# FERTILIZERS RECOMMMENDATION FOR DISEASE PREDICTION

**IBM PROJECT REPORT**

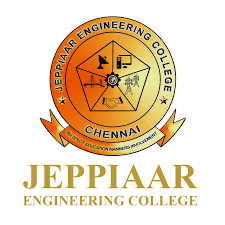
SUBMITTED BY

**TEAM ID:PNT2022TMID09649**

HIMANSHU ROY (310819104037)

VIKRAM M (310819104092)

LESSLIE JOHN D (310819104046)

MOTCHA ENOK C (310819104056)

**JEPPIAAR ENGINEERING COLLEGE**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE**

**CHENNAI.**

**NOV 2022**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TABLE OF CONTENTS** | **PAGE NO** |

|  |  |  |
| --- | --- | --- |
| **1.** | **INTRODUCTION** | **3** |
|  | **1.1 Project Overview** | **3** |
|  | **1.2 Purpose** | **4** |
| **2.** | **LITERATURE SURVEY** | **5** |
|  | **2.1 Existing Problem** | **5** |
|  | **2.2 References** | **7** |
|  | **2.3 Problem Statement Definition** | **10** |
| **3.** | **IDEATION & PROPOSED SOLUTION** | **11** |
|  | **3.1 Empathy Map Canvas** | **11** |
|  | **3.2 Ideation & Brainstorming** | **12** |
|  | **3.3 Proposed Solution** | **16** |
|  | **3.4 Problem Solution Fit** | **18** |
| **4.** | **REQUIREMENT ANALYSIS** | **19** |
|  | **4.1 Functional Requirement** | **19** |
|  | **4.2 Non-Functional Requirement** | **20** |
| **5.** | **PROJECT DESIGN** | **21** |
|  | **5.1 Data Flow Diagrams** | **22** |
|  | **5.2 Solution & Technical Architecture** | **23** |
|  | **5.3 User Stories** | **25** |
| **6.** | **PROJECT PLANNING & SCHEDULING** | **27** |
|  | **6.1 Sprint Planning and Estimation** | **28** |
|  | **6.2 Sprint Delivery Schedule** | **30** |
| **7.** | **CODING & SOLUTIONING** | **31** |
|  | **7.1 Feature 1** | **31** |
|  | **7.2 Feature 2** | **51** |
| **8.** | **TESTING** | **62** |
|  | **8.1 Test Cases** | **62** |
|  | **8.2 User Acceptance Testing** | **64** |
| **9.** | **RESULTS** | **65** |
|  | **9.1 Performance Metrics** | **66** |
| **10.** | **ADVANTAGES & DISADVANTAGES** | **70** |
| **11.** | **CONCLUSION** | **72** |
| **12.** | **FUTURE SCOPE** | **74** |
| **13.** | **APPENDIX** | **76** |
|  | **Source Code** | **76** |
|  | **GitHub Link** | **79** |

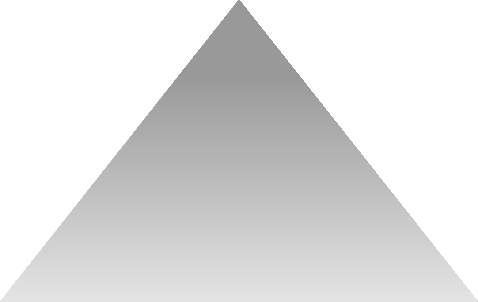
1. **INTRODUCTION**

### PROJECT OVERVIEW

Agriculture is the most important sector in today’s life. Most plants are affected by

a wide variety of bacterial and fungal diseases . Plant diseases cause yield reductions that have a direct influence on the domestic and international food production systems and lead to financial losses. The Food and Agriculture Organization (FAO) of the United Nations International Plant Protection Convention (2017) estimates that plant diseases and pests cause a 20% to 40% loss in worldwide food production. An estimated 13% of the global crop yield loss is attributable to plant diseases. These figures demonstrate how crucial it is to recognise plant diseases in order to reduce production losses. But first, it's essential to comprehend the causes of plant illnesses. Three factors aid disease formation in plants: the host, a favorable environment, and the pathogen. In most cases, diseases begin to show symptoms and affect the plant from the bottom up. The triangle of plant disease is formed by these elements. Plant pathologists can most easily detect foliar diseases, or plant diseases that manifest symptoms on leaves, by looking at these diseases' distinctive characteristics. Up to 50% of yield losses are specifically attributable to fungal infections.

HOST



DISEASE

PATHOGEN ENVIRONMENT

Therefore, the majority of contemporary studies employ computer vision, machine learning and deep learning methods to recognise illnesses in photos of plant leaves. The severity of corn streak disease was also quantified using image processing techniques, and it was shown that computer based approaches were more precise than conventional visual examination. Developed Technologies have provided the ability to produce sufficient food to meet the demand of society. The food's and the crops' safety and security, however, remained unachieved. Farmers face difficulties due to factors such as climate change, a decrease in pollinators, plant diseases, and other issues. These qualities require a strong foundation, which needs to be accomplished as soon as possible. The utilisation of analysis and detection techniques employing current technology aids farmers in solving such issues. The country depends on modern technologies in pandemic scenarios like COVID 19 to handle problems and stop the spread of the diseases. Because they can cause famines and droughts, plant diseases pose a serious threat to human survival.

## PURPOSE

Agriculture provides food to all the human beings even in case of rapid increase in the

population. It is recommended to predict the plant diseases at their early stage in the field of agriculture is essential to cater the food to the overall population. But it is unfortunate to predict the diseases at the early stage of the crops. The idea behind the project is to bring awareness amongst the farmers about the cutting-edge technologies to reduce diseases in plant leaf. Several factors associated with disease diagnosis in plants using deep learning techniques must be considered to develop a robust system for accurate disease management. However, despite the range of applications, several gaps within plant disease research are yet to be addressed to support disease management on farms. Thus, there is a need to establish a knowledge base of existing applications and identify the challenges and opportunities to help advance the development of tools that address farmers’ needs.

The aim of our project is to identify

* what disease does our crop have,
* what is the cause of disease,
* how to prevent the disease,
* how to cure the disease,
* fertilizer recommendation to cure the disease

## LITERATURE SURVEY

### EXISTING PROBLEM

1. This method used datasets to find diseased and healthy plant leaves. we introduced a deep convolutional neural network to identify crop series and diseases that may not be present in the plant tissue. The model trained on the test set has an accuracy of 99.35%. This process is enabled by deep learning, machine learning and digital epidemiologyA neural network associates images of diseased plants and crops as a pair. A neural network node is a mathematical function that receives numerical inputs from input edges and provides numerical outputs as output edges. We analyze 54,306 images of plant leaves that have been assigned a variance of 38 class labels. We resize the images to 256x256 pixels and perform both model optimization and prediction on these reduced images.

**Adavantages:** this system identifies the diseases that may not be present in the plant issue.

**Disadvantages:** analysing all the images of plant might be difficult and time consuming.

**Algorithm used:** deep convolutional neural network

**Technologies Used:** deep learning, machine learning and digital epidemiology

1. This system explains about Plant identification system developed by computer vision researchers to know plant diseases. In this article, A network (CNN) can be used to gain an intuition of selected features based on a deconvolution network (DN) approach. Different order of veins is the best representative feature. We observe a multi-level representation of leaf data compared to that of contour shapes, showing hierarchical transformation of features. From a lower abstraction to a higher abstraction corresponding to the seed class. These insights gave us insight into the design of new hybrid feature extraction models that can continue to improve.

The uniqueness of the plant classification system.

**Advantages:** Features learned using deep learning can improve plant recognition performance

**Disadvantages:** defining featuresparts or patterns of an object in an image that help to identify.

**Keywords:** Plant recognition, deep learning, feature visualisation.

1. This explains about the several ways to recognize plant medical condition. Some diseases have no visible symptoms, or takes effect too late to act, and Advanced analytics require Changes in symptoms exhibited by diseased plants. Evaluate the performance of the detection algorithm. To distinguish between diseased and healthy leaves, another class was added to the dataset.The source was removed using a developed Python script comparison procedure. Script will remove duplicates and Compare image metadata (name, size, date).

**Advantages:** datasets were introduced to detect each area of the leaf (size,veins,thickness).

**Diadavantages:** resolving image size less than 500px is not considered.

1. This proposed system explains about the water needs of plants vary from place to place due to changes in soil content, texture, climatic factors, and more. In addition to water requirements, plant diseases can also cause plants not to grow properly. In this article, we proposed a new intelligent irrigation system that can automatically control irrigation using an Android mobile application. In addition, photos of plant leaves are captured and sent to the cloud server. This is further processed and compared with images of diseased plant leaves in the cloud database. Based on the comparison, a list of suspected plant diseases is displayed to the user via an Android mobile application.
2. The proposed method makes use of soil and PH samples as input and helps predict plants that can be recommended for soil and fertilizer that can be suitable. Information on the ground is collected by sensors and the data is transmitted from the Arduino via Zigbee and WSN (Wireless Sensor Network) to MATLAB. Analysis and processing of soil data are performed using ANN (Artificial Neural Neural Networks) and crop recommendations are carried out using SVMs (Support Vector Machines).

**Advantage:** It helps to improve production at field and income rates, improved crop prediction.

**Disadvantages:** Crop sicknesses cannot be detected and prevented at earlier stage. **Algorithms used:** ANN (Artificial Neural Network), SVM ( Support Vector Machine ) . **Hardware and Software:** Arduino, Zigbee, MATLAB, WSN.

[6]This paper presents a methodology for classifying three major leaf diseases of banana using local textural characteristics. Disease-affected regions are identified using image enhancement and color segmentation. The segmented image is transformed into one transform domain using three Image transforms (DWT, DTCWT, and ranklet transforms). Feature vectors are extracted from transform-domain images using LBP and its variants (ELBP, MeanELBP, and MedianELBP). Experimental results showed the best classification performance of ELBP features extracted from the DTCWT domain (accuracy 95.4%, accuracy 93.2%, sensitivity 93.0%, Fscore 93.0%, and specificity 96.4%).

**Advantage:** Compared with conventional methods of trait extraction, this new method of merging DTCWT with ELBP traits achieved a high level of accuracy in accurately detecting and classifying banana fungal diseases at an early stage.

**Disadvantage:** The plants which are at specific Euclidean distance can only be classified.

**Algorithms used:** DWT (Discrete wavelet transform), LBP (Linear Binary Pattern)

### REFERENCES

#### Using Deep Learning for Image-Based Plant Disease Detection

S. Sankaran, A. Mishra, R. Ehsani, and C. Davis, “A review of advanced techniques for detecting plant diseases,” Computers and Electronics in Agriculture.

#### How Deep Learning Extracts and Learns Leaf Features for Plant Classification

Sue Han Leea, Chee Seng Chan, corresponding authora, Simon Joseph Mayob, Paolo Remagninoc.

#### Deep Neural Networks Based Recognition of Plant Diseases by Leaf ImageClassification

Srdjan Sladojevic , 1 Marko Arsenovic, Andras Anderla, Dubravko Culibrk , and Darko Stefanovic. Department of Industrial Engineering and Management , Faculty of Technical Sciences University of Novi Sad , Trg Dositeja Obradovica 6 , 21000 Novi Sad, Serbia .

#### Using Deep Learning for Image-Based Plant Disease Detection

S. Sankaran, A. Mishra, R. Ehsani, and C. Davis, “A review of advanced techniques for detecting plant diseases,” Computers and Electronics in Agriculture.

#### How Deep Learning Extracts and Learns Leaf Features for Plant Classification

Sue Han Leea, Chee Seng Chan, corresponding authora, Simon Joseph Mayob, Paolo Remagninoc.

#### Deep Neural Networks Based Recognition of Plant Diseases by Leaf ImageClassification

Srdjan Sladojevic , 1 Marko Arsenovic, Andras Anderla, Dubravko Culibrk , and Darko Stefanovic. Department of Industrial Engineering and Management , Faculty of Technical Sciences University of Novi Sad , Trg Dositeja Obradovica 6 , 21000 Novi Sad, Serbia .

#### Cloud based automated irrigation and plant leaf disease detection system using an android application

O. T. Zaheema ,Department of Computer Science Engineering, Eranad Knowlededge City Technical Campus, Manjeri, Kerala, India

#### Agro based crop and fertilizer recommendation system using machine learning

Preethi G , Rathi Priya V , Sanjula S M ,Lalitha S D , Vijaya Bindhu B. Final Year CSE,

R.M.K Engineering College , [rath16309.cs@rmkec.ac.in](mailto:rath16309.cs@rmkec.ac.in) , Assistant Professor , R.M.K. Engineering College , [sdi.cse@rmkec.ac.in](mailto:sdi.cse@rmkec.ac.in) , Cognizant Technology Solutions - 2020.

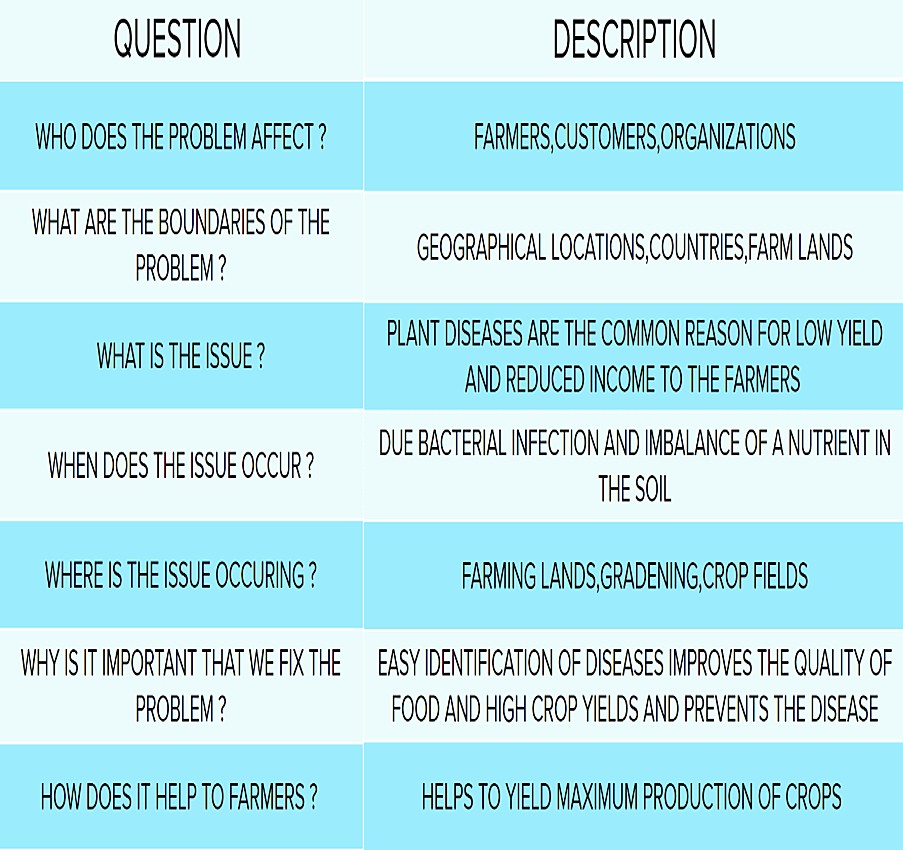
#### Foliar fungal disease classification in banana plants using elliptical local binary pattern on multiresolution dual tree complex wavelet transform domain

Deepthy Mathew, C. Sathish Kumar, K. Anita Cherian. Department of Electronics and Communication Engineering , Rajiv Gandhi Institute of Technology , APJ Abdul Kalam Technological University, Kottayam 686501, India . Department of Electronics and Communication Engineering , Government Engineering College, Idukki 685603, India . Department of Plant Pathology , College of Horticulture , Kerala Agricultural University, Thrissur 680656, India-2020.

