FERTILIZERS RECOMMENDATION SYSTEM FOR DISEASE PREDICTION

A PROJECT REPORT

Submitted by

TEAM ID : PNT2022TMID42096

PRIYADHARSHINI S 623319104001

VINITH R 623319104002

KEERTHIKA P 623319106003

THENMOZHI R 623319106005

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

VETRI VINAYAHA COLLEGE OF ENGINEERING AND TECHNOLOGY

THOTTIAM, TRICHY

TABLE OF CONTENTS

1.INTRODUCTION

1.1 Overview:

In 2019, the United Nations estimated 2 billion increase in the worlds' population by next 30 years, a significant increase of nearly 25%. According to the report of the Food and Agricultural Organization (FAO), to feed this population, about 70-90% more food will be required. Of the total agricultural crop production worldwide, damage of nearly 16% has been caused by the microbial diseases.

In order to minimize the occurrence of diseases as well as maximizing the productivity and ensuring agricultural sustainability, there is a need for advanced disease detection in preventing damages to crops. Hence, predetermining plant diseases and their prevention have raised a great interest in researchers. Diseases prediction in crops depends on various environmental and weather conditions, under which a pathogen can survive. When pathogen comes in contact with a susceptible host, it can infect and can cause severe losses to the agriculture production.

The diseases in plants cause a drop in the quality and quantity of the agricultural output. One of the most common diseases is fungi, present in the plant leaves. Fungi is the most diverse group of plant pathogens, accounting for over 70-80% of plant diseases. There are over 20,000 species of fungi that are parasitic and responsible for infections in crops and plants, thereby the quality of leaves, fruits, stem, vegetables, and their products gets suffered.

There are two key factors 'Disease' and 'Disorder' that affect the crops and their products. Disease, the biotic factors, are caused either by fungi or by bacteria or algae, and the disorder are the abiotic factors caused by the atmospheric conditions (temperature, rainfall, moisture etc.). These infectious crop diseases, if not treated timely, can significantly reduce the yield, thus endangering global food security.

Early disease diagnosis and providing the control measures can help the farmers to save the crops. These measures include direct or indirect disease identification methods. Direct detection methods mainly include laboratory-based techniques, while indirect methods use optical sensors for thermography, fluorescence imaging, and hyper spectral techniques.

The limitation of various optical sensing techniques is the large amount of data acquired and the complexity of the data collected. In order to effectively utilize these techniques, it requires high setup and computational costs along with the knowledge of data analytics and statistical methods Manual prediction of potato disease is time-consuming, hard and expensive, while the computerized system is cost effective and more efficient.

Recently, machine Learning (ML) system are extensively used to automate the processes. Machine learning can be an efficient way to monitor plant health status and early plant disease predictions. Every disease has some noticeable symptoms on the plants' leaves.

These symptoms, for example, a visible pattern of the affected leaves help to predict the disease. In this way, machine learning (ML) provides a solution to agricultural productivity issues and guarantees food safety

The occurrence of plant diseases has a negative impact on agricultural production. If plant diseases are not discovered in time, food insecurity will increase.

Early detection is the basis for effective prevention and control of plant diseases, and they play a vital role in the management and decisionmaking of agricultural production. In recent years, plant disease identification has been a crucial issue. Disease-infected plants usually show obvious marks or lesions on leaves, stems, flowers, or fruits. Generally, each disease or pest condition presents a unique visible pattern that can be used to uniquely diagnose abnormalities.

Usually, the leaves of plants are the primary source for identifying plant diseases, and most of the symptoms of diseases may begin to appear on the leaves .The best solution to the problem is to identify the disease of the plant so that precautionary steps can be taken to safeguard the same. This paper implements the concept of applying convolutional neural network implementation to the detection of leaf disease in the plant and suggests a suitable solution to the farmer to recover the same.

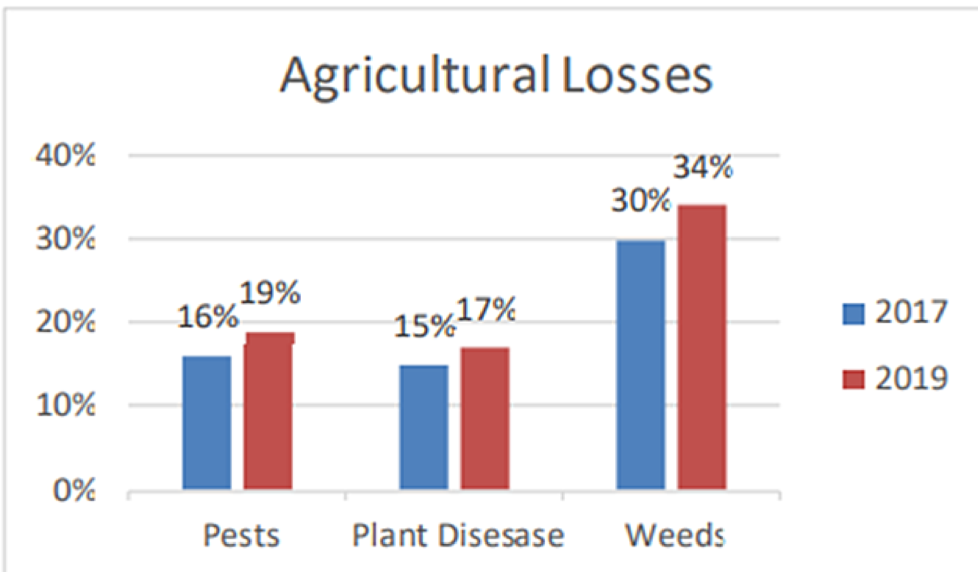


FIGURE 1.1 Agricultural Losses

Farmers with less experience may misjudgment and use drugs blindly during the identification process. Quality and output will also bring environmental

pollution, which will cause unnecessary economic losses. To counter these challenges, research into the use of image processing techniques for plant disease recognition has become a hot research topic.

Deep learning models became an attractive and efficient alternative for leaf disease detection when compared with traditional models. The rationale behind this is that the deep models could handle large datasets and support for pre-trained models. For image-based detection of diseases and classification using leafs as input, deep learning models explored different crops in agriculture.

This study has assumed significance in the wake of Precision Agriculture (PA) efforts across the globe. Technology driven approach in detection of crop diseases lead to innovations in early identification of problems in agricultural crops and take necessary steps. Many researchers. Most of the research papers use Convolutional Neural Network (CNN) architectures for deep learning-based disease detection. CNN is used in and for disease detection in Maize plants. In and also CNN is used for disease detection using plant leaves' images. There are some research pertaining to leaf disease datasets and the impact of size as explored in.

Deep CNN is used in for rice diseases prediction. From the literature, it is understood that the existing methods are based on CNN for deep learning. However, there is need for novel architectures with pre-trained models and the existing models have not used transfer learning. This paper uses transfer learning with a deep learning framework with pre-trained deep models to classify diseases of Apple crop.

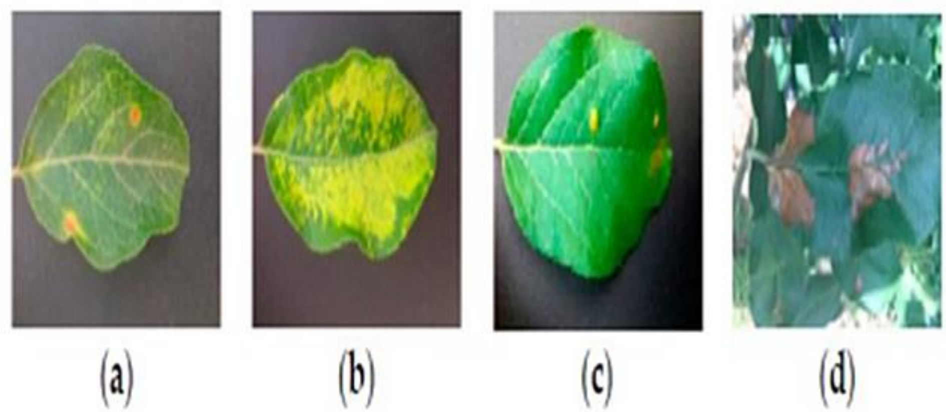


FIGURE 1.2 The four type of leaf disease

(a) Leaf spots lesions are one kind of disease,

(b) The yellow color lesion on leaf is called Mosaic disease,

(c) The yellow color spot on the leaf is the symptom of Rust disease and

(d) Brown spot disease

1.2 Purpose:

This project is used to test the fruits and vegetables sample and identify the different diseases. Also, this project recommends fertilizers for predicted diseases.

2.LITERATURE SURVEY

2.1.EXISTING METHOD:

Develop An Automatic Diagnosis Method to differentiate various Wheat Diseases.

Disease diagnosis based on the detection of early symptoms is a usual threshold taken into account for integrated pest management strategies. Early phytosanitary treatment minimizes yield losses and increases the efficacy and efficiency of the treatments. However, the appearance of new diseases associated to new resistant crop variants complicates their early identification delaying the application of the appropriate corrective actions.

The use of image based automated identification systems can leverage early detection of diseases among farmers and technicians but they perform poorly under real field conditions using mobile devices. A novel image processing algorithm based on candidate hot-spot detection in combination with statistical inference methods is proposed to tackle disease identification in wild conditions.

This work analyses the performance of early identification of three European endemic wheat diseases – septoria, rust and tan spot. The analysis was done using 7 mobile devices and more than 3500 images captured in two pilot sites in Spain and Germany during 2014, 2015 and 2016.

Segment the leaf area and lesion region area.

Fungi-caused diseases in sugarcane are the most predominant diseases which appear as spots on the leaves. If not treated on time, causes the severe loss. Excessive use of pesticide for plant diseases treatment increases the cost and environmental pollution so their use must be minimized.

This can be achieved by targeting the diseases places, with the appropriate quantity and concentration of pesticide by estimating disease severity using image processing technique. Simple threshold and Triangle thresholding methods are used to segment the leaf area and lesion region area respectively.

Detection of Disease in Tomato Leaf

In the agriculture sector, one of the major problems in the plants is its diseases. The plant diseases can be caused by various factors such as viruses, bacteria, fungus etc. Most of the farmers are unaware of such diseases. That's why the detection of various diseases of plants is very essential to prevent the damages that it can make to the plants itself as well as to the farmers and the whole agriculture ecosystem.

Regarding this practical issues, this research aimed to classify and detect the plant's diseases automatically especially for the tomato plant. As per the hardware requirement, Raspberry Pi is the major computing unit. Image processing is the key process of the project which includes image acquisition, adjusting image ROI, feature extraction and convolution neural network (CNN) based classification. Here, Python programming language, OPENCV library is used to manipulate raw input image.

To train on CNN architecture and creating a machine learning model that can predict the type of diseases, image data is collected from the authenticated online source. As the result , few diseases that usually occurs in tomato plants such as Late blight (training 100, test 21), Gray spot (training 95, test 18) and bacterial canker (training 90, test 21) are detected.

