PROJECT REPORT

FERTILIZERS RECOMMENDATION SYSTEM FOR DISEASE PREDICTION

SUBMITTED BY,

Team ID : PNT2022TMID16174

GOKUL - 927619BIT4025

BALAVIGNESHWAR -972619BIT4006

MURALIDHARAN -972619BIT4060

HIRUTHIK -927619BIT4036

1. INTRODUCTION

1.1 Project Overview

Detection and recognition of plant diseases using machine learning are very efficient in providing symptoms of identifying diseases at its earliest. Plant pathologists can analyses the digital images using digital image processing for diagnosis of plant diseases. Application of computer vision and image processing strategies simply assist farmers in all the regions of agriculture. Generally, the plant diseases are caused by the abnormal physiological functionalities of plants. Therefore, the characteristic symptoms are generated based on the differentiation between normal physiological functionalities and abnormal physiological functionalities of the plants. Mostly, the plant leaf diseases are caused by Pathogens which are positioned on the stems of the plants. These different symptoms and diseases of leaves are predicted by different methods in image processing. These different methods include different fundamental processes like segmentation, feature extraction and classification and so on. Mostly, the prediction and diagnosis of leaf diseases are depending on the segmentation such as segmenting the healthy tissues from diseased tissues of leaves.

1.2 Purpose

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

2. LITERATURE SURVEY

2.1 Existing problem

Existing problem Indumathi proposed a method for leaf disease detection and suggest fertilizers to cure leaf diseases. But the method involves a smaller number of train and test sets which results in poor accuracy. Pandi selvi proposed a simple prediction method for soil-based fertilizer recommendation system for predicted crop diseases. This method gives less accuracy and prediction. Shiva Reddy proposed an IoT based system for leaf disease detection and fertilizer recommendation which is based on Machine Learning techniques yields less 80 percentage accuracy.

2.2 References

[1] The proposed method uses SVM to classify tree leaves, identify the disease and suggest the fertilizer. The proposed method is compared with the existing CNN based leaf disease prediction. The proposed SVM technique gives a better result when compared to existing CNN. For the same set of images, F-Measure for CNN is 0.7and 0.8 for SVM, the accuracy of identification of leaf disease of CNN is 0.6 and SVM is 0.8.

[2] Detection of Leaf Diseases and Classification using Digital Image Processing International Conference on Innovations in Information, Embedded and Communication Systems(ICIIECS), IEEE, 2017.

[3] Cloud Based Automated Irrigation and Plant Leaf Disease Detection System Using An Android Application. International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017.