#### An automated segmentation and classification model for banana leaf disease detection

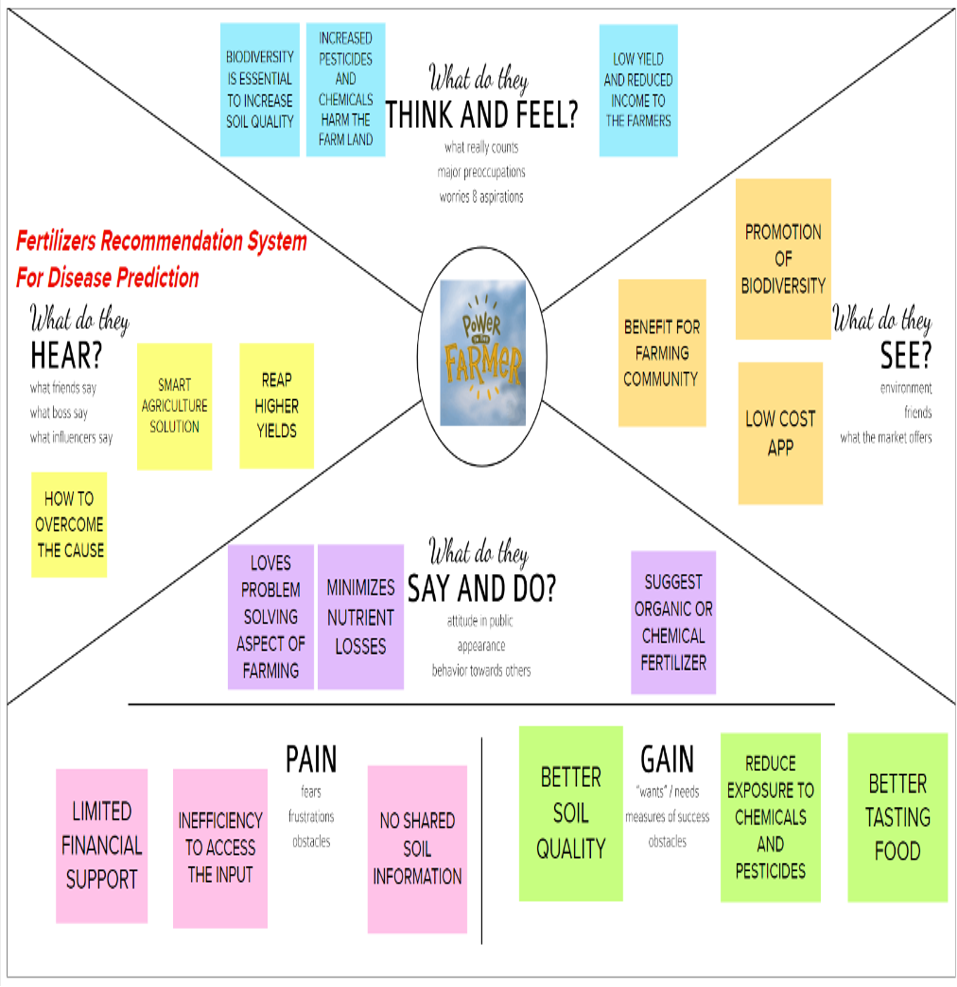
V. Gokula Krishnan1 , J. Deepa , Pinagadi Venkateswara Rao , V. Divya , S. Kaviarasan.Associate Professor , CSIT Department , CVR College of Engineering , Hyderabad , India.Assistant Professor , CSE Department , EaswariEngineering College , Chennai ,India- 2022.

### PROBLEM STATEMENT DEFINITION

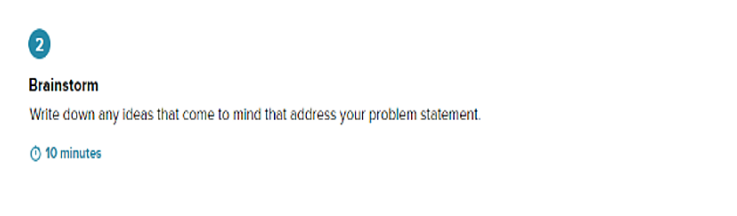
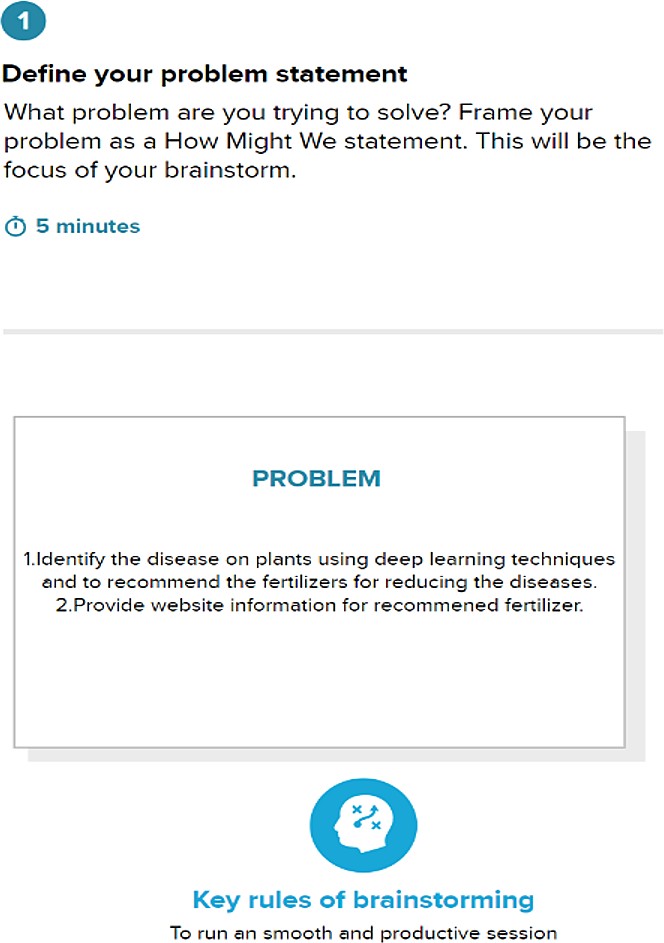


1. **IDEATION & PROPOSED SOLUTION**

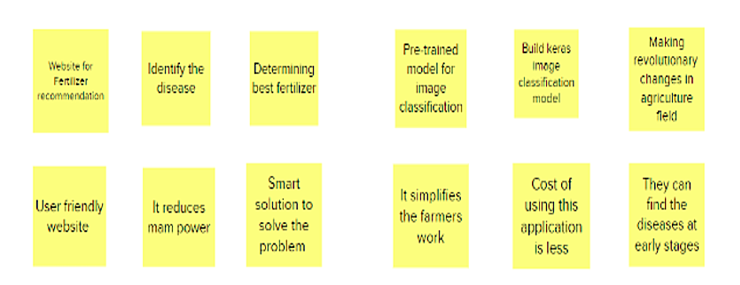
### EMPATHY MAP



* 1. **IDEATION AND BRAINSTROMING**

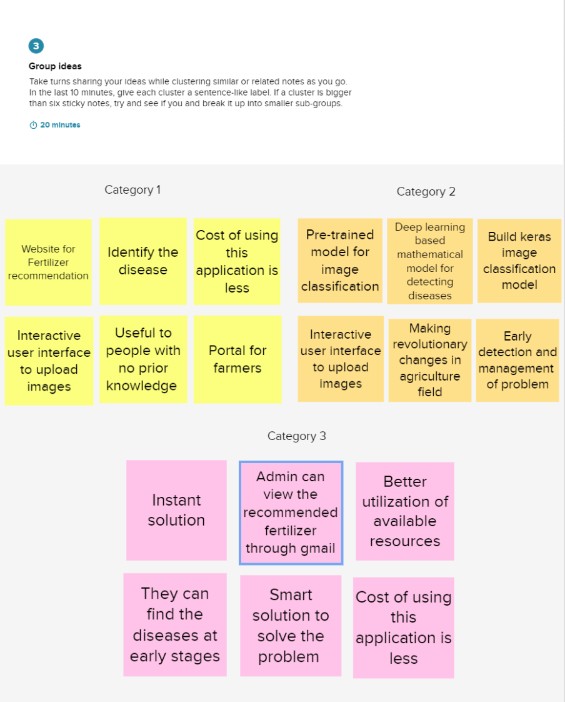
****

HIMANSHU ROY MOTCHA ENOK C



LESSLIE JOHN D VIKRAM M





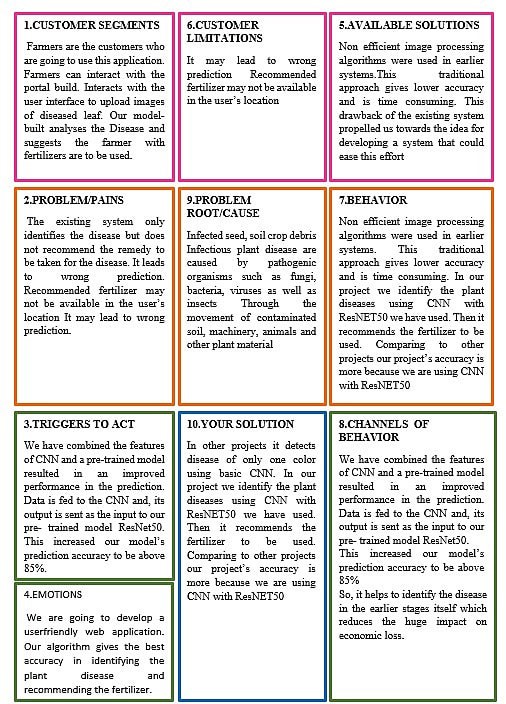


### PROPOSED SOLUTION

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| 1. | Problem Statement (Problem to be solved) | In agricultural aspects, if the plant is affected by leaf disease, then it reduces the growth and productivity. Generally, the plant diseases are caused by the abnormal physiological functionalities of plants. The issue occurs in agriculture practicing areas, particularly in rural regions. |
| 2. | Novelty / Uniqueness | 1. We have combined the features of CNN and a pre-trained model resulted in an improved performance in the prediction. 2. Data is fed to the CNN. And, its output is sent as the input to our pre- trained model ResNet50. This increased our model’s prediction accuracy to be above   85% |
| 3. | Social Impact / Customer Satisfaction | * By recommending the appropriate   fertilizers for the diseases predicted, the quality of food products improves. It also controls the disease in plants. |

|  |  |  |
| --- | --- | --- |
|  |  | * This also maximizes the crop yield by using the land efficiently. |
| 4. | Business Model (Revenue Model) | * Helps farmers to grow more food using fewer resources by reducing the damage caused by irrelevant fertilizers and diseases attacked. * With the proposed model crop yield, farm efficiency, agricultural product output will be increased * A high gain can be seen in the agricultural output and profit will be increased. |
| 5. | Scalability of the Solution | * Deep learning technique is used to identify the diseases and better fertilizer suggestions that can be recommended for those diseases. * Using Deep Learning techniques for recommendation reduces the time taken to detect diseases than other traditional methods. |

* 1. **PROBLEM SOLUTION FIT**



## REQUIREMENT ANALYSIS

### FUNCTIONAL REQUIREMENT

Following are the functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement** | **Sub Requirement** |
| FR-1 | User Registration | Registration through form Registration through Gmail Registration through LinkedIn |
| FR-2 | Image Capture | Take image of a leaf  Check the leaf is captured under given parameters |
| FR-3 | Image Processing | Upload the leaf image Click the predict button |
| FR-4 | Updated Native Language | Languages can be changed according to the user, which he is more understandable with. (Ex: English,  Hindi, Tamil) |
| FR-5 | Leaf Prediction | Add the pesticides and fertilizers to be used for an unhealthy leaf |
| FR-6 | Image Description | Show the prescribed fertilizer and description of the disease for curing a unhealthy leaf |
| FR-7 | Providing Datasets | Training datasets Tes |
| FR-8 | Adding Datasets | Fruit datasets for fruits Vegetable  datasets for vegetables |

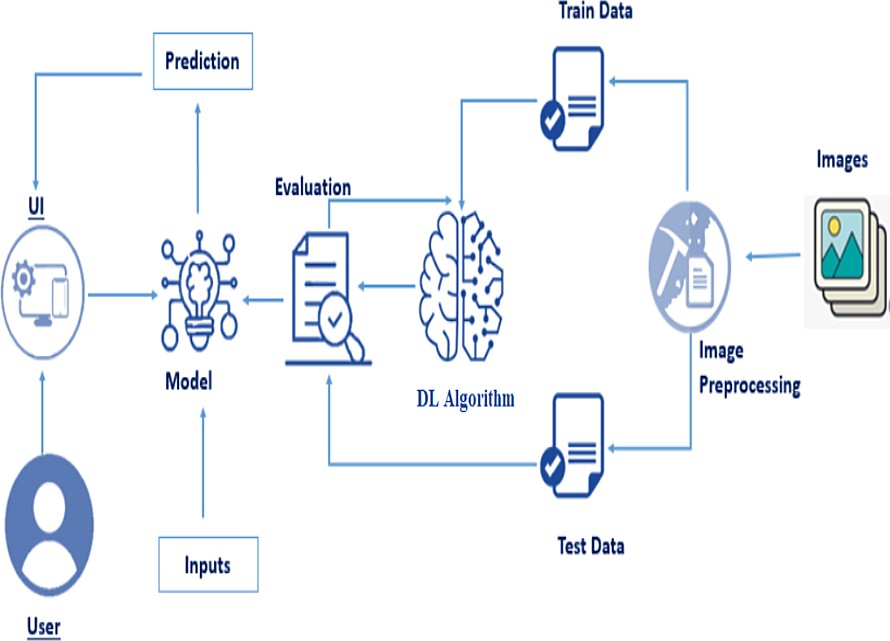
### NON - FUNCTIONAL REQUIREMENTS

Following are the non-functional requirements of the proposed solution.