Automatic technique is used for detecting little leaf disease found in pine tree

Agricultural productivity is something on which economy highly depends. This is the one of the reasons that disease detection in plants plays an important role in agriculture field, as having disease in plants are quite natural. If proper care is not taken in this area then it causes serious effects on plants and due to which respective product quality, quantity or productivity is affected. For instance a disease named little leaf disease is a hazardous disease found in pine trees in United States.

Detection of plant disease through some automatic technique is beneficial as it reduces a large work of monitoring in big farms of crops, and at very early stage itself it detects the symptoms of diseases i.e. when they appear on plant leaves. This paper presents an algorithm for image segmentation technique which is used for automatic detection and classification of plant leaf diseases.

It also covers survey on different diseases classification techniques that can be used for plant leaf disease detection. Image segmentation, which is an important aspect for disease detection in plant leaf disease, is done by using genetic algorithm.

2.2 REFERENCES:

[1] Z. Li et al., “Non-invasive plant disease diagnostics enabled by smartphone-based fingerprinting of leaf volatiles,” Nature Plants, vol. 5, no. 8, pp. 856–866, Aug. 2019.

[2] G. Litjens et al., “A survey on deep learning in medical image analysis,” Med. Image Anal., vol. 42, pp. 60–88, Dec. 2017.

[3] W. Min, S. Jiang, L. Liu, Y. Rui, and R. Jain, “A survey on food computing,” ACM Comput. Surv., vol. 52, no. 5, pp. 92:1–92:36, 2019.

[4] E. Moen, D. Bannon, T. Kudo, W. Graf, M. Covert, and D. Van Valen, “Deep learning for cellular image analysis,” Nature Methods, vol. 16, no. 12, pp. 1233–1246, 2019.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Neural Inf. Process. Syst., 2012, pp. 1106–1114.

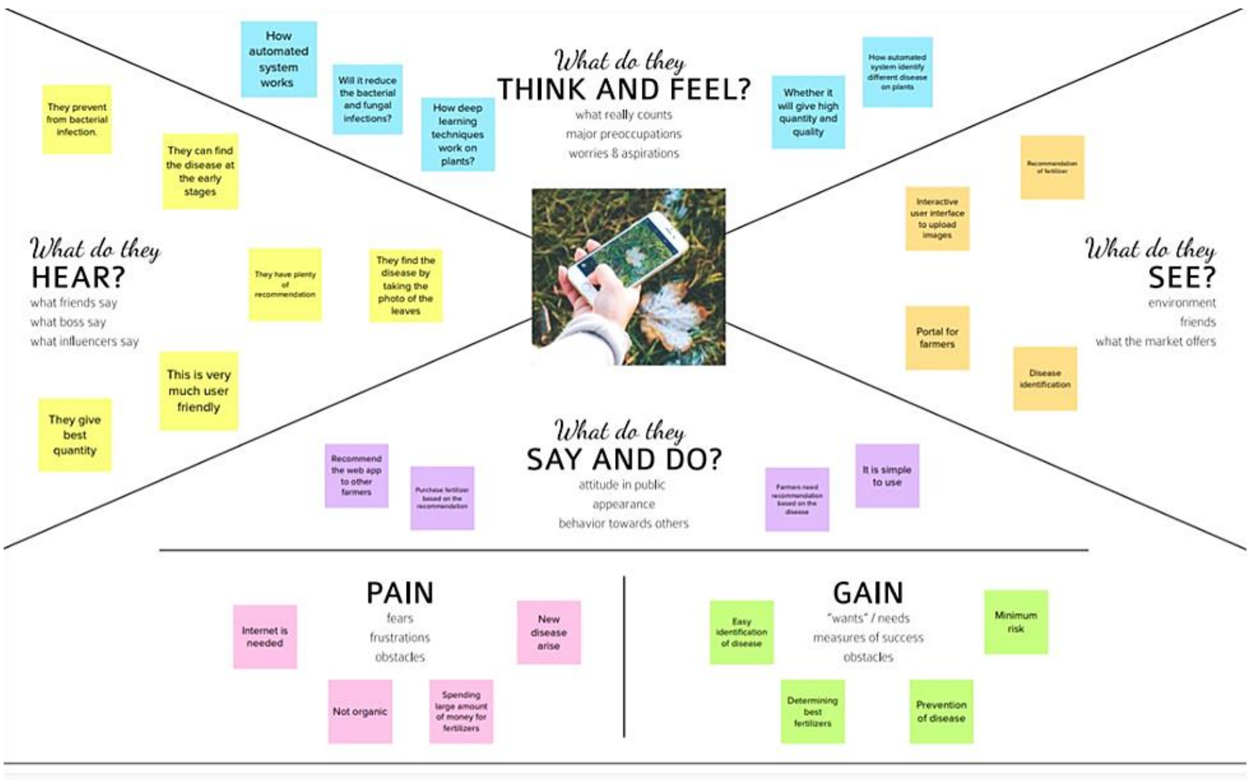
2.3.PROBLEM STATEMENT DEFINITION:

Agriculture is the most important sector in today’s life. Doing agriculture is the very hard in current scenario because of many natural disasters are happening every day. Each crop is detected by many different types of plant pathogens, causing different diseases and some of them are significant and occur most widely around the world. Crop diseases are 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. Identifying the disease in early stage in early stage is very important and easy to cure that.

I am	A farmer trying to grow crops he is very controlled in the application of fertilizer to the crops.
I am trying to	Use recent technologies are used to identify the diseases and suggest the precautions that can be taken for those diseases and trying increase the quantity and maximize the crop yield.
But	The technology that can help me a lot to predict the disease but we can’t diagnose the disease and use the right fertilizer.
Because	I don’t want to spoil the soil quality and crops quality.
Which makes me feel	Early disease diagnosis and providing the control measures can help the farmer to save the crops.

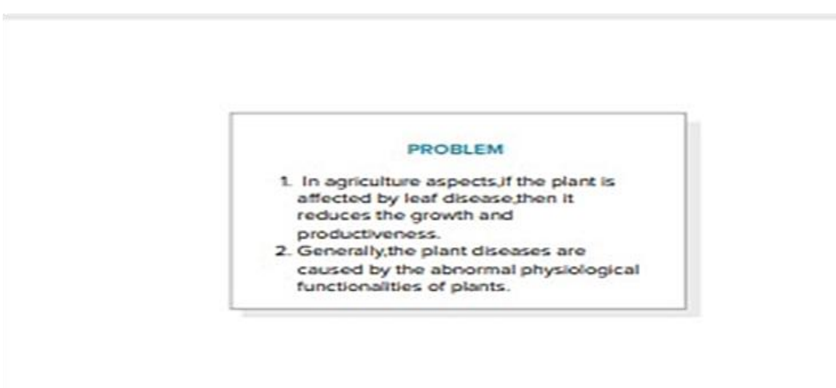
3.IDEATION & PROPOSED SOLUTION

3.1.EMPATHY MAP CANVAS:

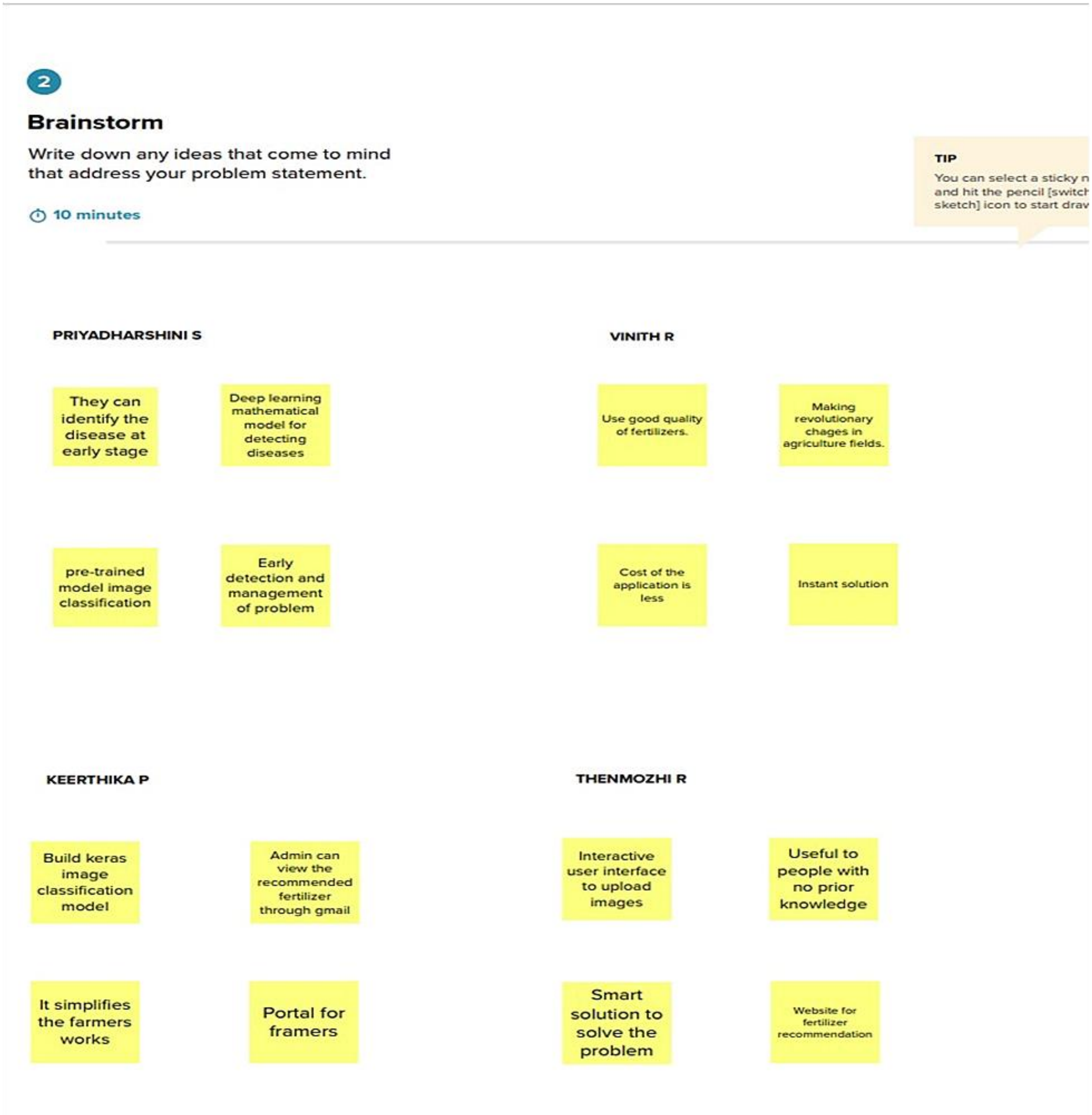


3.2.BRAINSTORMING:

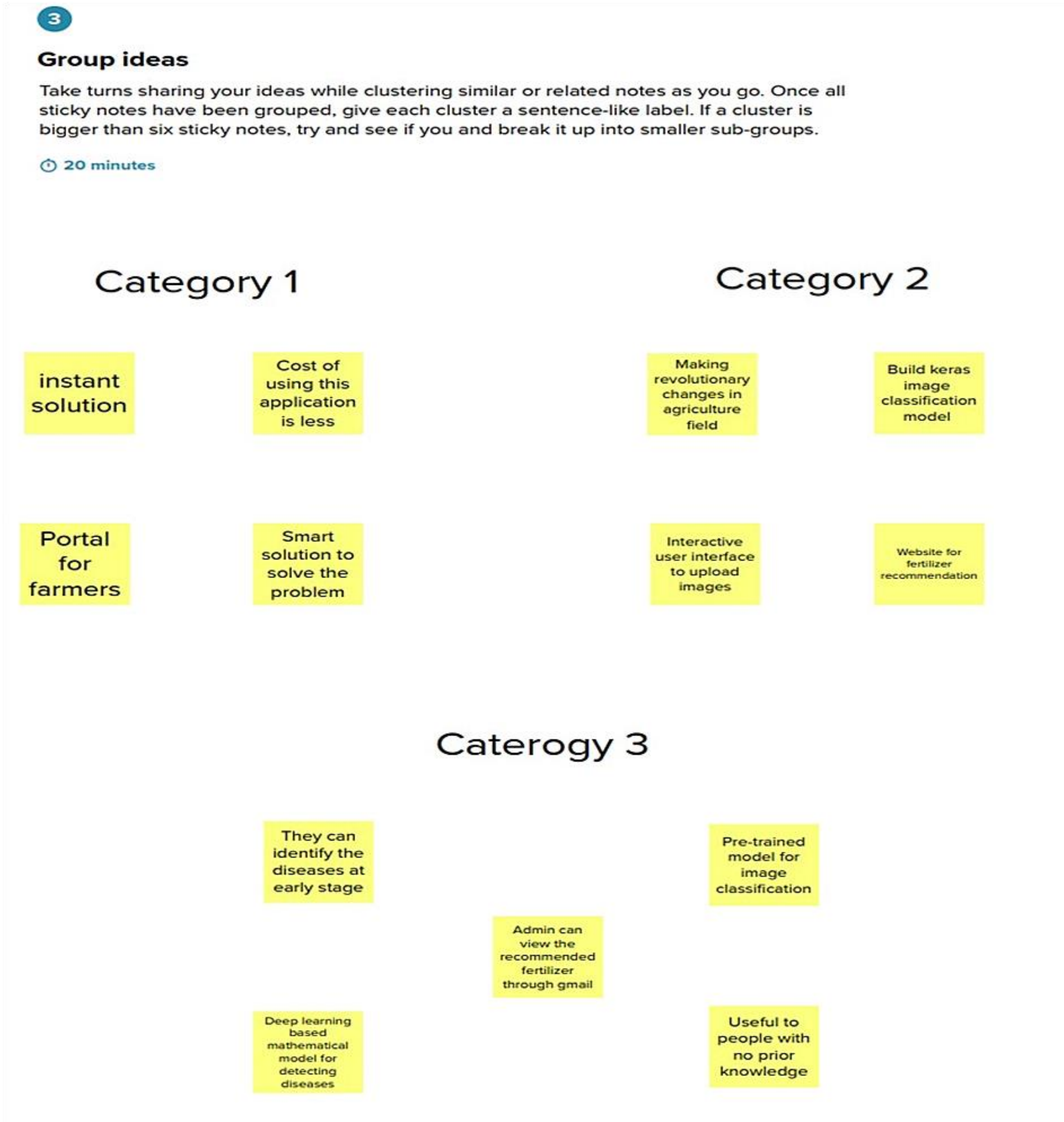
Problem Statements:



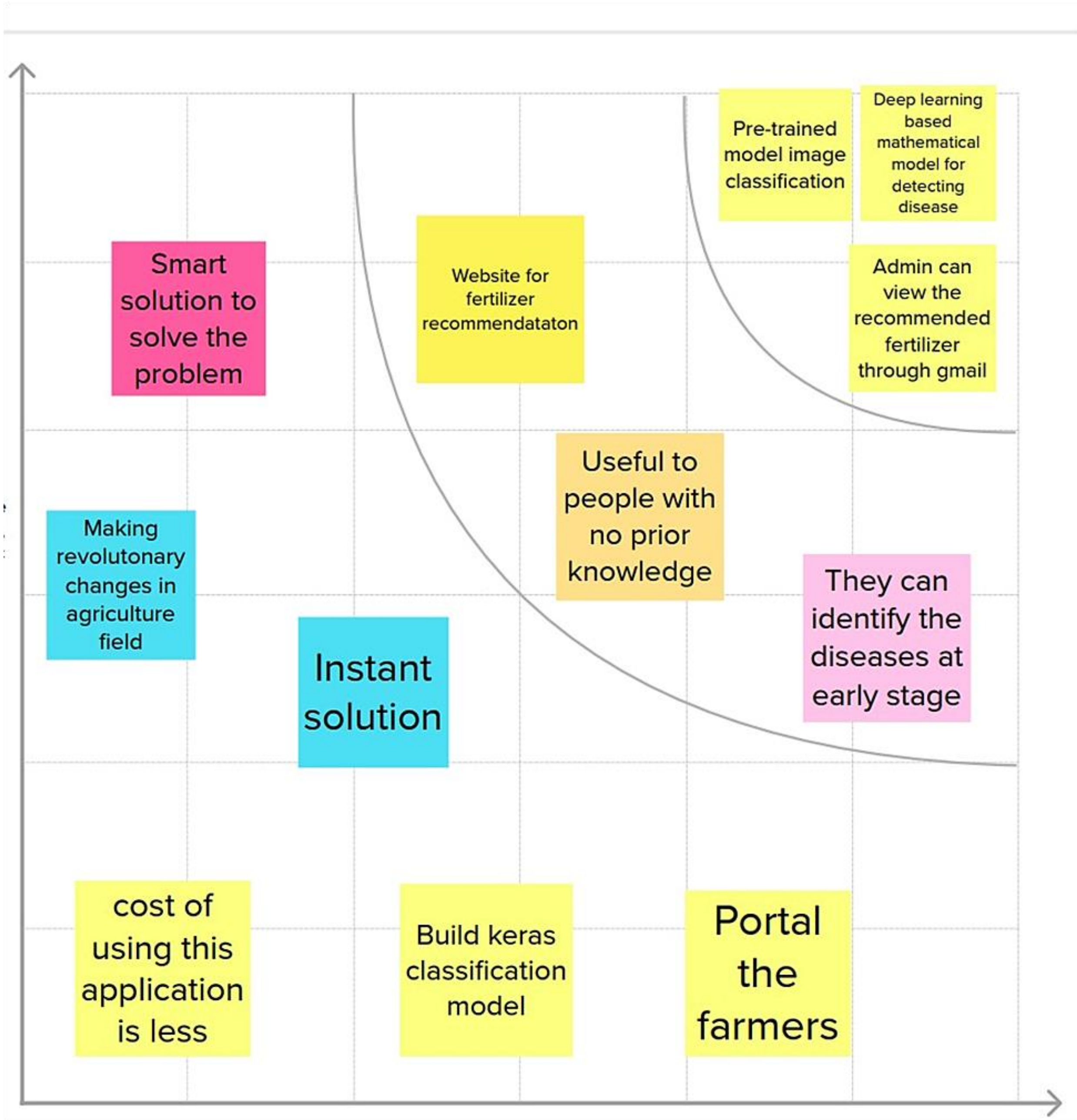
Brainstorming:



Group ideas:


3
Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes
Category 1
instant solution
Cost of using this application is less
Portal for farmers
Smart solution to solve the problem
Category 2
Making revolutionary changes in agriculture field
Build keras image classification model
Interactive user interface to upload images
Website for fertilizer recommendation
Caterogy 3
They can identify the diseases at early stage
Pre-trained model for image classification
Admin can view the recommended fertilizer through gmail
Deep learning based mathematical model for detecting diseases
Useful to people with no prior knowledge

Priortize :



Pre-trained model image classification
Deep learning based mathematical model for detecting disease
Smart solution to solve the problem
Website for fertilizer recommendataton
Admin can view the recommended fertilizer through gmail
Useful to people with no prior knowledge
Making revolutonary changes in agriculture field
They can identify the diseases at early stage
Instant solution
cost of using this application is less
Build keras classification model
Portal the farmers

3.3.PROPOSED SOLUTION:

S.No	Parameter	Description
1.	Problem statement (Problemto be solved)	To make an efficient use of Machine Learning Algorithm which reduces time and cost Farmer to detect the plant disease,its effect on crop yieldandsuggest the pesticides for plant disease.
2.	Idea/Solution description	Ourresearch aims to solve theproblem of detecting and preventing diseases of agricultural crops. To determine the optimal architecture for deep learning, we considered several models. As a source of the training data, we use the plant villageopen database for this approach automatic classifier Convolutional NeuralNetworks(CNN) model will be used for classification based on learning with some training samples. The developed model is deployed as a web Application which detect 15 types of disease among plants viz.Tomato,Potato and Pepper.

3.	Novelty/Uniqueness	This web application can suggest good fertilizer for thedisease in the plant by recognizing the image.
4.	Social Impact/Customer satisfaction	1. To designsuch system thatcan detect crop disease and pest accurately. 2. Create database of insecticides for respective pestand disease. 3. To provide remedy for the disease that is detected.
5.	Business Model (RevenueModel)	1. Disease prediction in plantis a more important factor infarmer industry and it let to economic development. 2. It is required for the growthof better quality good products.
6.	Scalability of the Solution	Deep learning techniques areused to identify the disease and suggest the precaution thatcan be taken for those disease.

3.4 Problem Solution Fit:

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Farmer is the first customer.crop disease are major threat to food security.plants are affected by leaf disease then it reduce the growth.finding the leaf disease and recommended the suitable fertilizer for the disease leaf.

6. CUSTOMER CONSTRAINTS CC

Capturing the image in a required pixels to get a accurate prediction of disease in the leaf.

5. AVAILABLE SOLUTIONS AS

CNN algorithm is used to predict leaf disease and recommended the fertilizers.

Merits:
Monitoring of large fields.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Collecting the datasets,then we have to scan the leaf and analyze the disease through pattern matching of the current datasets then the suitable fertilizer is recommended for te disease.

9. PROBLEM ROOT CAUSE RC

- Lack of affected plant.
- They did not know about the plant.
- Also insects on the plants can spread the disease.

7. BEHAVIOUR BE

Farmers implements scan the disease leaf to predict the disease and recommend the fertilizers.this technique is used to accuracy of leaf disease prediction and more flexible.

Focus on J&P, tap into BE, understand RC

3.TRIGGERS

Leaves are affected by bacteria, fungi and so on. Using CNN algorithm classifies the lead image as normal or affected.Then recommended the fertilizers.

4. EMOTIONS: BEFORE / AFTER EM

Before: The farmer did not identify the plant disease accurately.so, they will lose the field.
After: Our project recommended the fertilizer at its earliest stage.

10. YOUR SOLUTION

In this problem solution a Convolution Neural Network in Deep Learning based approach is proposed for predicting lead disease.This approach was evaluated with actual datasets collected from the images while capturing the crops. The evaluation process is conducted with manually labeled data and the proposed active deep learning shows a favorable performance.