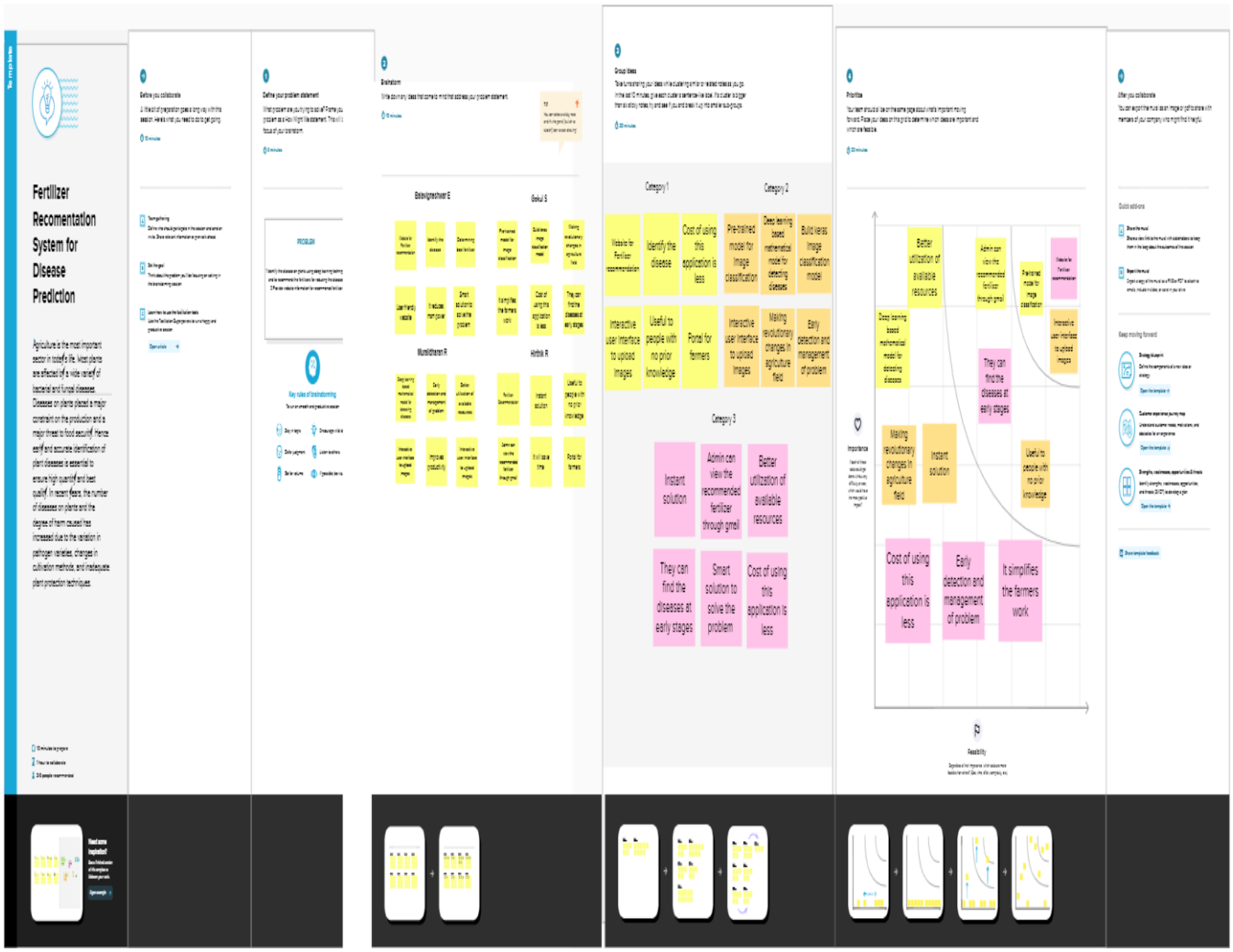
2.3 Problem Statement Definition

Mr.Narasimma Rao is a 65 years old man. He had a own farming land and do Agriculture for past 30 Years , In this 30 Years he Faced a problem in Choosing Fertilizers and Controlling of Plant Disease. • Narasimma Rao wants to know the better recommendation for fertilizers for plants with the disease. • He has faced huge losses for a long time. • This problem is usually faced by most farmers. • Mr. Narasimma Rao needs to know the result immediately.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming



3.3 Proposed Solution

Even when considering trees only, leaves show an impressively wide variety in shapes. It is however necessary to come up with a representation of what a leaf is, that is accurate enough to be fitted to basically any kind of leaf. The general shape of a leaf is a key component of the process of identifying a leaf. Botanists have a whole set of terms describing either the shape of a simple leaf, of the lobes of a palmate leaf, or of the leaflets of a compound leaf. Here present a study on segmentation of leaf images restricted to semi-controlled conditions, in which leaves are photographed against a solid light-colored background. Such images can be used in practice for plant species identification, by analyzing the distinctive shapes of the leaves. The most important of these are: the variety of leaf shapes, inevitable presence of shadows and specularities, and the time constraints required by interactive species identification applications. The identification of species is the first and essential key to understand the plant environment. In this project introduce a method designed to deal with the obstacles raised by such complex images, for simple and lobed tree leaves. A first segmentation step based on a light polygonal leaf model is first performed, and later used to guide the evolution of an active contour. Combining global shape descriptors given by the polygonal model with local curvature-based features, the leaves are then classified over leaf datasets. In this project we introduce a method designed to deal with the obstacles raised by such complex images, for simple and lobed tree leaves. A first segmentation step based on graph cut approach is first performed, and later used to guide the evolution of leaf boundaries. And implement classification algorithm to classify the diseases and recommend the fertilizers to affected leaves

3.4 Problem Solution Fit

Even though the botanical categorization was not founded on their characteristics, leaves are the most evident and popular choice for identifying different tree species. They are almost always to be found, simple to photograph, and have well-studied specificities in their morphologies that make identification conceivable, if not simple. The Folia application aim is to create a system for leaf shape analysis that, in contrast to previous work, analyses images taken in a natural setting. First, a light polygonal leaf model-based segmentation phase is carried out, and the results are then used to direct the development of an active contour. The leaves are then categorised across leaf datasets by combining global shape descriptors provided by the polygonal model with local curvature-based characteristics. In this research,we present a method for basic and lobed tree leaves that is intended to overcome the challenges presented by such complex images. In order to direct the evolution of leaf borders, a first segmentation step based on a graph cut technique is first carried out. implement a classification algorithm to categorise the diseases, and then provide fertilisers for leaves with impacted diseases.