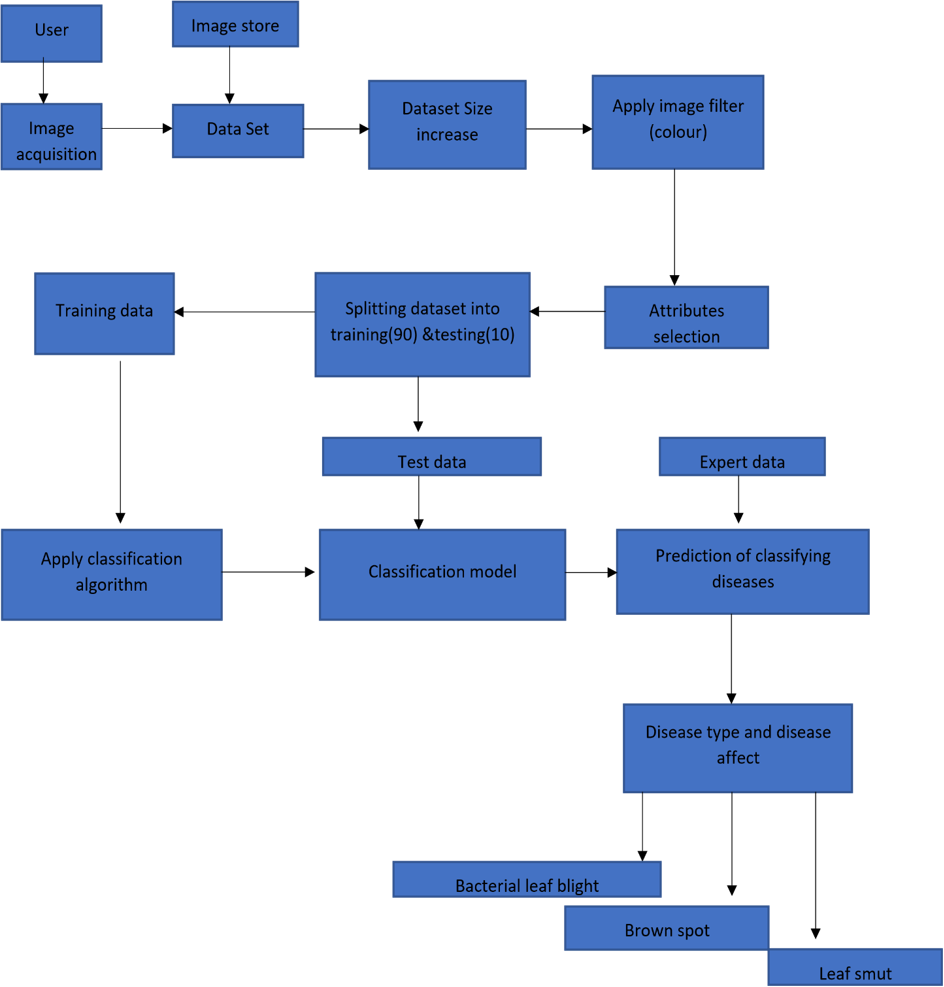
|  |  |  |
| --- | --- | --- |
| **NFR No.** | **Non-Functional**  **Requirement** | **Description** |
| NFR-1 | Usability | Leaf datasets can be used for detection of all kind of leafs Datasets can be reusable Data sets can be prepared according to the  leaf |
| NFR-2 | Security | User information and leaf data are secured  The algorithms used are more  secure |
| NFR-3 | Reliability | The leaf quality is more  The datasets and image capturing performs consistently well |
| NFR-4 | Performance | Leaf problem defines once the leaf is detected  Performs well according to the quality of leaf provides  certain cure to it. |
| NFR-5 | Availability | Quality of leaf will be used again for detection  Available and easy access of  datasets provided |
| NFR-6 | Scalability | Increase in growth of predicting  the results and defining a leaf |

## PROJECT DESIGN

An automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant. Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases.

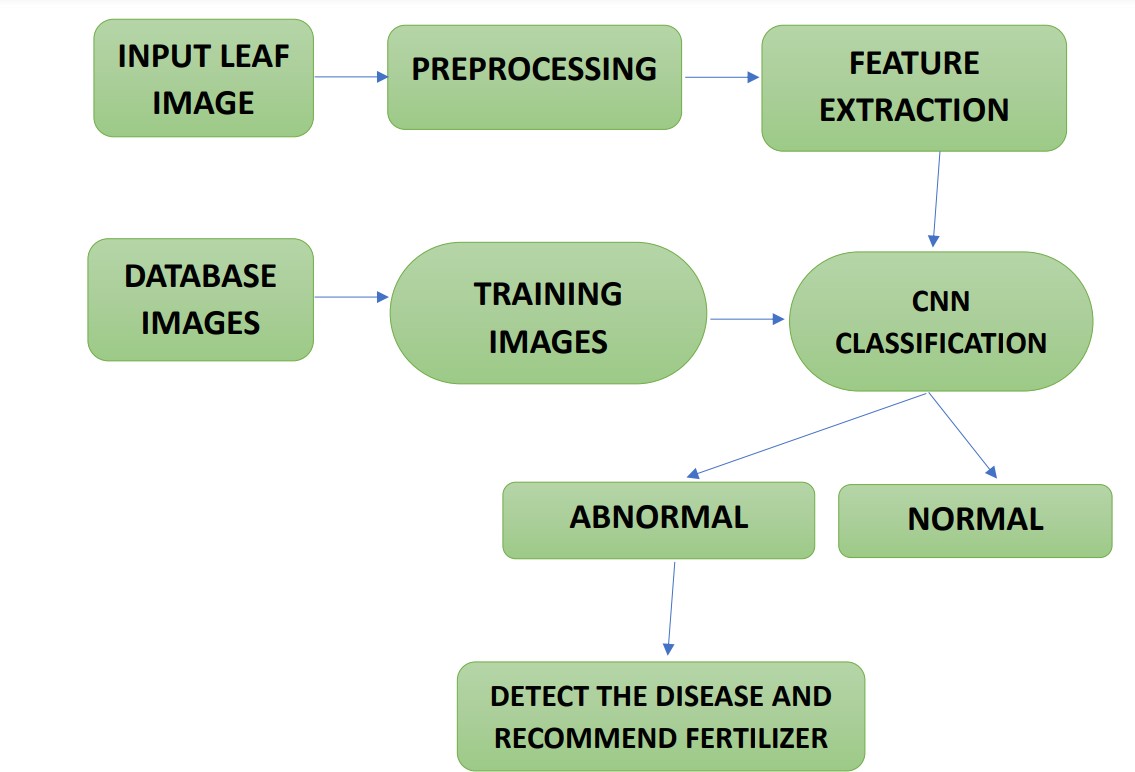


### DATA FLOW DIAGRAMS

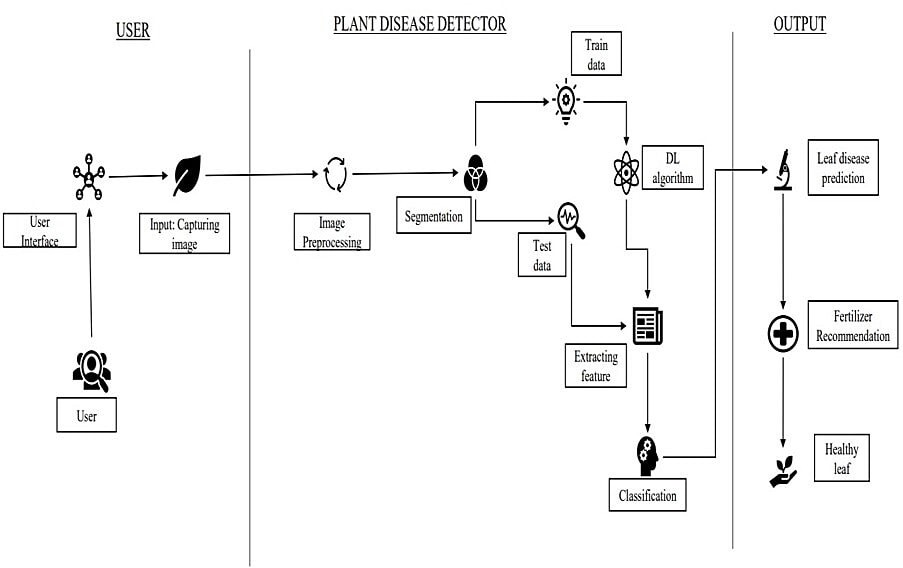


* 1. **SOLUTION & TECHNICAL ARCHITECTURE**

### SOLUTION ARCHITECTURE:



**TECHNICAL ARCHITECTURE:**



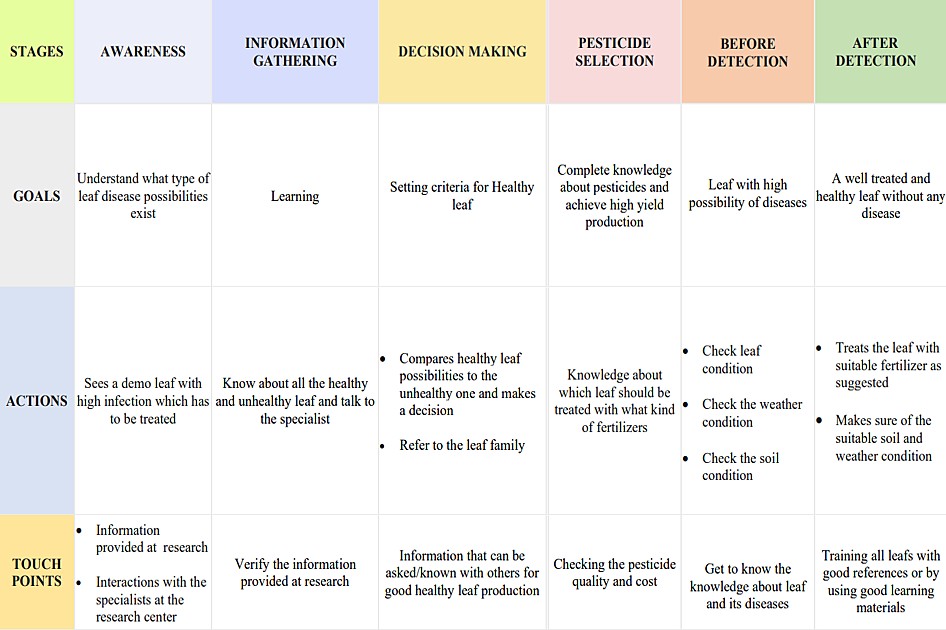
#### TABLE -1: COMPONENTS & TECHNOLOGIES:

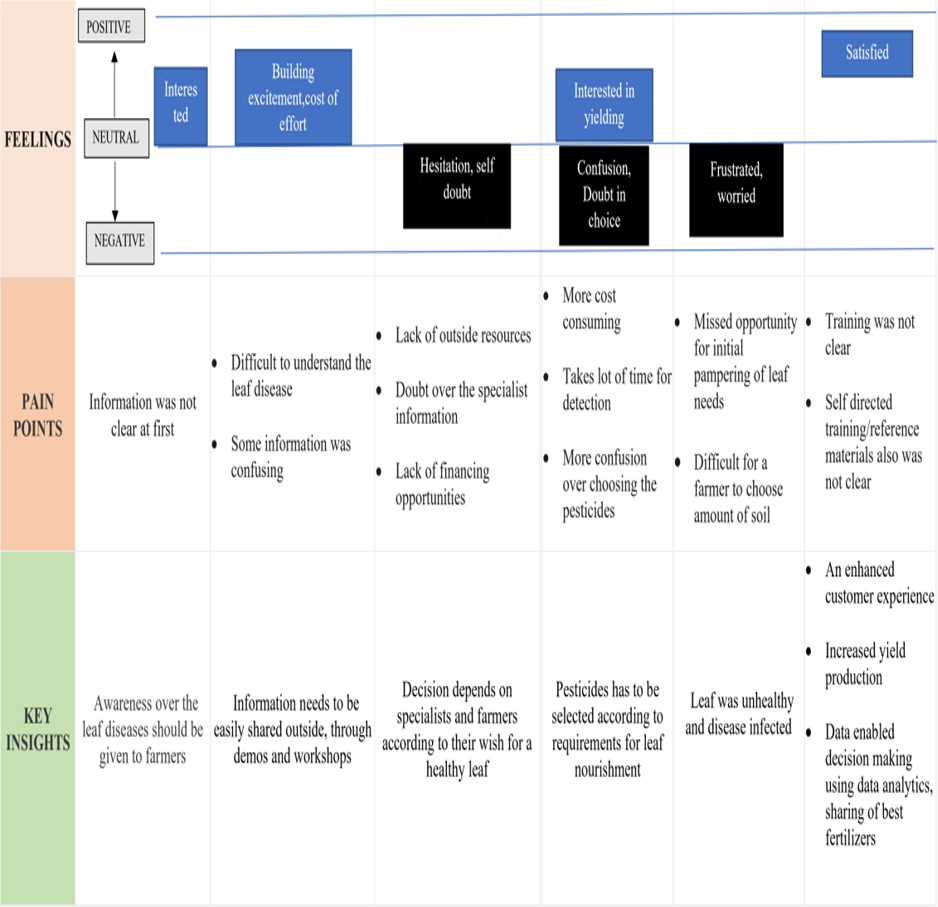
|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | Component | Description | Technology |
| 1, | User Interface | How user interacts with the website. | HTML,CSS, etc,. |
| 2, | Disease Prediction | Here the disease in the leaf is predicted | Keras,CNN. |
| 3. | Fertilizer Recommendation | The fertilizer is recommended for the predicted disease | User interface, HTML, CSS. |
| 4. | Dataset | The training and testing data are collectively stored | Kaggle.com, data.gov, UCI machine learning repository,  etc. |
| 5. | File Storage | File storage requirements | IBM, Local File system. |
| 6, | Modules | Purpose of deep learning modules | Image Recognition Modules,etc. |
| 7. | Infrastructure(Server) | Application development on Local System-local server configuration: | Local File system. |

