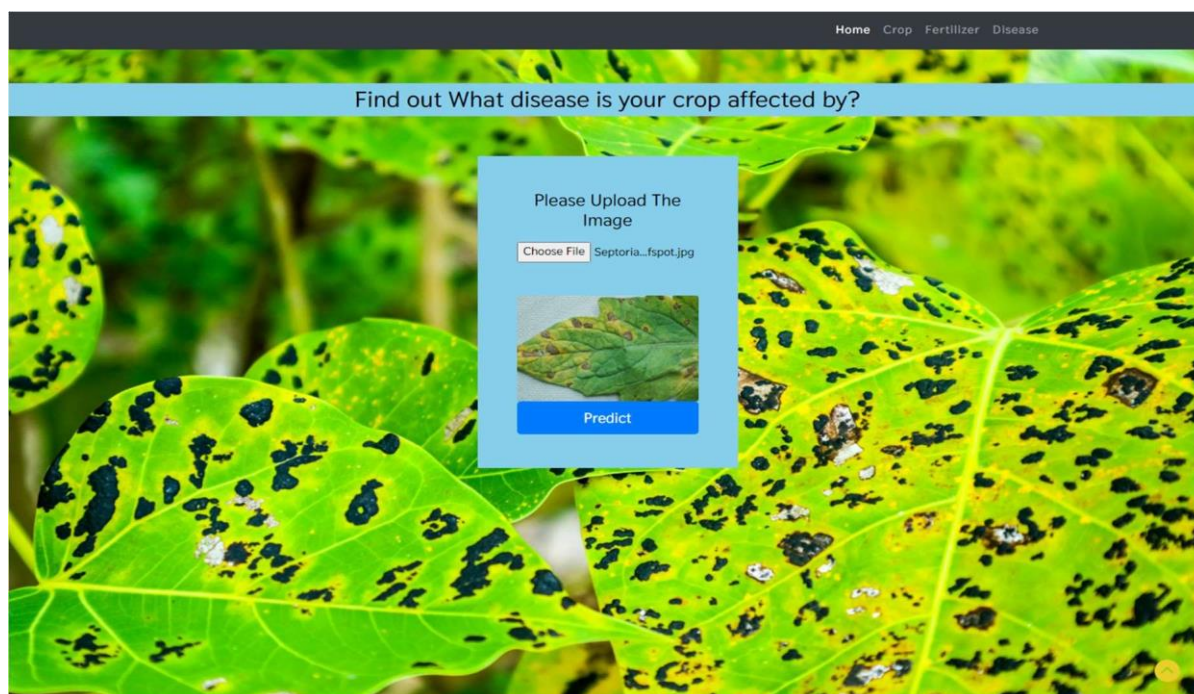


Fig C3.5 SCREENSHOT OF LEAF DISEASE MODULE

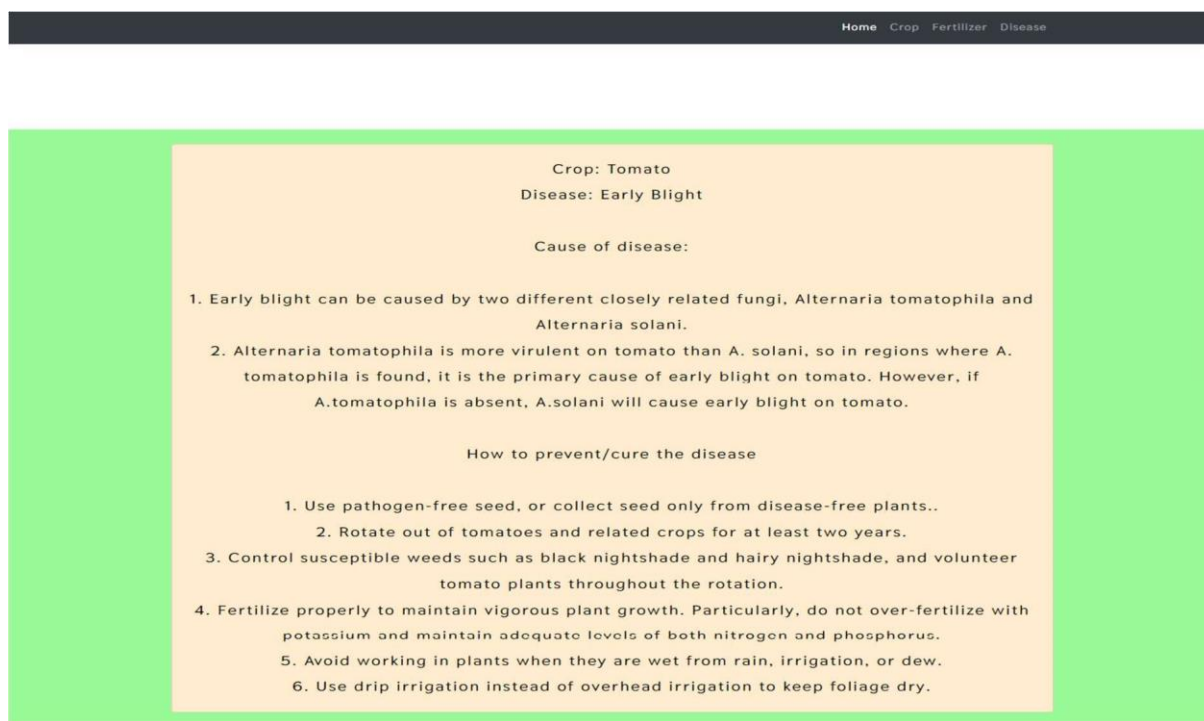


Fig C3.6 RESULT OF LEAF DISEASE MODULE

REFERENCES

- [1] Taranjeet Singh, Saurabh Anand, Anmol Sehgal, Siddhesh Mahajan, Prof. Pranoti Kavimandan, (2022) 'CROFED Crop and Fertilizer Recommendation and Disease diagnosis system using Machine Learning and Internet of Things', Published in, Volume 09, pp. 2349-6002.
- [2] P. Sahu, A. P. Singh, A. Chug and D. Singh, (2022) 'A Systematic Literature Review of Machine Learning Techniques Deployed in Agriculture: A Case Study of Banana Crop' in IEEE Access, vol. 10, pp. 87333-87360,
- [3] S. D. Khan and H. Ullah, “A survey of advances in vision-based vehicle re-identification,” Comput. Vis. Image Understand., vol. 182, pp. 50–63, May 2019.
- [4] A. A. Joshi and B. D. Jadhav, (2016), 'Monitoring and controlling rice diseases using Image processing techniques' , in International Conference on Computing, Analytics and Security Trends (CAST), pp. 471-476.
- [5] D. Elavarasan and P. M. D. Vincent, (2020), 'Crop Yield Prediction Using Deep Reinforcement Learning Model for Sustainable Agrarian Applications', in IEEE Access, vol. 8, pp. 86886-8690.
- [6] P. Rubini and P. Kavitha, 'Deep Learning model for early prediction of plant disease', 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), Tirunelveli, India, 2021, pp. 1104-1107.

- [7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Boston, MA, USA, Jun. 2020, pp. 1–9.
- [8] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, Jun. 2020, pp. 770–778.
- [9] T. Islam, T. A. Chisty and A. Chakrabarty, (2018), 'A Deep Neural Network Approach for Crop Selection and Yield Prediction in Bangladesh', 2018 IEEE Region 10 Humanitarian Technology Conference (RIO-HTC), pp. 1-6,
- [10] Y. M. Fernandez-Ordonez and J. Soria-Ruiz, (2017), 'Maize crop yield estimation with remote sensing and empirical models', in IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pp. 3035-3038.