The accuracy of leaf disease prediction is to be above 95% using the Neural Network algorithm.

8.CHANNELS of BEHAVIOR CH

Online:
Information about the disease leaf and recommended fertilizers.

Offline:
People trying to identify the disease by the quality of the leaf is difficult.

4.REQUIREMENT ANALYSIS

4.1Functional Requirement:

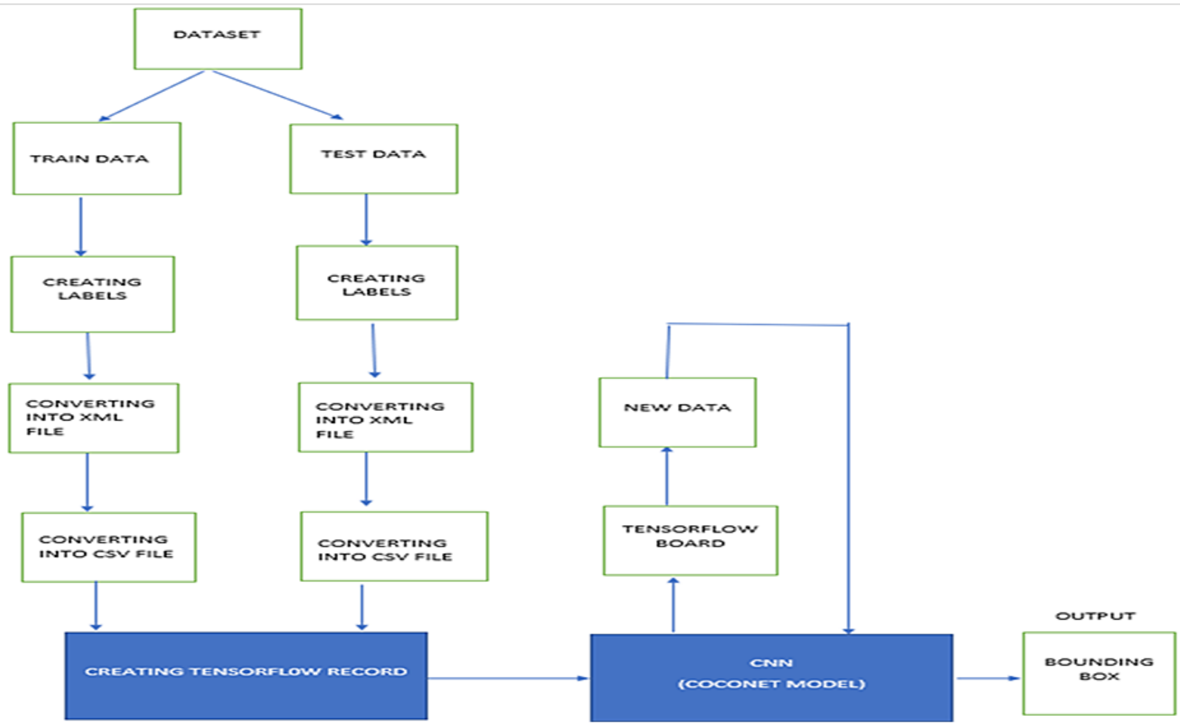
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Capturing image	Capture the imageof the leafand check the parameter of thecapture image.
FR-2	Image processing	Upload the imagefor the prediction of the disease in the leaf.
FR-3	Leaf identification	Identify the leaf and predictthe disease in the leaf.
FR-4	Image description	Suggesting the bestfertilizer for thedisease.

4.2 Non-Functional Requirement:

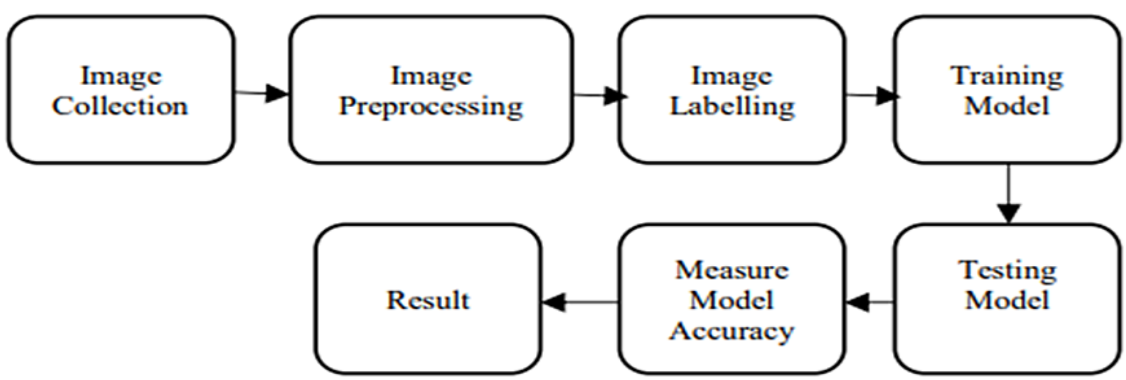
FR No.	Non-Functional Requirement	Description
NFR -1	Usability	Datasets of all the leaves are used to detectthe disease thatis present in the leaf.
NFR-2	Security	The information belongsto the user and the leaves are secured highly.
NFR-3	Reliability	The leavesquality is very important for predicting the disease in leaves.
NFR-4	Performance	The performance is based on the quality of the leaf used for diseaseprediction.
NFR-5	Availability	It is available for all the users to predict the diseasein the plants.
NFR-6	Scalability	Increasing the prediction of the disease in the leaves.

5.PROJECT DESIGN

5.1 Data Flow Diagram :



5.2 Solution & Technical Architecture:



5.3User Stories :

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					

6. PROJECT PLANNING & SCHEDULING

6.1.SPRINT PLANNING & ESTIMATION:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Download data set	USN-1	The data is downloaded from the Kaggle website and then the data set is classified into training and testing images.	10	High	Priyadharshini S Vinith R
Sprint-1	Collect the Data, assess the data set.	USN-1	It is necessary for an animal rights activist to gather information about forest fires.	10	High	Keerthika P Thenmozhi R
Sprint-2	Image preprocessing	USN-2	In Image processing technique the first step is usually importing the libraries that will be needed In the program. Import Keras library from that library and import the ImageDataGenerator Library to your Python script.	20	High	Priyadharshini S Vinith R Keerthika P Thenmozhi R

			The next step is defining the arguments for the ImageDataGenerator And next step is applying the ImageDataGenerator arguments to the train and test dataset.			
Sprint-3	Training image	USN-3	In this training phase the ImageDataGenerator arguments is applied to the training images and the model is tested with several images and the model is saved.	20	High	Priyadharshini S Vinith R Keerthika P Thenmozhi R
Sprint-4	Testing Image, Evaluation metrics and accuracy	USN-4	In this testing phase the Image processing techniques is applied to the testing images and executed for prediction. In this phase the result, prediction, accuracy, and performance of the	20	High	Priyadharshini S Vinith R Keerthika P Thenmozhi R

MILESTONE & ACTIVITY LIST:

Activity Number	Activity Name	Detailed Activity Description	Assigned To	Status / Comments
1.1	Access Resources	Access the resources (courses) in project dashboard.	All Members	COMPLETED
1.2	Rocket chat registration	Join the mentoring channel via platform & rocket-chat mobile app.	All Members	COMPLETED
1.3	Access workspace	Access the guided project workspace.	All Members	COMPLETED
1.4	IBM Cloud registration	Register on IBM Academic Initiative & Apply Feature code for IBM Cloud Credits.	All Members	COMPLETED
1.5	Project Repository Creation	Create GitHub account & collaborate with Project Repository in project workspace.	All Members	COMPLETED
1.6	Environment Setup	Set-up the Laptop / Computers based on the pre-requisites for each technology track.	All Members	COMPLETED
2.1	Literature survey	Literature survey on the selected project & Information Gathering.	All Members	COMPLETED
2.2	Technology Training	Attend the technology trainings as per the training Calendar.	All Members	COMPLETED
2.3	Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	All Members	COMPLETED
2.4	Technology Training	Attend the technology trainings as per the training Calendar.	All Members	COMPLETED
2.5	Brainstorming	List the ideas (at least 4 per each team member) by	All Members	COMPLETED

		organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.		
2.6	Technology Training	Attend the technology trainings as per the training Calendar.	All Members	COMPLETED
3.1	Proposed Solution Document	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	All Members	COMPLETED
3.2	Technology Training	Attend the technology trainings as per the training Calendar.	All Members	COMPLETED
3.3	Problem - Solution fit & Solution Architecture	Prepare problem - solution fit document & Solution Architecture.	All Members	COMPLETED
3.4	Technology Training	Attend the technology trainings as per the training Calendar.	All Members	COMPLETED
4.1	Customer Journey Map	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	All Members	COMPLETED
4.2	Technology Training	Attend the technology trainings as per the training Calendar.	All Members	COMPLETED
4.3	Functional Requirements & Data Flow Diagrams	Prepare the Functional Requirement Document & Data-Flow Diagrams.	All Members	COMPLETED
4.4	Technology Architecture	Prepare Technology Architecture of the solution.	All Members	COMPLETED
4.5	Technology Training	Attend the technology trainings as per the training Calendar.	All Members	COMPLETED
5.1	Milestone & Activity List	Prepare Milestone & Activity List.	All Members	COMPLETED
5.2	Sprint Delivery Plan	Prepare Sprint Delivery Plan.	All Members	IN PROGRESS

6	Data Collection	Collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc.	All Members	COMPLETED
7.1	Image Preprocessing	Importing the ImageDataGenerator Library	All Members	COMPLETED
7.2	Image Preprocessing	Define the parameters/arguments for ImageDataGenerator class.	All Members	COMPLETED
7.3	Image Preprocessing	Applying ImageDataGenerator functionality to trainset and test set.	All Members	COMPLETED
8.1	Model Building	Importing the model building libraries.	Priyadharshini S Thenmozhi R	IN PROGRESS
8.2	Model Building	Initializing the model.	Keerthika P Vinith R	IN PROGRESS
8.3	Model Building	Adding CNN Layers.	Priyadharshini S Thenmozhi R	IN PROGRESS
8.4	Model Building	Adding Dense Layers	Keerthika P Vinith R	IN PROGRESS
8.5	Model Building	Configuring the learning process	Priyadharshini S Thenmozhi R	IN PROGRESS
8.6	Model Building	Training the Model	Keerthika P Vinith R	IN PROGRESS
8.7	Model Building	Save the model	Priyadharshini S Thenmozhi R	IN PROGRESS
8.8	Model Building	Predictions	Keerthika P	IN PROGRESS