**TABLE – 2: APPLICATION CHARACTERISTICS:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | Characteristics | Description | Technology |
| 1. | Opensource Framework | List of the opensource framework used | Open source-PyCharm, anaconda navigator, flask framework. |
| 2. | Login | List of the access control implementation | Security - OWASP |
| 3. | Scalable Architecture | Justify the scalable architecture | PyCharm |
| 4. | Availability | Justify the availability of website | Web application access to all. |
| 5. | Performance | Design consideration for the performance of the website | Convolutional Neural Networks. |

* 1. **USER STORIES**





## PROJECT PLANNING & SCHEDULING

### SPRINT PLANNING & ESTIMATION

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement**  **(Epic)** | **User Story**  **Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| Sprint-1 | Data collection and  preprocessing | USN-1 | Collecting plant disease dataset | 2 | Low | HIMANSHU ROY |
| Sprint-1 |  | USN-2 | Labelling the dataset according to  class | 3 | Medium | LESSLIE JOHN D |
| Sprint-1 |  | USN-3 | 38 types of plant  diseases is labeled accordingly | 2 | Medium | MOTCHA ENOK C |
| Sprint-1 |  | USN-4 | Data set Will contain both healthy and  diseased data | 1 | Low | VIKRAM M |
| Sprint-1 | Preprocessing | USN-5 | To prepare raw data in a format that the  network can accept | 2 | High | HIMANSHU ROY |
| Sprint-1 |  | USN-7 | Shear range image will be distorted along an axis, mostly to create or rectify the  perception angle | 3 | High | MOTCHA ENOK C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 |  | USN-8 | Zoom  Augmentation will randomly zoom the image and adds new pixels for the image | 3 | High | LESSLIE JOHN D |
| Sprint-1 |  | USN-9 | Flipping the entire  pixels of an image horizontally | 3 | High | VIKRAM M |
| Sprint-2 | Training ,  Testing and Creating a model | USN-10 | Start initiating the  model | 3 | Medium | LESSLIE JOHN D |
| Sprint-2 |  | USN-11 | Adding different  layers of cnn( convolution, pooling dense , flatten ) | 2 | Medium | MOTCHA ENOK C |
| Sprint-2 |  | USN-12 | Creating/compiling  with adam optimizer | 1 | Medium | HIMANSHU ROY |
| Sprint-2 |  | USN-13 | Keras - Categorical  Cross Entropy Loss Function for multi- class classification | 2 | Medium | VIKRAM M |
| Sprint-2 |  | USN-14 | creating metrics | 2 | Medium | MOTCHA ENOK C |
| Sprint-2 |  | USN-15 | train the data with  20 epoch | 3 | High | LESSLIE JOHN D |
| Sprint-2 |  | USN-16 | testing the model | 5 | High | HIMANSHU ROY |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-2 |  | USN-17 | save the model | 2 | Medium | VIKRAM M |
| Sprint-3 | Flask and  Frame workdesign | USN-18 | Creating backend  framework with flask | 8 | High | MOTCHA ENOK C |
| Sprint-3 |  | USN-19 | importing the  model file | 3 | Medium | LESSLIE JOHN D |
| Sprint-3 |  | USN-20 | Create route to link  htmlRoutes and View Functions in Flask Framework index file | 5 | High | VIKRAM M |
| Sprint-3 |  | USN-21 | Server Startup,  requests and services in a loop | 4 | Medium | HIMANSHU ROY |
| Sprint-4 | Front end web  application development | USN-22 | creating a html  template with css file | 8 | High | LESSLIE JOHN D |
| Sprint-4 |  | USN-23 | user can import  diseased plant leaf in web page | 2 | Medium | MOTCHA ENOK C |
| Sprint-4 |  | USN-24 | predicting what is  the type of disease occurred for the given input | 2 | Medium | VIKRAM M |
| Sprint-4 |  | USN-25 | User can classify as  healthy or diseased | 2 | Medium | HIMANSHU ROY |
| Sprint-4 |  | USN-26 | if plant has disease  then suggest fertilizer and pesticides | 3 | Medium | LESSLIE JOHN D |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-4 |  | USN-27 | alert the admin  about the prediction with the gmail | 3 | Medium | MOTCHA ENOK C |

* 1. **SPRINT DELIVERY SCHEDULE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Total Story Points** | **Duration** | **Sprint Start Date** | **Sprint End Date (Planned)** | **Story Points Completed (as on Planned End**  **Date)** | **Sprint Release Date (Actual)** |
| Sprint-1 | 20 | 6 Days | 24 Oct 2022 | 29 Oct 2022 | 20 | 27 Oct 2022 |
| Sprint-2 | 20 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | 20 | 3 Nov 2022 |
| Sprint-3 | 20 | 6 Days | 07Nov  2022 | 12 Nov 2022 | 20 | 10 Nov 2022 |
| Sprint-4 | 20 | 6 Days | 14Nov 2022 | 19 Nov 2022 | 20 | 17 nov 2022 |

## CODING & SOLUTIONING

### FEATURE 1 DATASET

Two datasets will be used, we will be creating two models one to detect vegetable leaf diseases like tomato, potato, and pepper plants and the second model would be for fruits diseases like corn, peach, and apple.

### Downloading the Plant Disease dataset from the below link

<https://drive.google.com/file/d/1Jq2f7SNgBO6J13YQU3BJFg69xNKN7M-o/view?usp=drivesdk>

### IMAGE PREPROCESSING

Before training the model, you have to pre-process the images and then feed them on to the model for training. We make use of Keras ImageDataGenerator class for image pre-processing.

Image Pre-processing includes the following main tasks

* Import ImageDataGenerator Library.
* Configure ImageDataGenerator Class.
* Applying ImageDataGenerator functionality to the trainset and test set.

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.

There are five main types of data augmentation techniques for image data; specifically:

* Image shifts via the width\_shift\_range and height\_shift\_range arguments.
* The image flips via the horizontal\_flip and vertical\_flip arguments.
* The image rotates via the rotation\_range argument
* Image brightness via the brightness\_range argument.
* The image zooms via the zoom\_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.

### Image agumentation

**from** keras.preprocessing.image **import** ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale **=** 1.**/**255, shear\_range **=** 0.2,zoom\_range **=**

0.2,horizontal\_flip **=True**)

test\_datagen **=** ImageDataGenerator(rescale **=** 1)

x\_train=train\_datagen.flow\_from\_directory('/content/drive/MyDrive/fruit-dataset/fruit- dataset/train',batch\_size**=**32,target\_size**=**(128,128),color\_mode**=**'rgb',class\_mode**=**'categorical')

x\_test=test\_datagen.flow\_from\_directory('/content/drive/MyDrive/fruit-dataset/fruit- dataset/test',batch\_size**=**32,target\_size**=**(128,128),color\_mode='rgb',class\_mode='categorical ')

### OUTPUT:

Found 5384 images belonging to 6 classes. Found 1686 images belonging to 6 classes.

## MODEL BUILDING FOR FRUIT DISEASE PREDICTION

For model building we are following the below steps

* + Import the libraries
  + Initializing the model
  + Add CNN layers
  + Add dense layer
  + Train and Save the model

**IMPORT THE LIBRARIES**

Here we have Imported the libraries that are required to initialize the neural network layer, and create and add different layers to the neural network model.

**from** keras.models **import** Sequential

**from** keras.layers **import** Dense

**from** keras.layers **import** Convolution2D **from** keras.layers **import** MaxPooling2D **from** keras.layers **import** Flatten

**INITIALIZING THE MODEL**

Keras has 2 ways to define a neural network:

* Sequential
* Function API

Here we are using Sequential class . The Sequential class is used to define linear initializations onetwork layers which then, collectively, constitute a model.

We will use the Sequential constructor to create a model, which will then have layers added to it using the add () method.Now, will initialize our model.

Initialize the neural network layer by creating a reference/object to the Sequential class.

### model=Sequential()

#### ADD CNN LAYERS

We will be adding three layers for CNN

* Convolution layer
* Pooling layer
* Flattening layer

#### ADD CONVOLUTION LAYER

The first layer of the neural network model, the convolution layer will be added. To create a convolution layer, Convolution2D class is used. It takes a number of feature detectors, feature detector size, expected input shape of the image, and activation function as arguments. This layer applies feature detectors on the input image and returns a feature map (features from the image).

Activation Function: These are the functions that help us to decide if we need to activate the node or not. These functions introduce non-linearity in the networks.

### model.add(Convolution2D(32,(3,3),input\_shape=(128,128,3),activation='relu'

))

## Add the pooling layer

Max Pooling selects the maximum element from the region of the feature map covered by the filter. Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

After the convolution layer, a pooling layer is added. Max pooling layer can be added using MaxPooling2D class. It takes the pool size as a parameter. Efficient size of the pooling matrix is (2,2). It returns the pooled feature maps. (Note: Any number of convolution layers, pooling and dropout layers can be added

**model.add(MaxPooling2D(pool\_size = (2,2)))**

## Add the flatten layer

The flatten layer is used to convert n-dimensional arrays to 1-dimensional arrays. This 1D array will be given as input to ANN layers.

**model.add(Flatten()**

## ADD DENSE LAYERS

This step is to add a dense layer (output layer) where you will be specifying the number of classes your dependent variable has, activation function, and weight initializer as the arguments. We use the add () method to add dense layers. the output dimensions here is 6

**model.add(Dense(40, 'relu')) model.add(Dense(20, 'relu')) model.add(Dense(6, 'softmax', ))**

## TRAIN AND SAVE THE MODEL

#### COMPILE THE MODEL

After adding all the required layers, the model is compiled, for this step, loss function, optimizer and metrics for evaluation can be passed as arguments

**model.compile(optimizer='adam', loss ="categorical\_crossentropy" , metrics**

**=['accuracy'])**

## FIT AND SAVE THE MODEL

Fit the neural network model with the train and test set

### model.fit(x\_train,epochs=20,steps\_per\_epoch=89,validation\_data = x\_test, validation\_steps = 27)

The weights are to be saved for future use. The weights are saved in as .h5 file using save().

## model.save("fruit.h5")

**model.summary()** can be used to see all parameters and shapes in each layer in our models

### OUTPUT:

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 126, 126, 32) 896

max\_pooling2d (MaxPooling2D (None, 63, 63, 32) 0

)

|  |  |  |
| --- | --- | --- |
| flatten (Flatten) | (None, 127008) | 0 |
| dense (Dense) | (None, 40) | 5080360 |
| dense\_1 (Dense) | (None, 20) | 820 |
| dense\_2 (Dense) | (None, 6) | 126 |

=================================================================

Total params: 5,082,202

Trainable params: 5,082,202

Non-trainable params: 0

Epoch 1/20

89/89 [==============================] - 717s 8s/step - loss: 1.3023 - accuracy:

0.5609 - val\_loss: 59.3136 - val\_accuracy: 0.7199

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Epoch 2/20 |  | | | | | | | |
| 89/89 [==============================] | - 354s | | 4s/step | - loss: | | 0.6571 | - accuracy: | |
| 0.7882 - val\_loss: 60.1567 - val\_accuracy: 0.7824 |  | |  |  | |  |  | |
| Epoch 3/20 |  | |  |  | |  |  | |
| 89/89 [==============================] | - 183s | | 2s/step | - loss: | | 0.4134 | - accuracy: | |
| 0.8615 - val\_loss: 124.2583 - val\_accuracy: 0.6863 |  | |  |  | |  |  | |
| Epoch 4/20 |  | |  |  | |  |  | |
| 89/89 [==============================] | - 108s | | 1s/step | - loss: | | 0.3113 | - accuracy: | |
| 0.8982 - val\_loss: 615.5879 - val\_accuracy: 0.4329 |  | |  |  | |  |  | |
| Epoch 5/20 |  | |  |  | |  |  | |
| 89/89 [==============================] - | 75s | 836ms/step | | - | loss: | 0.2583 - | | accuracy: |
| 0.9129 - val\_loss: 541.0003 - val\_accuracy: 0.4641 |  |  | |  |  |  | |  |
| Epoch 6/20 |  |  | |  |  |  | |  |
| 89/89 [==============================] - | 60s | 673ms/step | | - | loss: | 0.2481 - | | accuracy: |
| 0.9112 - val\_loss: 663.6074 - val\_accuracy: 0.4630 |  |  | |  |  |  | |  |
| Epoch 7/20 |  |  | |  |  |  | |  |
| 89/89 [==============================] - | 54s | 599ms/step | | - | loss: | 0.2167 - | | accuracy: |
| 0.9252 - val\_loss: 504.1471 - val\_accuracy: 0.4850 |  |  | |  |  |  | |  |
| Epoch 8/20 |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |

**FEATURE 2**

**MODEL BUILDING FOR VEGETABLE DISEASE PREDICTION**

For model building we are following the below steps

* + Import the libraries
  + Initializing the model
  + Add CNN layers
  + Add dense layer
  + Train and Save the model

#### IMPORT THE LIBRARIES

Here we have Imported the libraries that are required to initialize the neural network layer, and create and add different layers to the neural network model.

**from** keras.models **import** Sequential

**from** keras.layers **import** Dense

**from** keras.layers **import** Convolution2D **from** keras.layers **import** MaxPooling2D **from** keras.layers **import** Flatten

#### INITIALIZING THE MODEL

Keras has 2 ways to define a neural network:

* Sequential
* Function API

Here we are using Sequential class . The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model.

We will use the Sequential constructor to create a model, which will then have layers added to it using the add () method.Now, will initialize our model.

Initialize the neural network layer by creating a reference/object to the Sequential class.