4. REQUIREMENT ANALYSIS

4.1 Functional Requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	User Confirmation	Confirmation via Email
FR-3	User Profile	Filling the profile page after logging in.
FR-4	Uploading Dataset (Leaf)	Images of the leaves are to be uploaded.
FR-5	Requesting solution	Uploaded images is compared with the pre-defined Model and solution is generated.
FR-6	Downloading Solution	The Solution in pdf format which contains the recommendations of fertilizers and the possible diseases.

4.2 Non – Functional Requirement

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

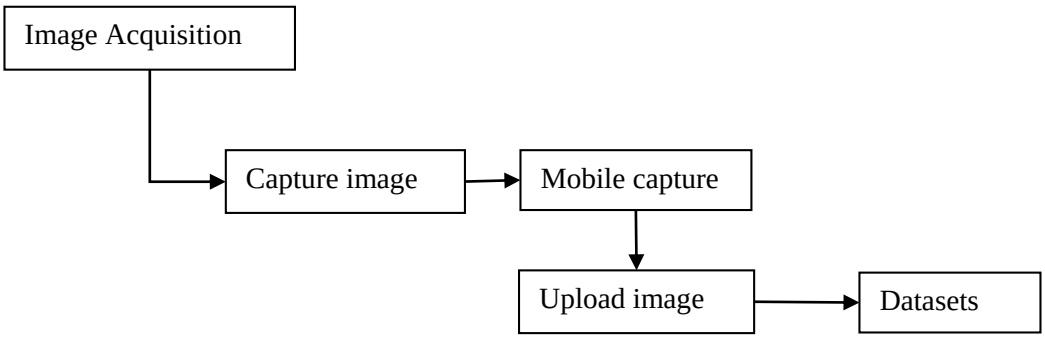
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The system allows the user to perform the tasks easily and efficiently and effectively.
NFR-2	**Security**	Assuring all data inside the system or its part will be protected against malware attacks or unauthorized access.
NFR-3	**Reliability**	The website does not recover from failure quickly ,it takes time as the application is running in single server
NFR-4	**Performance**	Response Time and Net Processing Time is Fast
NFR-5	**Availability**	The system will be available up to 95% of the time
NRF-6	**Scalability**	The website is scalable.