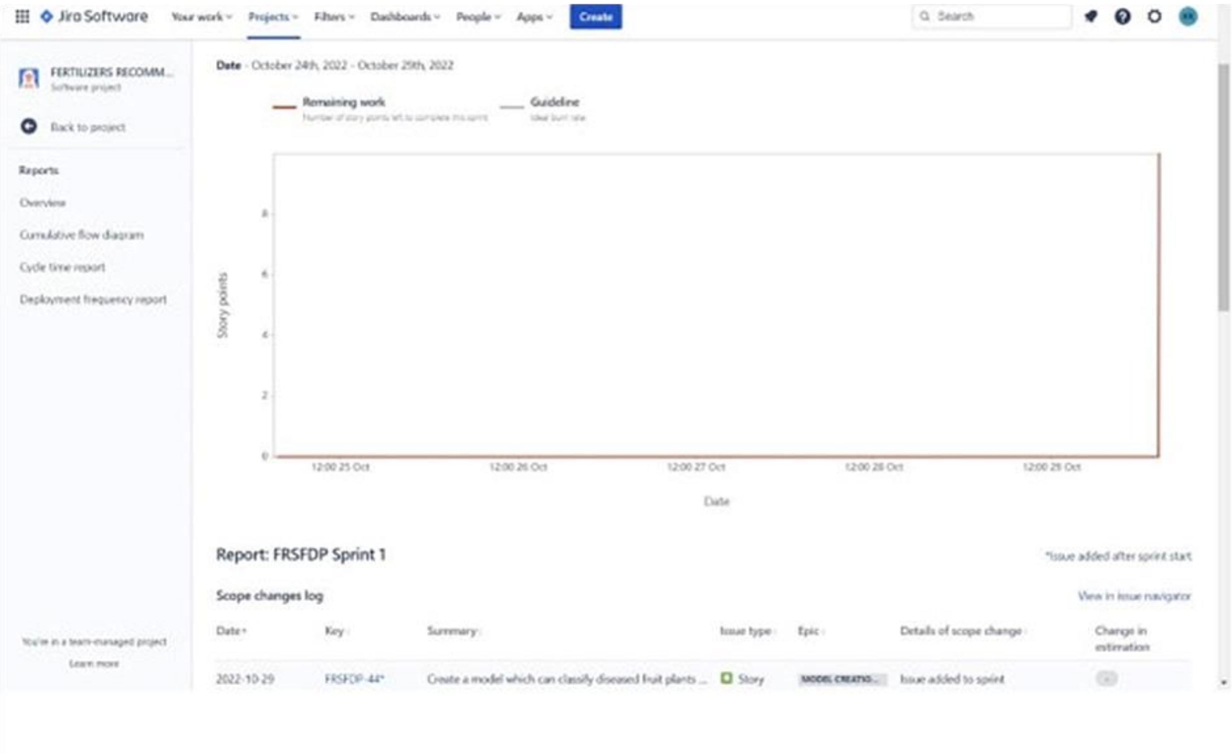
6.2.SPRINT DELIVERY SCHEDULE:

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	10	6 Days	24 Oct 2022	29 Oct 2022	10	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	18 Nov 2022

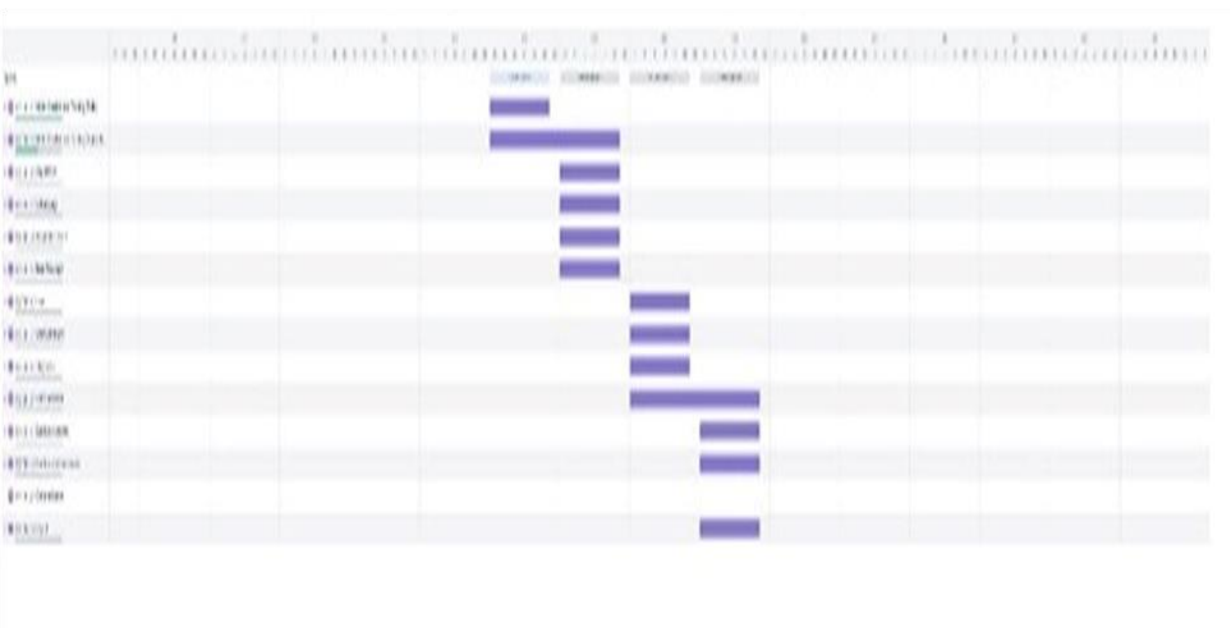
6.3 Reports from JIRA:

ACTIVITY LIST

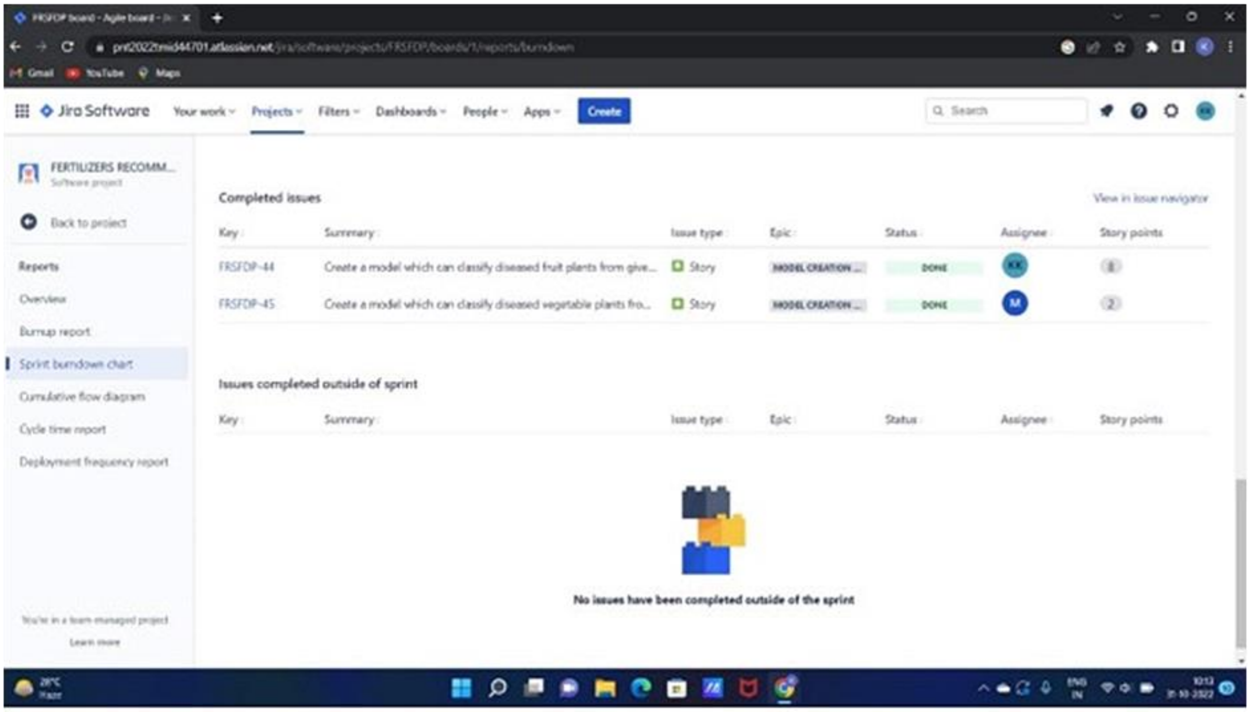
BURNDOWNCHART


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
FERTILIZERS RECOMM...
Software project
Back to project
Reports
Overview
Cumulative flow diagram
Cycle time report
Deployment frequency report
Date - October 24th, 2022 - October 29th, 2022
Remaining work
Guideline
Story points
12:00 25 Oct
12:00 26 Oct
12:00 27 Oct
12:00 28 Oct
12:00 29 Oct
Date
Report: FRSFDP Sprint 1
*Issue added after sprint start
Scope changes log
View in issue navigator
Date
Key
Summary
Issue type
Epic
Details of scope change
Change in estimation
2022-10-29
FRSFDP-44*
Create a model which can classify diseased fruit plants ...
Story
Issue added to sprint

ROAD MAP



SPRINT BURNDOWNCHART



7.CODING & SOLUTIONING

7.1.FEATURE 1:

1.IMAGE DATA GENERATOR

Keras ImageDataGenerator is used for getting the input of the original data and further, it makes the transformation of this data on a random basis and gives the output resultant containing only the data that is newly transformed. It does not add the data.

```
from keras.preprocessing.image import ImageDataGenerator
```

2.PARAMETRES

2.1.Rescale:

The ImageDataGenerator class can be used to rescale pixel values from the range of 0-255 to the range 0-1 preferred for neural network models. Scaling data to the range of 0-1 is traditionally referred to as normalization.

2.2.Shear Range:

Shear range means that the image will be distorted along an axis, mostly to create or rectify the perception angles. It's usually used to augment images so that computers can see how humans see things from different angles.

2.3.Rotation range:

ImageDataGenerator class allows you to randomly rotate images through any degree between 0 and 360 by providing an integer value in the rotation_range argument. When the image is rotated, some pixels will move outside the image and leave an empty area that needs to be filled in.

2.4.Zoom Range:

The zoom augmentation method is used to zooming the image. This method randomly zooms the image either by zooming in or it adds some pixels aroundthe image to enlarge the image. This method uses the zoom_range argument of the ImageDataGenerator class. We can specify the percentage value of the zooms either in a float**,** range in the form of an array.

2.5.Horizontal Flip:

Horizontal flip basically flips both rows and columns horizontally. So for this, we have to pass the horizontal_flip=True argument in the ImageDataGenerator constructor.

3.CONVOLUTION NEURAL NETWORK:

A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data. There are other types of neural networks in deep learning, but for identifying and recognizing objects, CNNs are the network architecture of choice. The layers used in the CNN algorithm is Convolutional ,maxpooling, and flatten layer.

3.1.Convolutional Layer:

A convolutional layer is the main building block of a CNN. It contains a set of filters (or kernels), parameters of which are to be learned throughout the training. The size of the filters is usually smaller than the actual image. Each filter convolves with the image

Convolution layer is used for a image processing to blur and sharpen images, but also to perform other operations.

```
from keras.layers import Convolution2D
```

3.2.Maxpooling Layer:

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter.

```
from keras.layers import MaxPooling2D
```

3.3.Flatten Layer:

Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector. The flattened matrix is fed as input to the fully connected layer to classify the image.

```
from keras.layers import Flatten
```

4.DENSE LAYER:

Dense Layer is used to classify image based on output from convolutional layers.