### model=Sequential()

#### ADD CNN LAYERS

We will be adding three layers for CNN

* Convolution layer
* Pooling layer
* Flattening layer

### model.add(Convolution2D(32,(3,3),input\_shape = (128,128,3),activation = 'relu'))

**model.add(MaxPooling2D(pool\_size = (2,2))) model.add(Flatten())**

#### ADD DENSE LAYERS

This step is to add a dense layer (output layer) where you will be specifying the number of classes your dependent variable has, activation function, and weight initializer as the arguments. We use the add () method to add dense layers. the output dimensions here is 6

### model.add(Dense(300, 'relu')) model.add(Dense(150, 'relu')) model.add(Dense(75, 'relu')) model.add(Dense(9, 'softmax', ))

#### TRAIN AND SAVE THE MODEL

**COMPILE THE MODEL**

After adding all the required layers, the model is compiled, for this step, loss function, optimizer and metrics for evaluation can be passed as arguments

### model.compile(optimizer='adam', loss = "categorical\_crossentropy" , metrics

**=['accuracy'])**

#### FIT AND SAVE THE MODEL

Fit the neural network model with the train and test set

### model.fit(x\_train,epochs=20,steps\_per\_epoch=89,validation\_data = x\_test, validation\_steps = 27)

The weights are to be saved for future use. The weights are saved in as .h5 file using save().

### model.save("veg.h5")

**TEST BOTH THE MODEL**

Now that we have trained both the models, testing both the models by loading the saved models.

#### TEST THE MODEL

The model is tested with different images to know if it is working correctly

**Import the packages and load the saved model**

#### Import the libraries

**from** tensorflow.keras.preprocessing **import** image

**from** tensorflow.keras.preprocessing.image **import** img\_to\_array

**from** tensorflow.keras.models **import** load\_model

**import** numpy **as** np

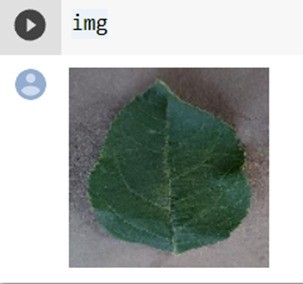
Initially, we will be loading the fruit model and test it with the vegetable model in a similar way.

model = load\_model(**'fruit.h5'**)

Pre-processing the image includes converting the image to array and resizing according to the model.The pre-processed image is given to the model to know to which class your model belongs to.

**img = image.load\_img**(r"/home/wsuser/work/Dataset Plant Disease/fruit-dataset/fruit- dataset/test/Apple healthy/0a553fc0-fc2c-4598-baba-3bc10191447c RS\_HL 5969.JPG", **target\_size = (128,128))**

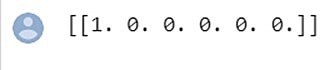
**OUTPUT:**



x=image.img\_to\_array(img) x=np.expand\_dims(x,axis=0) result=model.predict(img) **print**(result)

### OUTPUT

The predicted class is 1



## FEATURE 2

**Application Building**

After the model is built, we will be integrating it into a web application so that normal users can also use it. The new users need to initially register in the portal. After registration users can log in to browse the images to detect the disease.

In this section, you have to build

* + - HTML pages - front end
    - Python script - Server-side script

## Build Python Code

After the model is built, we will be integrating it into a web application so that normal users can also use it. The user needs to browse the images to detect the disease.

**Activity 1:** Build a flask application

**Step 1**: Load the required packages

### from future import division, print\_function import os

**import numpy as np**

### import cv2

# Keras

### from tensorflow.keras.models import load\_model

**from tensorflow.keras.preprocessing.image import img\_to\_array**

# Flask utils

### from flask import Flask, request, render\_template from werkzeug.utils import secure\_filename

**Step 2**: Initializing the flask app and loading the model

flask applications must create an application instance. The web server passes all the requests it receives from clients to objects for handling using a protocol for WSG from flask import Flask app = Flask ( name ) (An application instance is an object of class Flask.)

### app = Flask( name ) MODEL\_PATH = 'fruit.h5'

**MODEL LOADING**

### model = load\_model(MODEL\_PATH) model.make\_predict\_function() default\_image\_size = (128, 128)

**abels=["Apple Black\_rot","Apple healthy","Corn\_(maize) healthy", "Corn\_(maize) Northern\_Leaf\_Blight","Peach Bacterial\_spot","Peach**

### healthy"]

**def convert\_image\_to\_array(image\_dir): try:**

### image = cv2.imread(image\_dir)

**if image is not None:**

### image = cv2.resize(image, default\_image\_size) return img\_to\_array(image)

**else:**

### return np.array([]) except Exception as e: print(f"Error : {e}") return None

**def model\_predict(file\_path, model):**

### x = convert\_image\_to\_array(file\_path) x = np.expand\_dims(x, axis=0)

**preds = model.predict(x)**

### return preds

**Step 3:** Configure the home page

Routes and View Functions in Flask Framework Instance

Clients send requests to the webserver, in turn, sends them to the Flask application instance. The instance needs to know what code needs to run for each URL requested and map URLs to Python functions. The association between a URL and the function that handles it is called a route. The most convenient way to define a route in a Flask application is through the (app.route). Decorator exposed by the application instance, which registers the ‘decorated

function,’ decorators are python feature that modifies the behavior of a function.

### @app.route("/", methods=['GET']) def index():

**return render\_template("index.html", query="")**

**Step 4:** Pre-process the frame and run

Pre-process the captured frame and given it to the model for prediction. Based on the prediction the output text is generated and sent to the HTML to display.

#### Request

To process incoming data in Flask, you need to use the request object, including mime-type, IP address, and data. HEAD: Un-encrypted data sent to server w/o response.

#### GET

Sends data to the server requesting a response body.

#### POST

Read form inputs and register a user, send HTML data to the server are methods handled by the route. Flask attaches methods to each route so that different view functions can handle different request methods to the same URL.

### @app.route("/", methods=['GET', 'POST']) def upload():

**if (request.method == 'POST'): f = request.files['file']**

### basepath = os.path.dirname( file ) file\_path=os.path.join(basepath,'uploads',secure\_filename(f.filename)) f.save(file\_path)

**preds = model\_predict(file\_path, model) preds = np.argmax(preds)**

### result = labels[preds]

**return render\_template('index.html', prediction\_text=result) return None**

**Server Startup -** The application instance has a ‘run’ method that launches flask’s integrated development webserver –

### if name == " main ": app.run(debug=True)

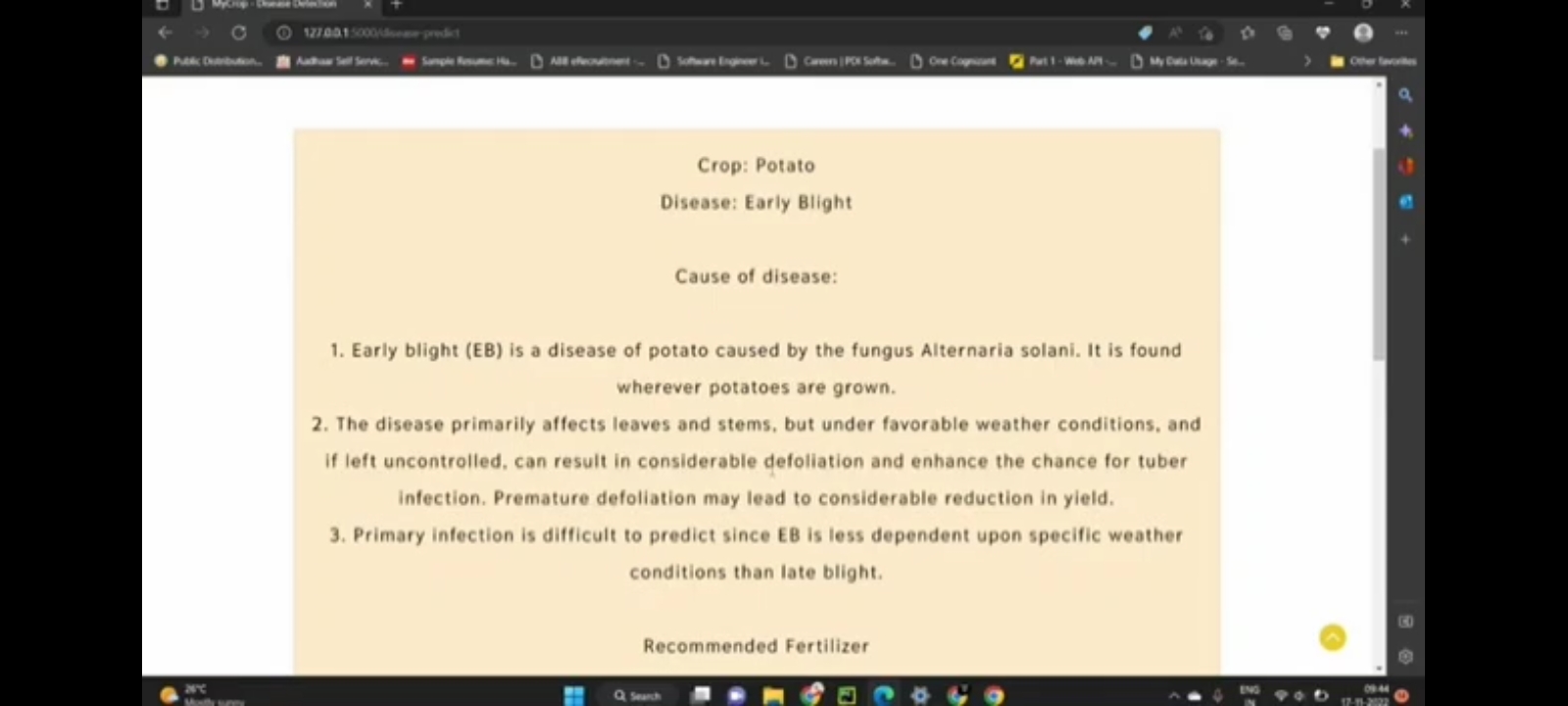
**Output:**

* Serving Flask app 'app'
* Debug mode: on

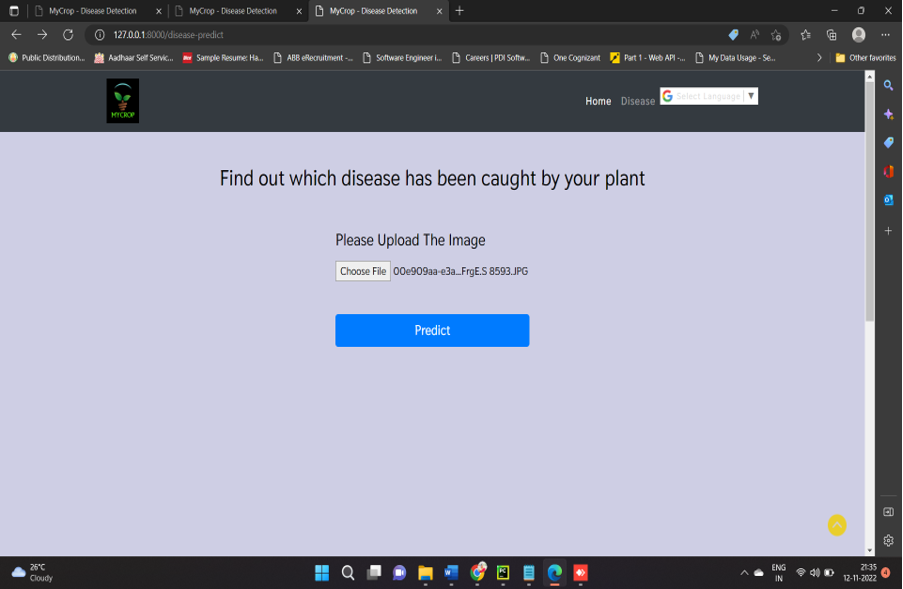
\* Running on http://127.0.0.1:500

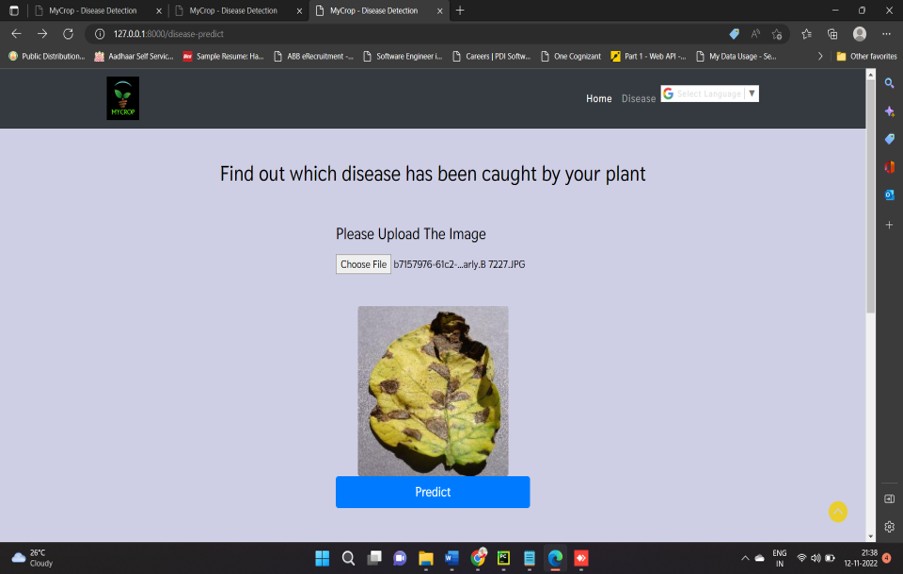
## Build the UI where a home page will have details about the application, a prediction page where a user is allowed to browse an image and get the predictions

**[WEB APPLICATION SCREEN SHOTS]**



**fter clicking on disease button, you will be redirected to the find out which disease has been caught by your plant page where you can browse the images.**





## RUN THE CODE

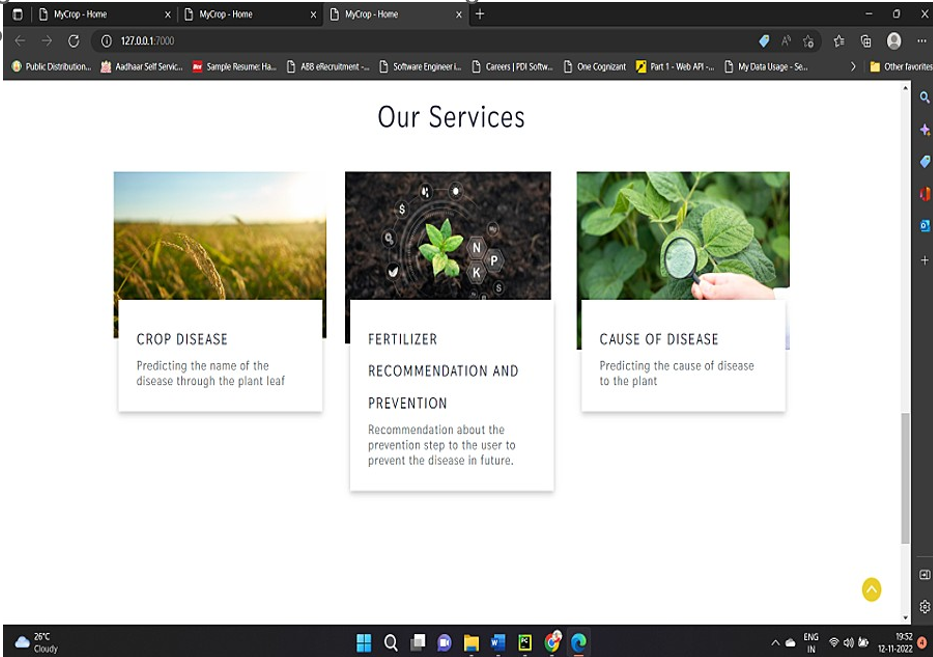
**Step 1**: Run the application.

Open the browser and navigate to localhost to check your web application

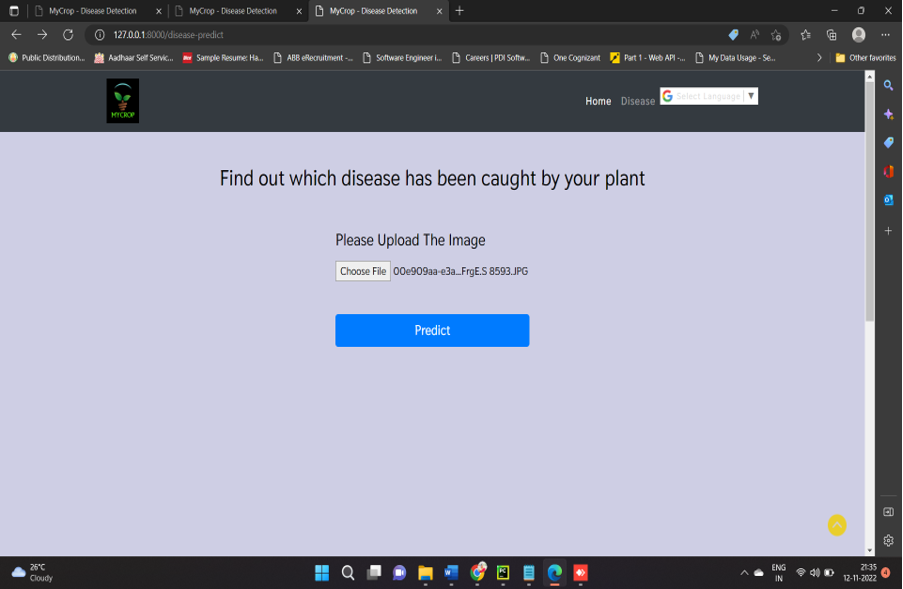
## Running the application on http://127.0.0.1:5000

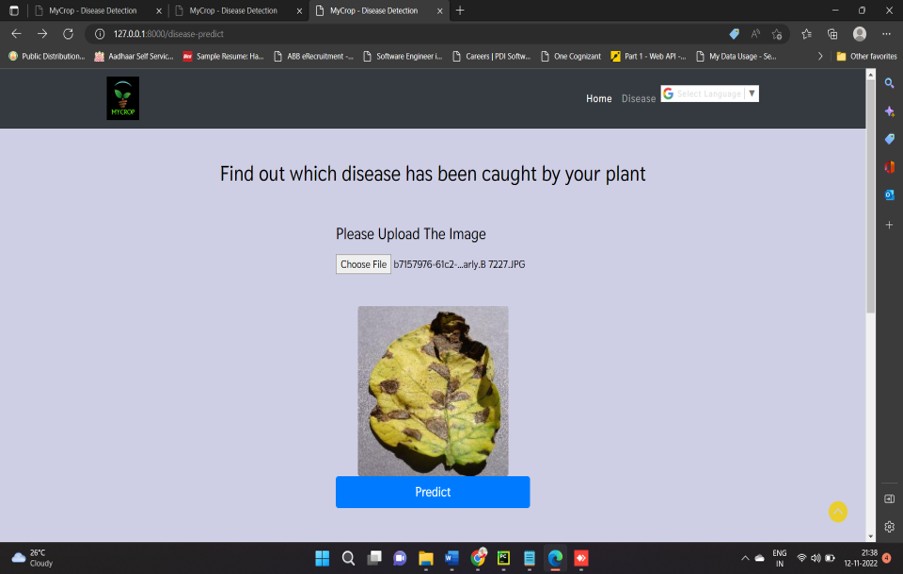
**Step 2:** The home page looks like this

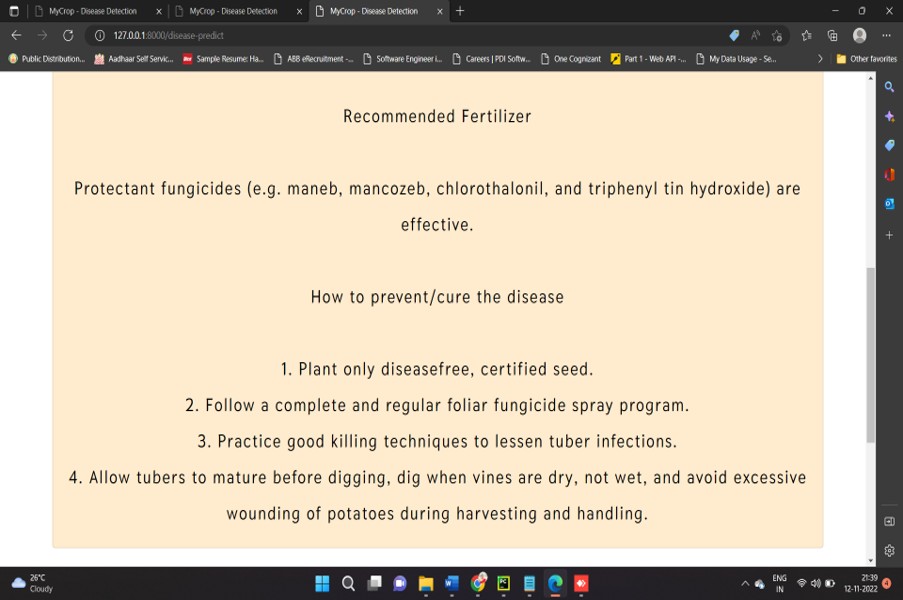




**After clicking on disease button, you will be redirected to the find out which disease has been caught by your plant page where you can browse the images**







## TESTING

* 1. **TEST CASES**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TEST SCENARIO** | **STEPS TO EXECUTE** | **TEST DATA** | **EXPECTED RESULT** | **ACTUAL RESULT** | **STATUS** |
| Verify user is able to run theapplication by login to the home page | 1. Click on the run.app 2. A link will be generated 3. click on the link provided to visit the home page | [**http://127.**](http://127/) **0.0.1:5000** | Home page is displayed | Home page is displayed | pass |
| Verify the user can see the home page and see the diseases | 1.Go to the homepage 2.Click on to the diseases  3. A predict button will be displayed to check the leaf diseases | [**http://127.**](http://127/) **0.0.1:5000** | Predict button page will be displayed | Predict button page will be displayed | pass |
| Verify the user can see the leaf  images by  clicking the | 1. Click the predict button 2. A list of images will be displayed | [**http://127.**](http://127/) **0.0.1:5000** | Images of the diseased leaves has to be displayed | Images of the diseased leaves has to displayed | pass |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| predict button | 3.Select a leaf image that has to be predicted 4.After the leaf is predicted, the leaf has to determine the diseases. |  |  |  |  |
| Verify the leaf disease is predicted correctly | 1.The information has to provide correct disease 2.if the disease is correct test case is passed,  or else the test case is fail. | [**http://127.**](http://127/) **0.0.1:5000** | Successfully predicted the disease and displays the fertilizer recommended | Have successfully predicted the disease and correctly recommended the fertilizer. | pass |

## User Acceptance Testing

### Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

### Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Resolution** | **Severity 1** | **Severity 2** | **Severity 3** | **Severity 4** | **Subtotal** |
| By Design | 6 | 4 | 2 | 3 | 15 |
| Duplicate | 1 | 0 | 3 | 0 | 4 |
| Not Reproduced | 0 | 0 | 1 | 0 | 1 |
| Skipped | 0 | 0 | 1 | 1 | 2 |
| Won't Fix | 0 | 1 | 2 | 2 | 5 |
| Totals | 7 | 5 | 9 | 6 | 27 |

### Test Case Analysis

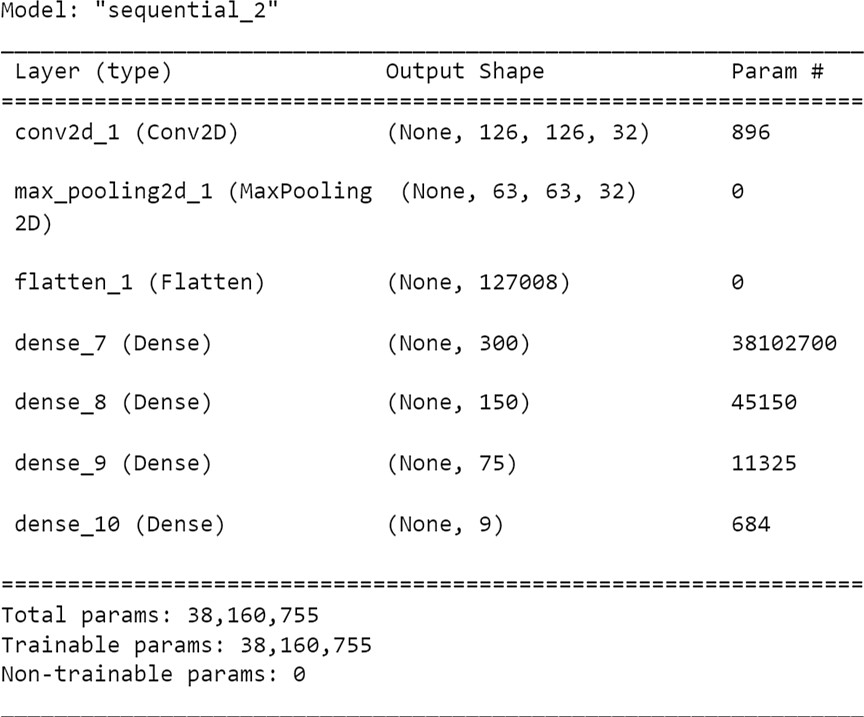
This report shows the number of test cases that have passed, failed, and untested

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Section** | **Total Cases** | **Not Tested** | **Fail** | **Pass** |
| Client Application | 5 | 0 | 0 | 5 |
| Security | 2 | 0 | 0 | 2 |
| Final Report Output | 4 | 0 | 0 | 4 |
| Version Control | 2 | 0 | 0 | 2 |