7.2.FEATURE 2(CODE):

Importing Keras libraries

```
import keras
```

Importing ImageDataGenerator from Keras

```
from matplotlib import pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
```

Defining the Parameters

```
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,
rotation_range=180,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,rotation_range
=180,zoom_range=0.2,horizontal_flip=True)
```

Applying ImageDataGenerator functionality to train dataset

```
from google.colab import drive
drive.mount('/content/drive')x_train=train_datagen.flow_from_directory('/content/d
rive/MyDrive/LEAF DISEASE/dataset/DATA
/train_set',target_size=(64,64),batch_size=32, class_mode='binary')
```

Applying ImageDataGenerator functionality to test dataset

```
x_test=test_datagen.flow_from_directory('/content/drive/MyDrive/LEAF DISEASE/dataset/test_set',target_size=(64,64),batch_size=32,
class_mode='binary')
```

Importing Model Building Libraries

```
#to define the linear Initialisation import sequential
from keras.models import Sequential
#to add layers import Dense
from keras.layers import Dense
#to create Convolutional kernel import convolution2D
from keras.layers import Convolution2D
#import Maxpooling layer
from keras.layers import MaxPooling2D
#import flatten layer
from keras.layers import Flatten
import warnings
warnings.filterwarnings('ignore')
```

Initializing the model

```
model = Sequential
```

Adding CNN Layers

```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),activation='relu'))
#add maxpooling layers
model.add(MaxPooling2D(pool_size=(2,2)))
#add faltten layer
model.add(Flatten())
```

Add Dense layers

```
#add hidden layers
model.add(Dense(150,activation='relu'))
#add output layer
model.add(Dense(1,activation='sigmoid'))
```

Configuring the learning process

```
model.compile(loss='binary_crossentropy',optimizer="adam",metrics=
["accuracy"])
```

Training the model

```
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=
x_test,validation_steps=4)
```

Save the model

```
model.save("MODEL.h5")
```

8.TESTING

8.1.Test Cases:

8.2.User Acceptance Testing:

1.Purposeof Document :

The purpose of this document is to briefly explain the test coverage and open issues of the [Fertilizer Recommendation system for plant disease prediction] project at the time of the release to UserAcceptance Testing (UAT).

2.Defect Analysis :

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

Resolution	Severity 1	Severity y2	Severity 3	Severity y4	Sub total
Common rusk	10	4	2	3	19
Bacterial leaf streak	5	6	3	6	23
Gray leaf spot	2	7	0	1	10
Brown spot	11	4	3	20	36
Anthranose leaf blight	3	2	1	0	6
Northern corn leaf blight	9	3	1	1	10
Eyespot	11	5	2	1	12
Totals	44	31	13	32	116

3.TestCase Analysis:

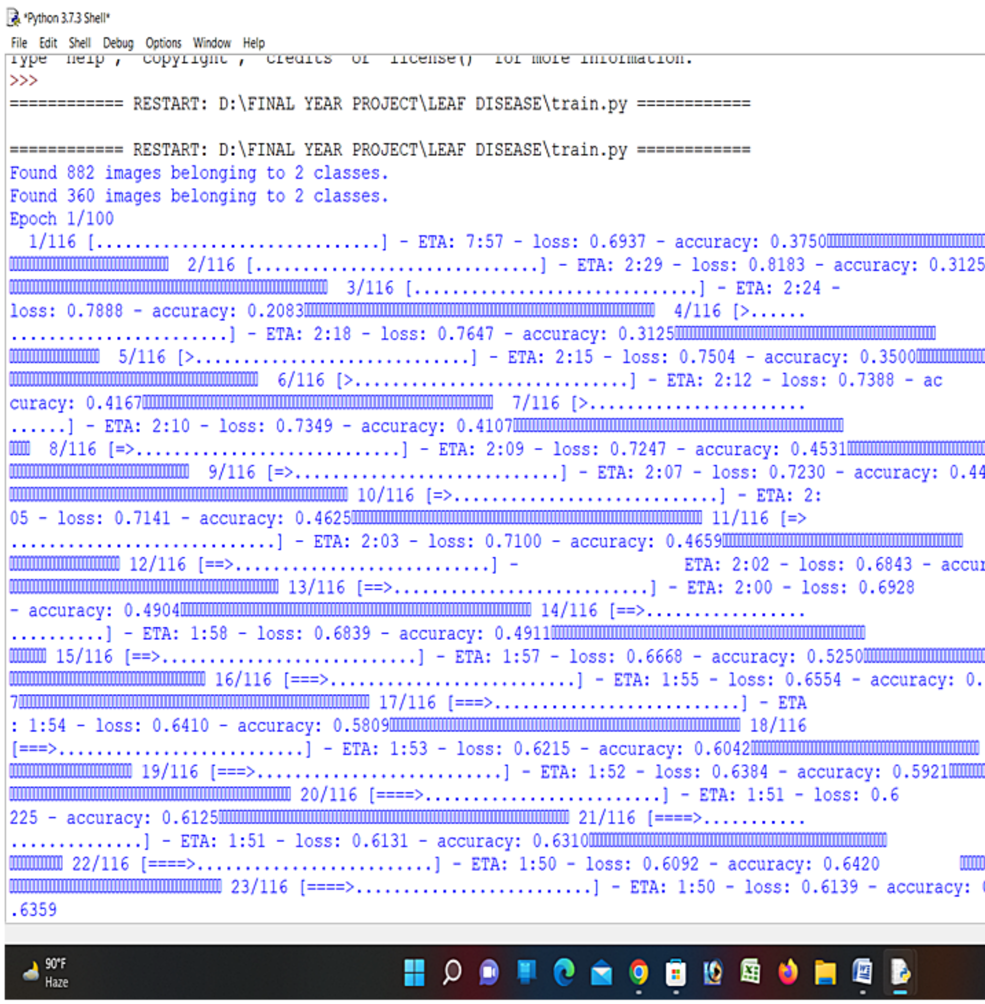
This report showsthe number of test cases that have passed, failed,and untested.

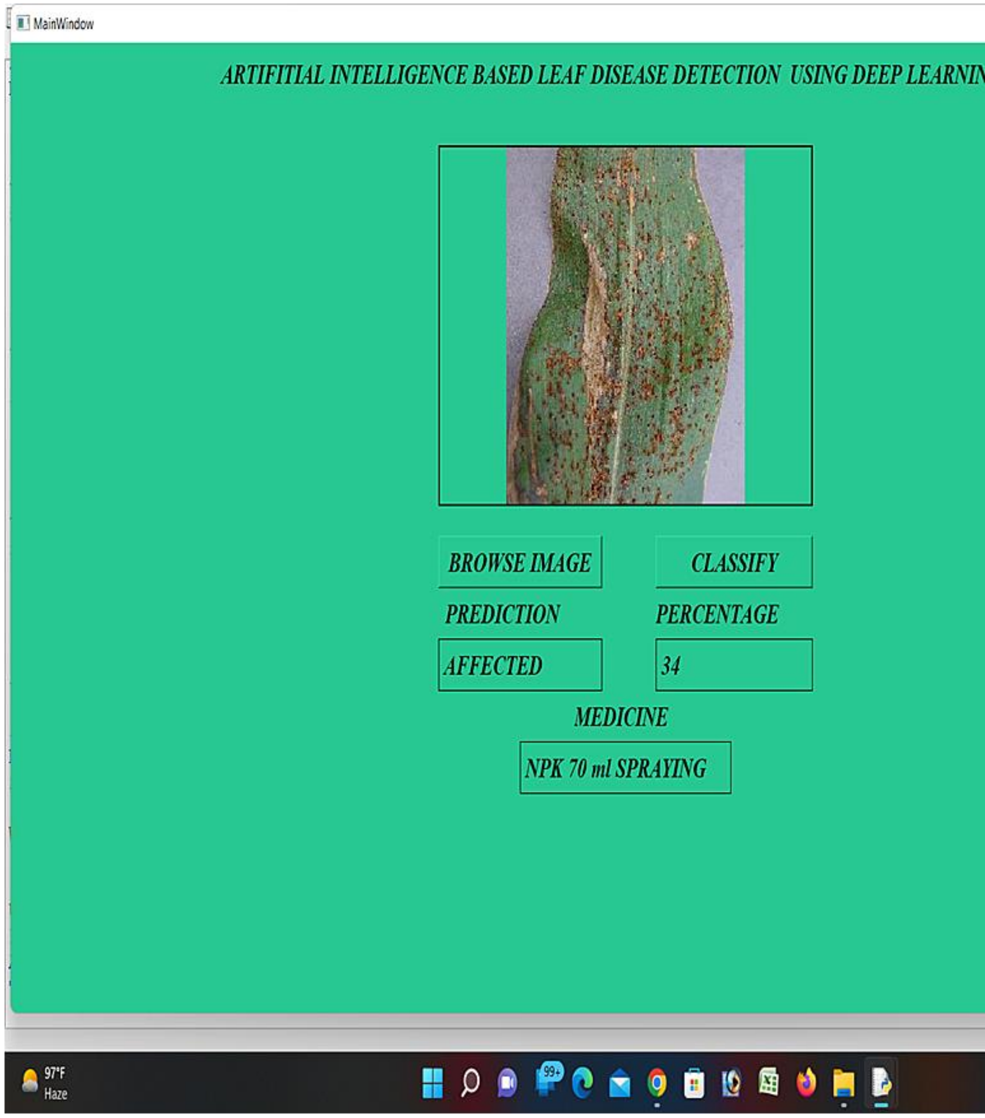
Section	Total Cases	Not Tested	Fail	Pass
Common rusk	17	0	0	17
Bacterial leaf streak	20	0	0	20
Gray leaf spot	7	0	0	7
Brown spot	9	0	0	9
Anthranose leaf blight	16	0	0	16
Northern corn leaf blight	51	0	0	51
Eyespot	2	0	0	2

9.1 Model Performance Testing:

s.no.	parameter	Values	screenshot
1.	Model summary	Total params: 1,572,768 Trainable params: 1,572,768 Non-trainable params:0	model.summary() Model: "sequential" ____ Layer (type) Output Shape Param # == = conv2d (Conv2D) (None, 510, 510, 16) 448 max_pooling2d (MaxPooling2D (None, 255, 255, 16) 0) conv2d_1 (Conv2D) (None, 253, 253, 32) 4640 max_pooling2d_1 (MaxPooling (None, 126, 126, 32) 0 2D) conv2d_2 (Conv2D) (None, 124, 124, 64) 18496 max_pooling2d_2 (MaxPooling (None, 62, 62, 64) 0 2D) conv2d_3 (Conv2D) (None, 60, 60, 128) 73856 max_pooling2d_3 (MaxPooling (None, 30, 30, 128) 0 2D) conv2d_4 (Conv2D) (None, 28, 28, 256) 295168 max_pooling2d_4 (MaxPooling (None, 14, 14, 256) 0

			2D) conv2d_5 (Conv2D) (None, 12, 12, 512) 1180160 max_pooling2d_5 (MaxPooling (None, 6, 6, 512) 0 2D) flatten (Flatten) (None, 18432) 0 === = Total params: 1,572,768 Trainable params: 1,572,768 Non-trainable params: 0 ____

2.	Accuracy	Training Accuracy:64.20 Validation Accuracy:80	Epoch 1/100 111/116 [===========================>..] - ETA: 11s - loss: 0.7071 WARNING:tensorflow:Your input ran out of data; interrupting training. Mal or generator can generate at least `steps_per_epoch * epochs` batches (in this You may need to use the repeat() function when building your dataset. 116/116 [==============================] - 369s 3s/step - loss: 0.7 val_loss: 2.0949 - val_accuracy: 0.8000 <keras.callbacks.History at 0x7ff1d8e88590> 

3.	Confidence Score (Only Yolo Projects)	Class Detected: 80% Confidence Score : 95%	

10.ADVANTAGES & DISADVANTAGES

ADVANTAGES:

1.An automatic plant-disease detection system provides clear benefit in **monitoring of large fields**, as this is the only approach that provides a chance to discover diseases at an early stage.