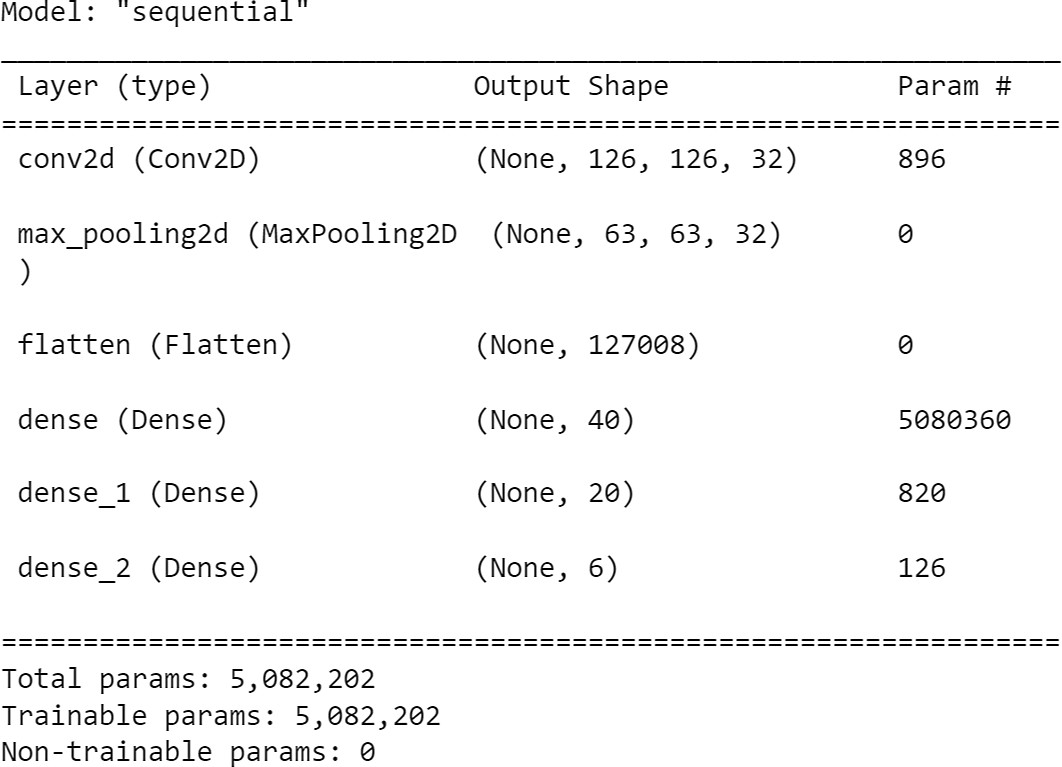
## RESULTS

### Performance Metrics

1. **PARAMETER: MODEL SUMMARY VEGETABLE-**

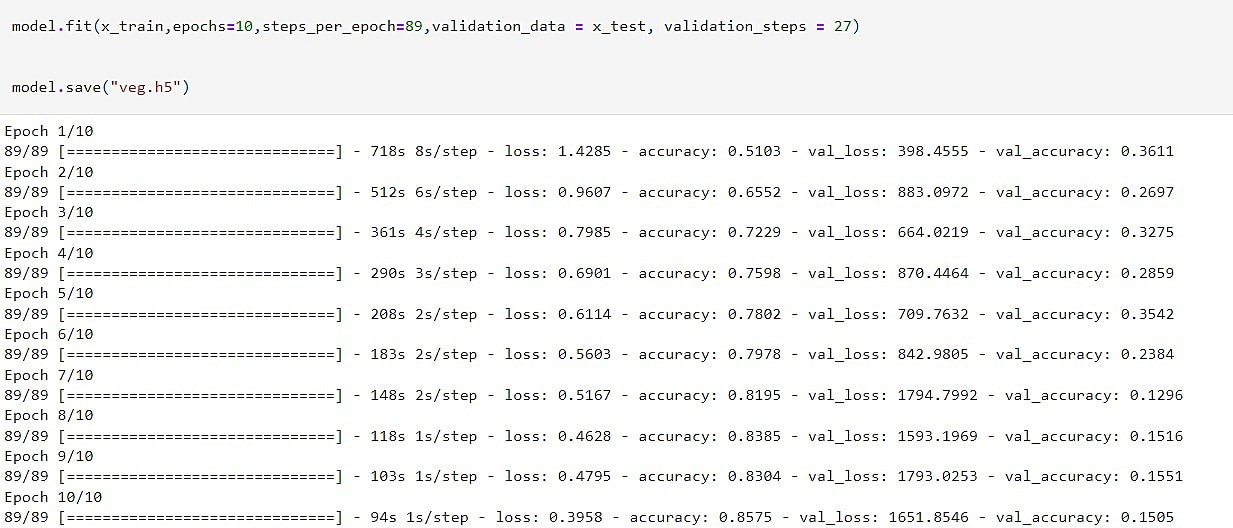


**FRUIT-**

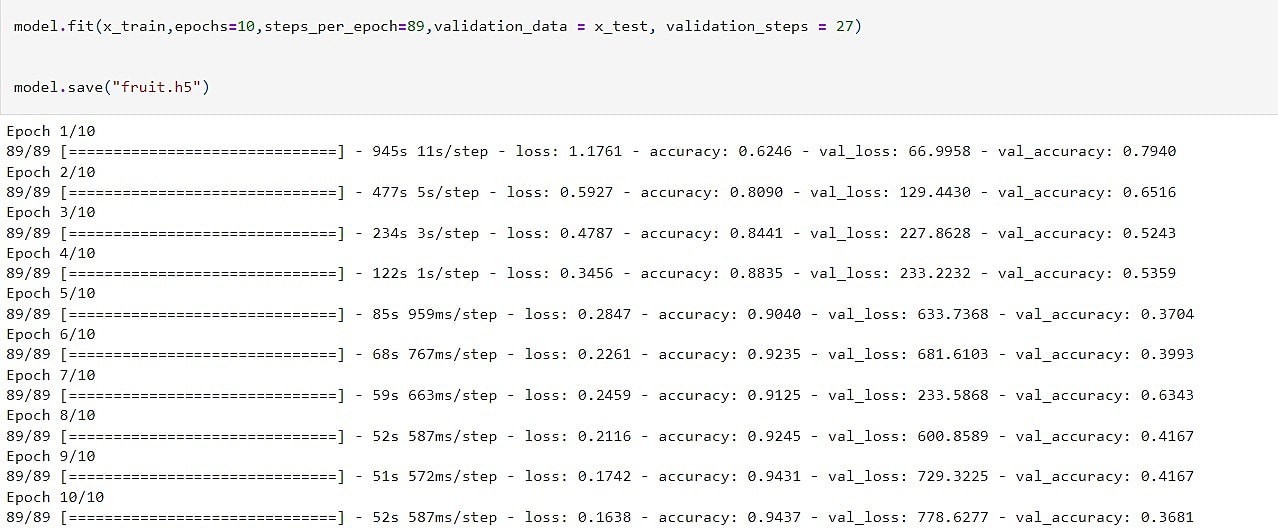


#### PARAMETER: ACCURACY

Training Accuracy - Validation Accuracy - **VEGETABLE-**



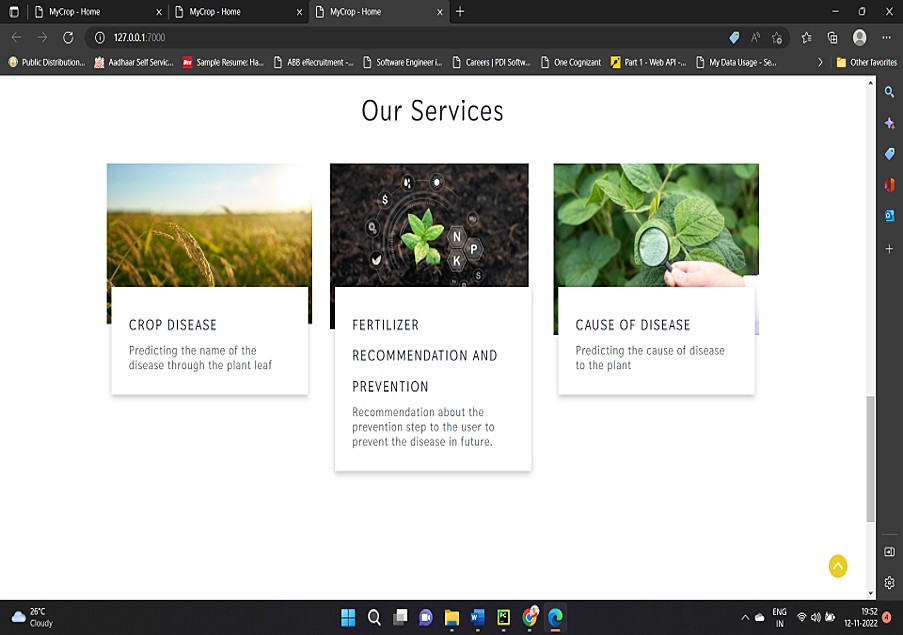
**FRUIT**

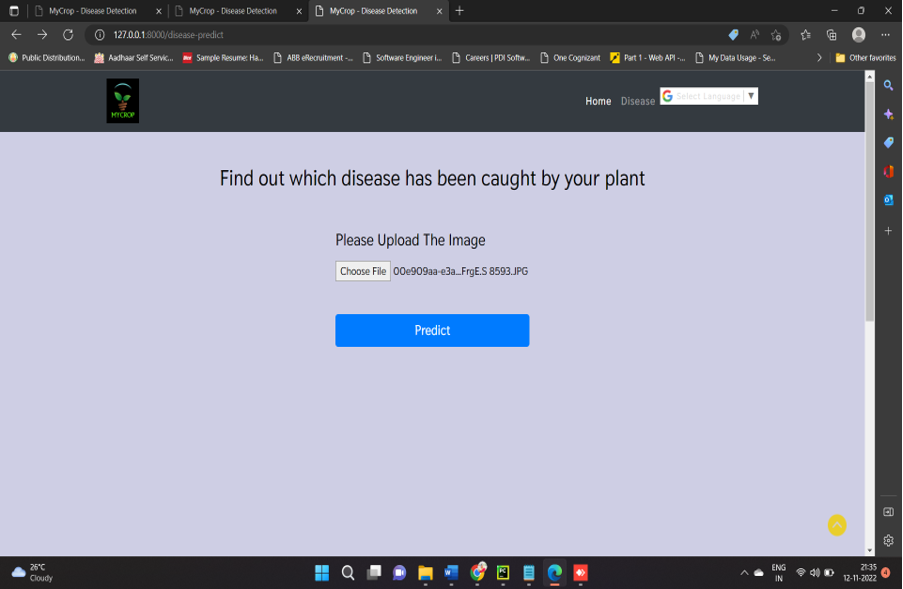


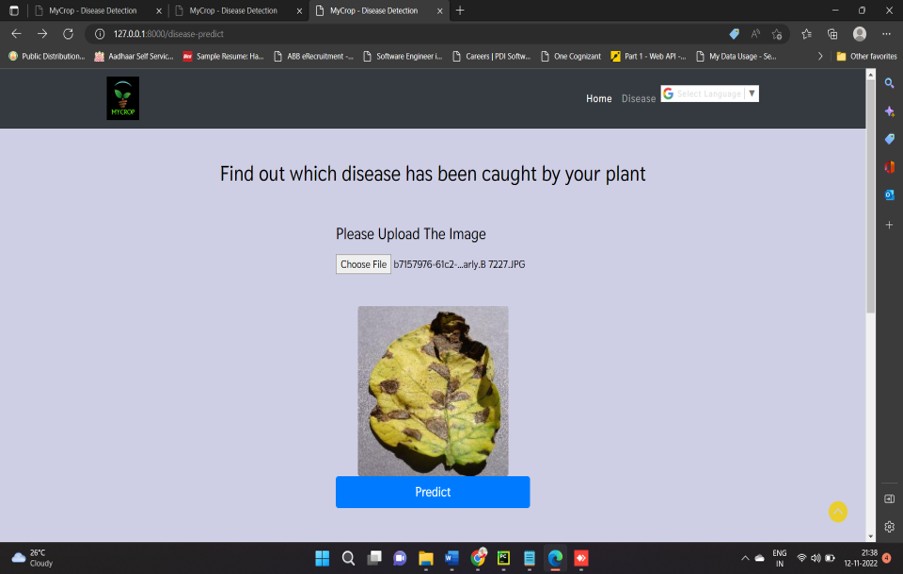
1. **Confidence Score (Only Yolo Projects) -** NOT A YOLO PROJECT

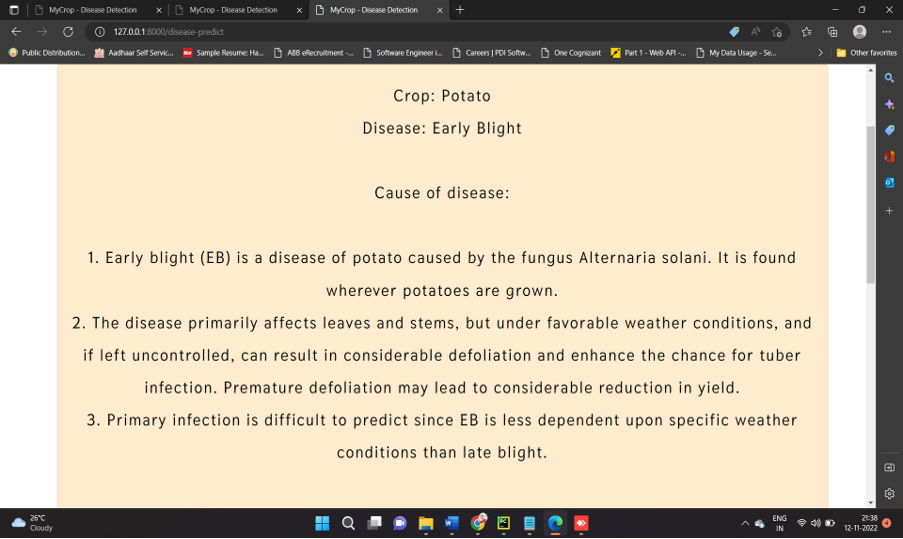
**OUTPUT SCREENSHOTS:**

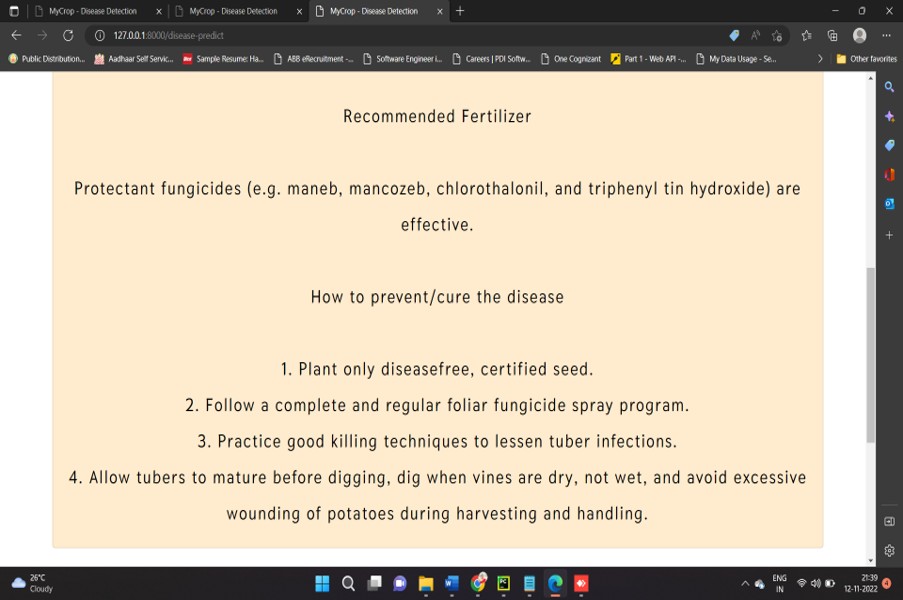












## 10.ADVANTAGES & DISADVANTAGES

**ADVANTAGES**

Farmers can interact with the portal build

* Interacts with the user interface to upload images of diseased leaf
* Our model-built analyses the Disease and suggests the farmer with fertilizers are to be used
* It is easy to maintain.
* It is user-friendly.
* The system can easily detect the leaf from the image.
* It will also detect which type of leaf it is.

The following are the areas where the plant disease detection system is used They are,

#### Agriculture

1. **Research and study**

#### AGRICULTURE

Crop disease are a major threat to food security, but their rapid identification remains difficult in many parts of the world due to the lack of the necessary infrastructure. The combination of increasing global smartphone penetration and recent advanced in computer vision made possible by deep learning has paved the way for smartphone-assisted disease diagnosis.

Overall, the approach of training deep learning models on increasingly large and publicly available image datasets presents a clear path toward smartphone-assisted crop disease diagnosis on a massive global state.

#### RESEARCH AND STUDY

In future, large dataset and different types of disease can be found easily by the growing machine learning technologies. Students and biologists can research about the new diseases in plants and help them to provide remedies and treatments based on the type of disease.

## DISADVANTAGES

* 1. More training samples **-** more speed of computing distances sensitive to irrelavant inputs so expensive testing everytime.
  2. It is slower in execution speed long training time .
  3. Sometimes it can predict the wrong disease which may cause difficulty to farmers.
  4. Recommending wrong fertilizers can damage the crops.

## CONCLUSION

We have proposed ban automated system to identify and classify the disease caused in plants at an earlier stage with pest management.to detect and identification of various diseases, we use the convolutional neural network (CNN) and deep learning. The result from can be used to identify the disease with high accurate and suggest solution . High performance model is obtained by using best hyper parameters and good training data . The final model will give high accuracy for the given data. An application to detect , controls , and monitor the plantdisease helps the farmer to reduce their work as well as time.This application helps the farmer to reduce their effort, and also helps in increasing the farm of production. The proposed method helps to find the plant disease and in monitoring the several environmental conditions the status of the leaf has been identified with the help of neural network classification . Then the environment circumstances such as temperature, humidity and moisture has been monitored the environmental condition is abnormal, then the pump will automatically.This project gives the executed results on different diseases classification techniques that can be used for plant leaf disease detection a. Therefore, related diseases for these plants were taken for identification. With very less computational efforts the optimum results were obtained, which also shows the efficiency of the proposed algorithm in recognition and classification of the leaf diseases. Another advantage of using this method is that the plant diseases can be identified at an early stage or the initial stage. By using this concept, the disease identification is done for all kinds of leafs and also the user can know the affected area of leaf in percentage by identifying the disease properly the user can rectify the problem very easy.

## FUTURE SCOPE

* + This system can be enhanced in future by using the trained model in android apps to make more feasible and efficiently.
  + In future, use of more advanced algorithms can beimplemented into the system to show high accuracy and less process time.
  + Using the camera we canimplement the system in continuous monitoring of crops and plants for detecting the texture of plants for more early detection of plants.
  + After the leaf undergoes detection, the disease is identified and check whether the leaf can be cured at certain conditions or not and fertilizersare recommended according to the leaf.

## APPENDIX

**Source Code**

#### Image Pre-processing

**from** keras.preprocessing.image **import** ImageDataGenerator

train\_datagen **=** ImageDataGenerator(rescale **=** 1.**/**255, shear\_range **=** 0.2,zoom\_range **=**

0.2,horizontal\_flip **=True**)

test\_datagen **=** ImageDataGenerator(rescale **=** 1)

#### FRUIT

x\_train**=** train\_datagen**.**flow\_from\_directory('/content/drive/MyDrive/fruit-dataset/fruit- dataset/train',batch\_size**=**32,target\_size**=**(128,128),

color\_mode**=**'rgb',class\_mode**=**'categorical')

x\_test **=** test\_datagen**.**flow\_from\_directory('/content/drive/MyDrive/fruit-dataset/fruit- dataset/test',batch\_size**=**32,target\_size**=**(128,128),

color\_mode**=**'rgb',class\_mode**=**'categorical')

#### VEGETABLE

x\_train**=** train\_datagen**.**flow\_from\_directory('/content/drive/MyDrive/Veg-dataset/Veg- dataset/train\_set',batch\_size**=**32,target\_size**=**(128,128),

color\_mode**=**'rgb',class\_mode**=**'categorical')

x\_test **=** test\_datagen**.**flow\_from\_directory('/content/drive/MyDrive/Veg-dataset/Veg- dataset/test\_set',batch\_size**=**32,target\_size**=**(128,128),

**Model Building For Fruit Disease Prediction**

**from** keras.models **import** Sequential

**from** keras.layers **import** Dense

**from** keras.layers **import** Convolution2D **from** keras.layers **import** MaxPooling2D **from** keras.layers **import** Flatten

**from** keras.preprocessing.image **import** ImageDataGenerator

train\_datagen **=** ImageDataGenerator(rescale **=** 1.**/**255, shear\_range **=** 0.2,zoom\_range **=**

0.2,horizontal\_flip **= True**)

test\_datagen **=** ImageDataGenerator(rescale **=** 1)

x\_train**=** train\_datagen**.**flow\_from\_directory('/content/drive/MyDrive/fruit-dataset/fruit- dataset/train',batch\_size**=**32,target\_size**=**(128,128),

color\_mode**=**'rgb',class\_mode**=**'categorical')

x\_test **=** test\_datagen**.**flow\_from\_directory('/content/drive/MyDrive/fruit-dataset/fruit- dataset/test',batch\_size**=**32,target\_size**=**(128,128),

color\_mode**=**'rgb',class\_mode**=**'categorical')

**from** tensorflow.keras.utils **import** Sequence model**=**Sequential()

model**.**add(Convolution2D(32,(3,3),input\_shape **=** (128,128,3),activation **=** 'relu')) model**.**add(MaxPooling2D(pool\_size **=** (2,2)))

model**.**add(Flatten()) model**.**add(Dense(40, 'relu')) model**.**add(Dense(20, 'relu')) model**.**add(Dense(6, 'softmax', ))

model**.**compile(optimizer**=**'adam', loss **=** "categorical\_crossentropy" , metrics **=**['accuracy'])

*# Visualize Model*

model**.**summary()

model**.**fit(x\_train,epochs**=**20,steps\_per\_epoch**=**89,validation\_data **=** x\_test, validation\_steps **=**

27)

model**.**save("fruit.h5")

**Model Building for Vegetable Disease Prediction**

**from** keras.models **import** Sequential

**from** keras.layers **import** Dense

**from** keras.layers **import** Convolution2D **from** keras.layers **import** MaxPooling2D **from** keras.layers **import** Flatten

**from** keras.preprocessing.image **import** ImageDataGenerator

train\_datagen **=** ImageDataGenerator(rescale **=** 1.**/**255, shear\_range **=** 0.2,zoom\_range **=**

0.2,horizontal\_flip **=True**)

test\_datagen **=** ImageDataGenerator(rescale **=** 1)

x\_train**=** train\_datagen**.**flow\_from\_directory('/content/drive/MyDrive/Veg-dataset/Veg- dataset/train\_set',batch\_size**=**32,target\_size**=**(128,128),

color\_mode**=**'rgb',class\_mode**=**'categorical')

x\_test **=** test\_datagen**.**flow\_from\_directory('/content/drive/MyDrive/Veg-dataset/Veg- dataset/test\_set',batch\_size**=**32,target\_size**=**(128,128),

color\_mode**=**'rgb',class\_mode**=**'categorical')

Found 11430 images belonging to 9 classes. Found 3416 images belonging to 9 classes.

**from** tensorflow.keras.utils **import** Sequence model**=**Sequential()

model**.**add(Convolution2D(32,(3,3),input\_shape **=** (128,128,3),activation **=** 'relu')) model**.**add(MaxPooling2D(pool\_size **=** (2,2)))

model**.**add(Flatten())

model**.**add(Dense(300, 'relu')) model**.**add(Dense(150, 'relu')) model**.**add(Dense(75, 'relu')) model**.**add(Dense(9, 'softmax', ))

model**.**compile(optimizer**=**'adam', loss **=** "categorical\_crossentropy" , metrics **=**['accuracy']) model**.**summary()

model**.**fit(x\_train,epochs**=**20,steps\_per\_epoch**=**89,validation\_data **=** x\_test, validation\_steps **=**

27)

model**.**save("veg.h5")

#### TEST BOTH THE MODEL

from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing.image import img\_to\_array from tensorflow.keras.models import load\_model

import numpy as np

model = load\_model('fruit.h5')

img = image.load\_img(r"/home/wsuser/work/Dataset Plant Disease/fruit-dataset/fruit- dataset/test/Apple healthy/0a553fc0-fc2c-4598-baba-

3bc10191447c RS\_HL 5969.JPG", target\_size = (128,128)) x=image.img\_to\_array(img)

x=np.expand\_dims(x,axis=0) result=model.predict(img) print(result)

**Build Python Code**

from future import division, print\_function import os

import numpy as np import cv2

# Keras

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing.image import img\_to\_array

# Flask utils

from flask import Flask, request, render\_template from werkzeug.utils import secure\_filename

app = Flask( name ) MODEL\_PATH = 'fruit.h5' MODEL LOADING

model = load\_model(MODEL\_PATH) model.make\_predict\_function() default\_image\_size = (128, 128)

abels=["Apple Black\_rot","Apple healthy","Corn\_(maize) health y","Corn\_(maize) Northern\_Leaf\_Blight","Peach Bacterial\_spot"," Peach healthy"]

def convert\_image\_to\_array(image\_dir):

try:

image = cv2.imread(image\_dir) if image is not None:

image = cv2.resize(image, default\_image\_size) return img\_to\_array(image)

else:

return np.array([]) except Exception as e: print(f"Error : {e}") return None

def model\_predict(file\_path, model):

x = convert\_image\_to\_array(file\_path) x = np.expand\_dims(x, axis=0)

preds = model.predict(x) return preds

@app.route("/", methods=['GET']) def index():

return render\_template("index.html", query="")

@app.route("/", methods=['GET', 'POST']) def upload():

if (request.method == 'POST'):

f = request.files['file']

basepath = os.path.dirname( file )

file\_path=os.path.join(basepath,'uploads',secure\_filename(f.filename)) f.save(file\_path)

preds = model\_predict(file\_path, model) preds = np.argmax(preds)

result = labels[preds]

return render\_template('index.html', prediction\_text=result) return None

if name == " main ": app.run(debug=True)

**Build HTML Pages**

**Index.html**

{% extends 'layout.html' %}

{% block body %}

<!-- banner -->

<section class="banner\_w3lspvt" id="home">

<div class="csslider infinity" id="slider1">

<div class="banner-top1">

<div class="overlay">

<div class="container">

<div class="w3layouts-banner-info text-center">

<h3 class="text-wh">MyCrop-Plant Disease Prediction</h3>

<h4 class="text-wh mx-auto my-4"><b>Get informed decisions about your farming strategy.<br>In Your Own Language.</b></h4>

<h4 class="text-wh mx-auto my-4"><strong> Here are some questions we'll answer</strong></h4>

<p class="text-li mx-auto mt-2">

1. Which disease do your crop have? <br>
2. What cause the disease to plant? <br>
3. How to prevent the disease?<br>
4. How to cure the disease?<br> 5.Fertilizer Recommended</p>

</div>

</div>

<div class="w3layouts-banner-info text-center">

<h3 class="text-wh">MyCrop-Plant Disease Prediction</h3>

<h4 class="text-wh mx-auto my-4"><b>Get informed decisions about your farming strategy.<br>In Your Own Language.</b></h4>

<h4 class="text-wh mx-auto my-4"><strong> Here are some questions we'll answer</strong></h4>

<p class="text-li mx-auto mt-2">

1. Which disease do your crop have? <br>
2. What cause the disease to plant? <br>
3. How to prevent the disease?<br>
4. How to cure the disease?<br> 5.Fertilizer Recommended</p>

</div>

</div>

</div>

</div></div>

</section>

<!-- //banner -->

<!-- core values -->

<section class="core-value py-5">

<div class="container py-md-4">

<h3 class="heading mb-sm-5 mb-4 text-center"> About Us</h3>

<div class="row core-grids">

<div class="col-lg-6 core-left"><br>

<img src="{{ url\_for('static', filename='images/13.jpg') }}" class="img-fluid" alt="" /></div>

<div class="col-lg-6 core-right">

<h3 class="mt-4">Improving Agriculture, Improving Lives, Cultivating Crops To Make Farmers Increase Profit.</h3>

<p class="mt-3">We use state-of-the-art machine learning and deep learning technologies to

<div class="w3layouts-banner-info text-center">

<h3 class="text-wh">MyCrop-Plant Disease Prediction</h3>

<h4 class="text-wh mx-auto my-4"><b>Get informed decisions about your farming strategy.<br>In Your Own Language.</b></h4>

<h4 class="text-wh mx-auto my-4"><strong> Here are some questions we'll answer</strong></h4>

<p class="text-li mx-auto mt-2">

1. Which disease do your crop have? <br>
2. What cause the disease to plant? <br>
3. How to prevent the disease?<br>
4. How to cure the disease?<br> 5.Fertilizer Recommended</p>

</div>

</div>

</div>

</div></div>

</section>

<!-- //core values -->

<!-- Products & Services -->

<section class="blog py-5">

<div class="container py-lg-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-md-5 pb-md-5 pb-5">

<a href="{{ url\_for('home') }}">

<img src="{{ url\_for('static', filename='images/s35.jpg') }}" class="img-fluid" alt="" />

<div class="blog-info">

<h4>Crop Disease</h4>

<p class="mt-1">Predicting the name of the disease through the plant leaf</p>

</div></a>

<br><br><br>

</div><div class="col-lg-4 col-md-6 blog-middle mb-lg-0 mb-md-5 pb-md-5 pb-5">

<a href="{{ url\_for('home') }}">

<img src="{{ url\_for('static', filename='images/s6.jpg') }}" class="img-fluid" alt="" />

<div class="blog-info">

<h4> Fertilizer Recommendation and Prevention</h4>

<p class="mt-1">Recommendation about the prevention step to the user to prevent the disease in future.</p>

</div>

</a>

<br>

<br>

</div>

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

<a href="{{ url\_for('disease\_prediction') }}">

<img src="{{ url\_for('static', filename='images/s7.jpg') }}" class="img-fluid" alt="">

<div class="blog-info">

<h4>Cause of Disease</h4>

<p class="mt-1">Predicting the cause of disease to the plant</p>

</div>

</a>

</div>

</div>

</section>

<style>

</style>

<!-- //Products & Services -->

</html>

{% endblock %}

**Disease-result.html**

{% extends 'layout.html' %}

{% block body %}

<div class="container py-2 mx-auto my-50 h-10 " style="margin: 9rem;"

<div class="row">

<div class="col-sm py-2 py-md-3">

<div class="card card-body" style="justify-content: center; background- color:blanchedalmond">

<p class="text-center" style="color: black; font-size: 22px;">{{ prediction }}

</p>

</div>

</div>

</div>

</div>

</a>

<br>

<br>

</div>

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

<a href="{{ url\_for('disease\_prediction') }}">

<img src="{{ url\_for('static', filename='images/s7.jpg') }}" class="img-fluid" alt="">

<div class="blog-info">

<h4>Cause of Disease</h4>

<p class="mt-1">Predicting the cause of disease to the plant</p>

</div>

</a>

</div>

</div>

</section>

<style>

</style>

<!-- //Products & Services -->

</html>

{% endblock %}

#### Disease.html

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

<style> html body {

background-color: rgb(206, 206, 228);

}

</style>

</a>

<br>

<br>

</div>

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

<a href="{{ url\_for('disease\_prediction') }}">

<img src="{{ url\_for('static', filename='images/s7.jpg') }}" class="img-fluid" alt="">

<div class="blog-info">

<h4>Cause of Disease</h4>

<p class="mt-1">Predicting the cause of disease to the plant</p>

</div>

</a>

</div>

</div>

</section>

<style>

</style>

<!-- //Products & Services -->

</html>

{% endblock %}

## GitHub & Project Demo Link

**GitHub Link:**

[**https://github.com/IBM-EPBL/IBM-Project-193-1658220651**](https://github.com/IBM-EPBL/IBM-Project-193-1658220651)