2.Leaves of a plant can be used to determine the health status of that plant. The proposed of this work is **to develop a system that capable to detect and identify the type of disease.**

3.The results is quite accurate with the accuracy upto 95%

DISADVANTAGES:

1.Individual learner is responsible for learning global information to avoid false positives.

2.The limited learning and perception ability of individual learners is not sufficient to make them perform well in complex tasks.

3.Proper connectivity and maintenance will be a complex task.

11.CONCLUSION

A Convolution Neural network Deep learning based approach is proposed for predicting leaf disease. The developed approach was evaluated with actual datasets collected from the images while capturing the crops. The evaluation process is conducted with manually labeled data and the proposed active deep learning shows a favorable performance. The accuracy of leaf disease prediction is to be above 95% using neural network algorithm. From this we can get better performance analysis.

12.FUTURE SCOPE

1. The challenge is the durability of the disease resistances, and their agronomic management. This challenge needs to be dealt with seriously, in order to convince a public often hostile to this technology. Durability is not a specific aspect of resistance genes obtained by genome editing, and the answers are the same as for introgressed resistance genes discovered in the genetic variability of the species:
2. (i) the stacking of several resistance genes, preferably with different modes of action,
3. (ii) a focus on systems other than NBS-LRR receptor kinases known to break down rapidly, and
4. (iii) good agronomic practices, including, in particular, crop rotation and the concomitant use of biocontrol agents.

13. APPENDIX

13.1 SOURCE CODE

13.1.1 Train Code

```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dropout
model = Sequential()
model.add(Conv2D(16, (3, 3), input_shape = (512,512, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Conv2D(32, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Conv2D(64, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Conv2D(128, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Conv2D(256, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Conv2D(512, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Flatten())
model.add(Dense(units = 128, activation = 'relu'))
model.add(Dense(units = 1, activation = 'sigmoid'))
```

```
 model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =
['accuracy'])
 train_datagen = ImageDataGenerator(rescale = 1./255,
                                    shear_range = 0.2,
                                    zoom_range = 0.2,
                                    horizontal_flip = True)
val_datagen = ImageDataGenerator(rescale = 1./255)
 training_set = train_datagen.flow_from_directory('data/train',
                                                  target_size = (512,512),
                                                  batch_size = 8,
                                                  class_mode = 'binary')
val_set = val_datagen.flow_from_directory('data/val',
                                          target_size = (512,512),
                                          batch_size = 8,
                                          class_mode = 'binary')
model.fit(training_set,
                         steps_per_epoch = 116,
                         epochs = 100,
                         validation_data = val_set,
                         validation_steps = 45)

model_json = model.to_json()
with open("model.json", "w") as json_file:
 json_file.write(model_json)
model.save_weights("model.h5")
print("Saved model to disk")
```

13.1.2 Test Code

```
from keras.models import model_from_json
import numpy as np
from keras.preprocessing import image
import pandas as pd
import cv2
from time import sleep
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)
model.load_weights("model.h5")
print("Loaded model from disk")
global img
def KNN():
    global img
    dataset = pd.read_csv("leaf_disease.csv")
    print(dataset)

    x = dataset.iloc[:,:-1] #independent
    y = dataset.iloc[:,-1] #dependent
    from sklearn.model_selection import train_test_split
    X_train, X_test, Y_train, Y_test = train_test_split(x,y, test_size=0.25,
   random_state=0)
    print(X_train)
    print(Y_train)
    print(X_test)
```

```
print(Y_test)
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=3)
classifier.fit(X_train, Y_train)
Y_predict = classifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(Y_test, Y_predict))
print(classification_report(Y_test, Y_predict))
from sklearn import metrics
#Model Acc555uracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(Y_test, Y_predict))
img = cv2.resize(img,(400,400))
cv2.imshow("Original Frame",img)
## convert to hsv
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
#cv2.imshow("hsv",hsv)
## mask of red (36,0,0) ~ (70, 255,255)
mask1 = cv2.inRange(hsv, (0,0,100), (0,0,255)) #red
#cv2.imshow("mask1",mask1)
red= cv2.countNonZero(mask1)
print("red = ",red)
img = cv2.GaussianBlur(img,(5,5),2)
im_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(im_gray,127,255,0)
count = cv2.countNonZero(thresh)
#print(count)
RED=((red+count)/2)*0.001000
```

```
    contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    for contour in contours:
      cv2.drawContours(im_gray, contours, -1, (0,255,0), 6)
      cv2.imshow("contour",im_gray)
    output = classifier.predict([[red]])
    print("Predicted New Output = ",output)
    if output == 1:
        print("Affected")
        print("Total Percentage of Affected = ",int(RED))
    if output == 0:
        print("Normal")
      def classify(img_file):
    global img
    img_name = img_file
    print(img_name)
    test_image = image.load_img(img_name, target_size = (512,512))
    test_image = image.img_to_array(test_image)
    test_image = np.expand_dims(test_image, axis=0)
    result = model.predict(test_image)
    print(result[0][0])
    if result[0][0] == 0:
        prediction = 'Corn Affected'
        img = cv2.imread(img_name)
        KNN()
    else:
        prediction = 'Corn Normal'
```

```
    print(prediction,img_name)
    import os
path = 'data/test'
files = []
print(path)
# r=root, d=directories, f = files
for r, d, f in os.walk(path):
   for file in f:
     if '.jpg' in file:
       files.append(os.path.join(r, file))
       for f in files:
   classify(f)
   print('\n')
```

13.1.3 GUI Coding

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'eye.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.
from PyQt5 import QtCore, QtGui, QtWidgets
from keras.models import model_from_json
import numpy as np
from keras.preprocessing import image
import pandas as pd
```

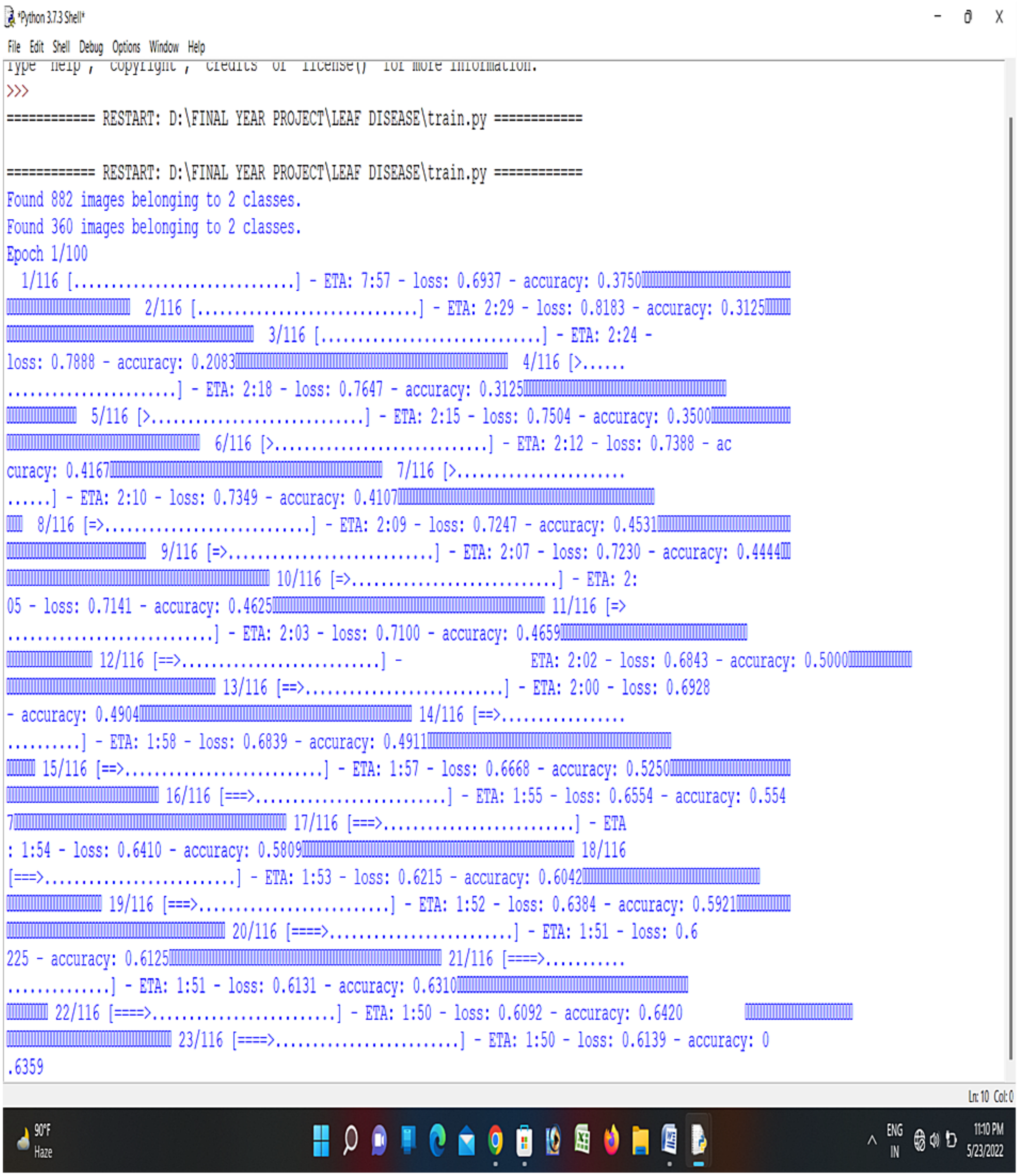
```
import cv2
from time import sleep
class Ui_CLASSIFY(object):
    def setupUi(self, CLASSIFY):
        CLASSIFY.setObjectName("CLASSIFY")
        CLASSIFY.resize(1969, 944)
        CLASSIFY.setStyleSheet("background-color: rgb(40, 200, 147);")
        self.centralwidget = QtWidgets.QWidget(CLASSIFY)
        self.centralwidget.setObjectName("centralwidget")
        self.TITTLE = QtWidgets.QLabel(self.centralwidget)
        self.TITTLE.setGeometry(QtCore.QRect(190, 0, 1391, 61))
        font = QtGui.QFont()
        font.setFamily("Times New Roman")
        font.setPointSize(16)
        font.setBold(True)
        font.setItalic(True)
        font.setUnderline(False)
        font.setWeight(75)
        font.setStrikeOut(False)
        font.setKerning(False)
        font.setStyleStrategy(QtGui.QFont.PreferDefault)
        self.TITTLE.setFont(font)
        self.TITTLE.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.TITTLE.setMouseTracking(False)
        self.TITTLE.setAlignment(QtCore.Qt.AlignCenter)
        self.TITTLE.setWordWrap(False)
        self.TITTLE.setObjectName("TITTLE")
```

```
self.IMAGESHOW = QtWidgets.QLabel(self.centralwidget)
self.IMAGESHOW.setGeometry(QtCore.QRect(630, 100, 551, 351))
font = QtGui.QFont()
font.setFamily("Times New Roman")
font.setPointSize(14)
font.setBold(True)
font.setItalic(True)
font.setWeight(75)
self.IMAGESHOW.setFont(font)
self.IMAGESHOW.setFrameShape(QtWidgets.QFrame.Box)
self.IMAGESHOW.setFrameShadow(QtWidgets.QFrame.Plain)
self.IMAGESHOW.setLineWidth(2)
self.IMAGESHOW.setMidLineWidth(0)
self.IMAGESHOW.setText("")
self.IMAGESHOW.setAlignment(QtCore.Qt.AlignCenter)
self.IMAGESHOW.setObjectName("IMAGESHOW")
self.BROWSEIMAGE = QtWidgets.QPushButton(self.centralwidget)
self.BROWSEIMAGE.setGeometry(QtCore.QRect(630, 480, 241, 51))
font = QtGui.QFont()
font.setFamily("Times New Roman")
font.setPointSize(16)
font.setBold(True)
font.setItalic(True)
font.setWeight(75)
self.BROWSEIMAGE.setFont(font)
self.BROWSEIMAGE.setObjectName("BROWSEIMAGE")
self.BROWSEIMAGE_2 = QtWidgets.QPushButton(self.centralwidget)
```

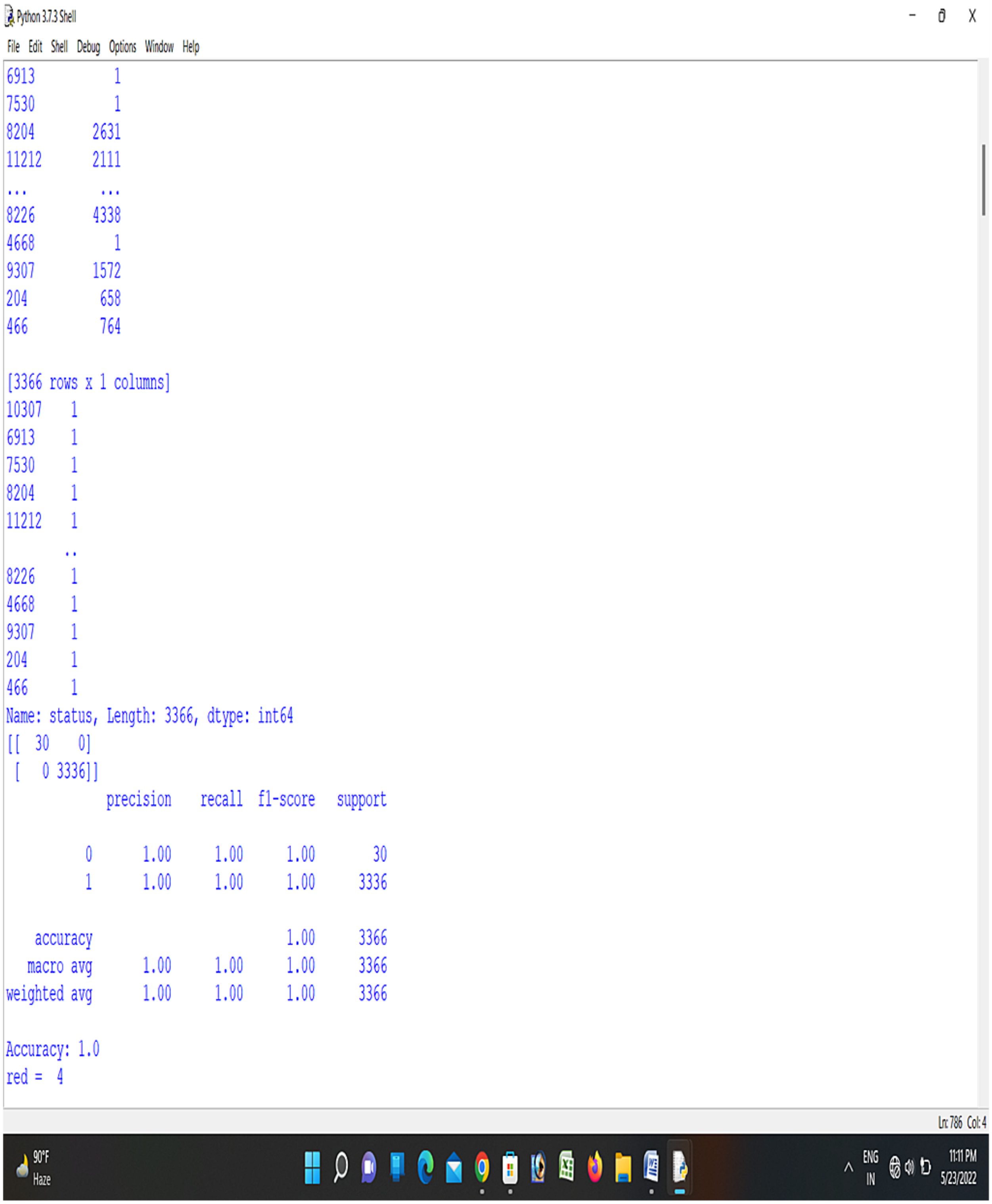
```
self.BROWSEIMAGE_2.setGeometry(QtCore.QRect(950, 480, 231, 51))
font = QtGui.QFont()
font.setFamily("Times New Roman")
font.setPointSize(16)
font.setBold(True)
font.setItalic(True)
font.setWeight(75)
self.BROWSEIMAGE_2.setFont(font)
self.BROWSEIMAGE_2.setObjectName("BROWSEIMAGE_2")
self.PREDICTION = QtWidgets.QLabel(self.centralwidget
```

13.2 SCREEN SHOTS

13.2.1 Train Data

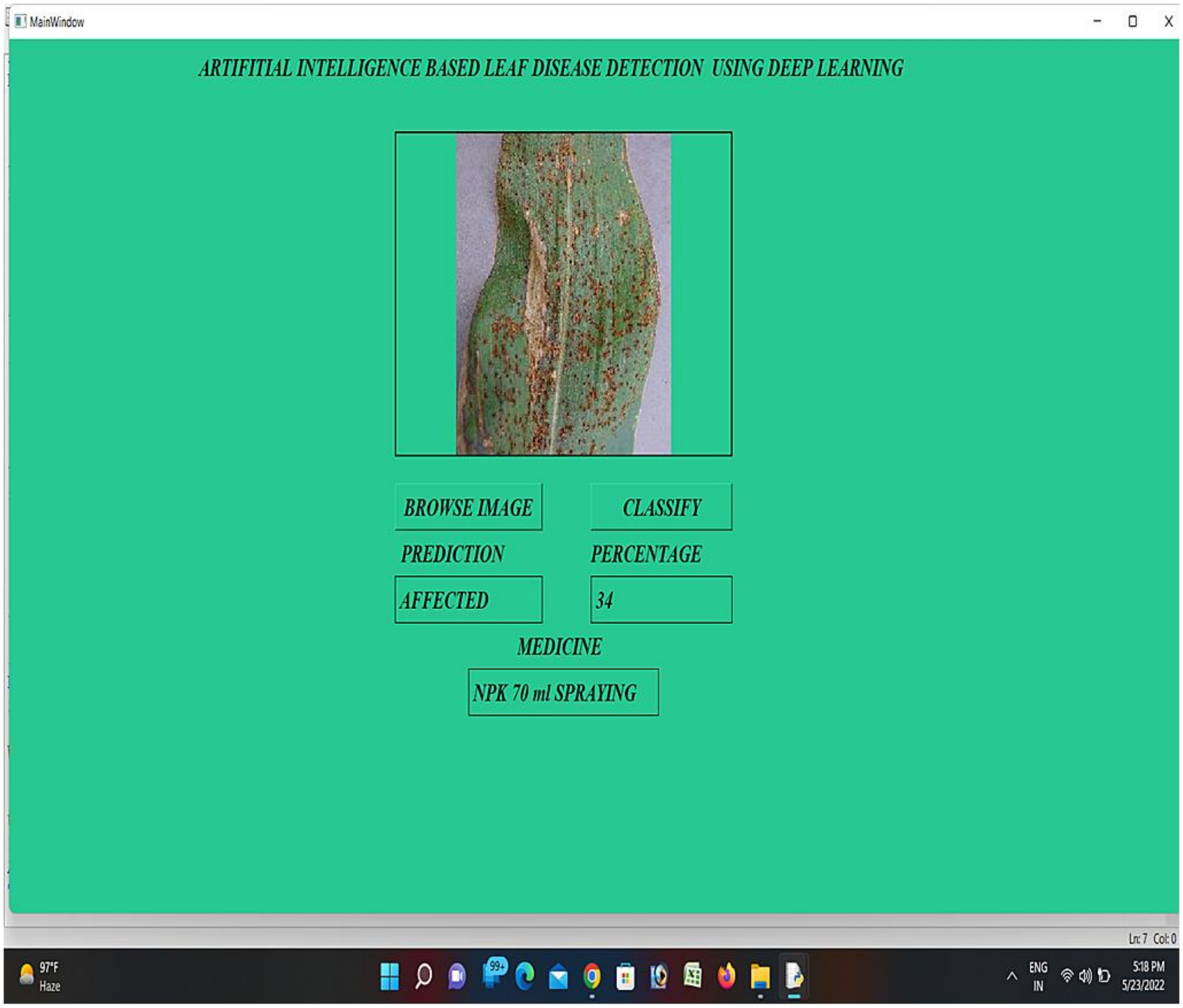


13.2.2 Test Data



```
6913         1
7530         1
8204      2631
11212     2111
...        ...
8226      4338
4668         1
9307      1572
204        658
466        764

[3366 rows x 1 columns]
10307    1
6913     1
7530     1
8204     1
11212    1
        ..
8226     1
4668     1
9307     1
204      1
466      1
Name: status, Length: 3366, dtype: int64
[[  30    0]
 [   0 3336]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        30
           1       1.00      1.00      1.00      3336

    accuracy                           1.00      3366
   macro avg       1.00      1.00      1.00      3366
weighted avg       1.00      1.00      1.00      3366

Accuracy: 1.0
red =  4
```

13.2.3 GUI Final Output



Our GitHub Link: https://github.com/IBM-EPBL/IBM-Project-51982-1660987690

Demo Link: https://youtu.be/9qDGwxQPgZM