5. PROJECT DESIGN

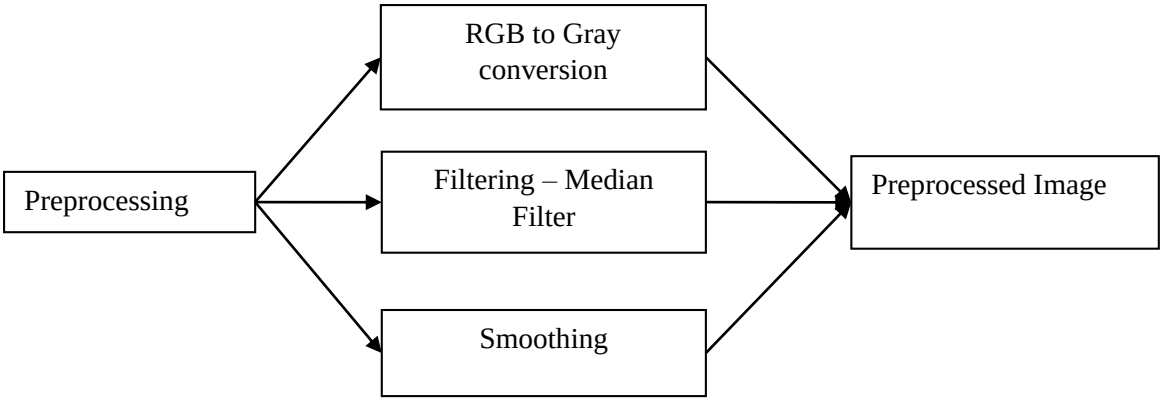
5.1 Data flow Diagrams

A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects**.**

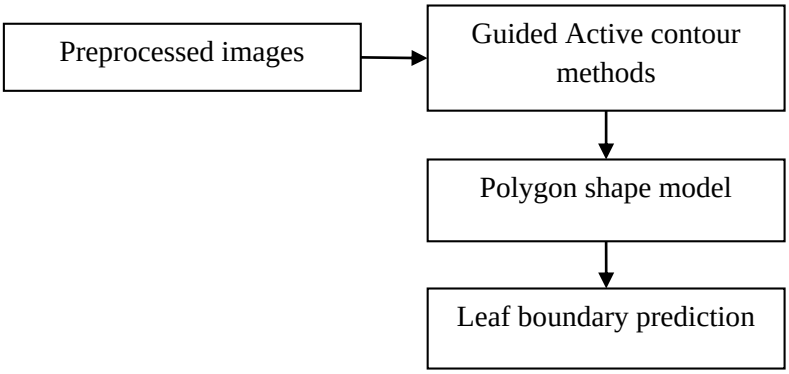
Level 0



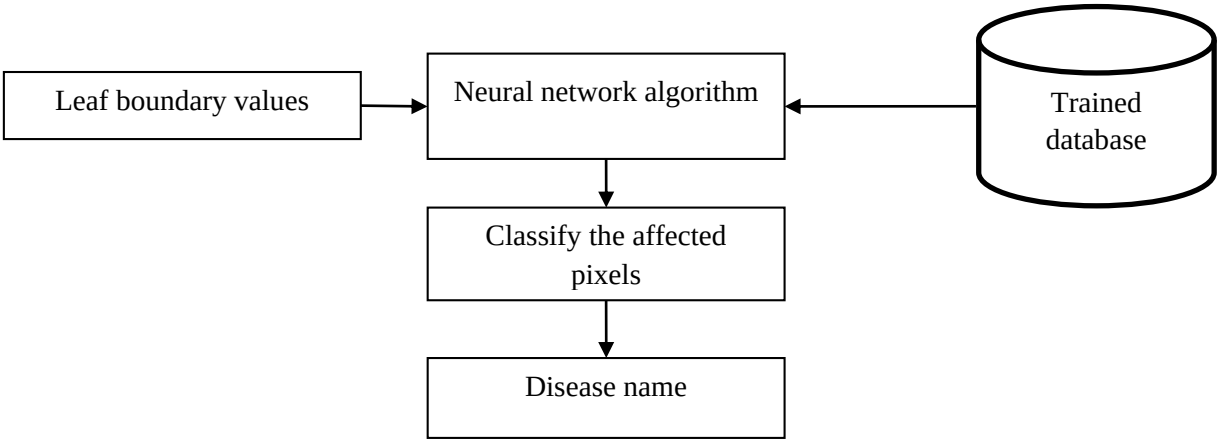
Level 1



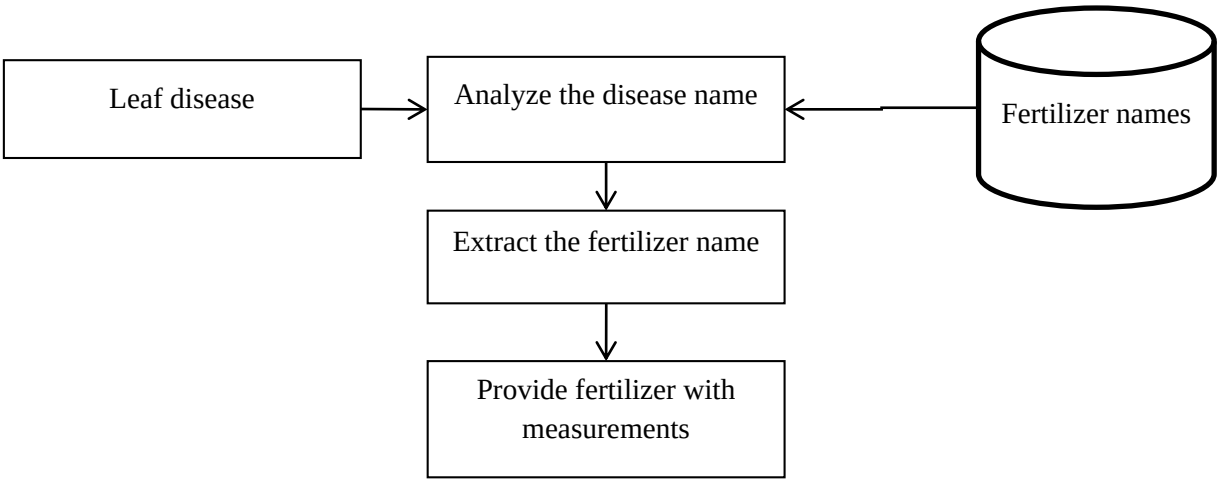
Level 2



Level 3

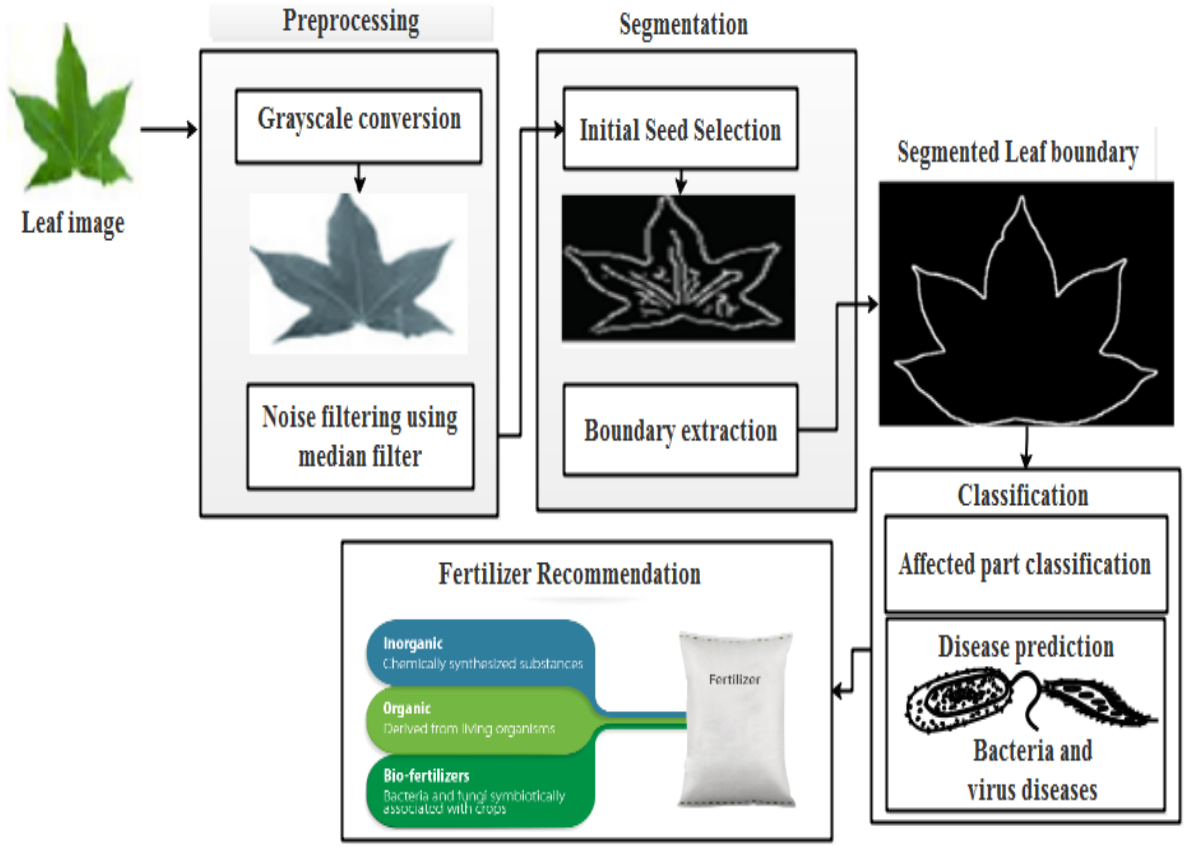


Level 4



5.2 Solution & Technical Architecture

System architecture involves the high level structure of software system abstraction, by using decomposition and composition, with architectural style and quality attributes. A software architecture design must conform to the major functionality and performance requirements of the system, as well as satisfy the non-functional requirements such as reliability, scalability, portability, and availability. System architecture must describe its group of components, their connections, interactions among them and deployment configuration of all components.



5.3 User Stories

User Type	Functional Requirements	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Customer	Accessing the Application	USN-l	As a user, I should be able to access the application from anywhere and use on any devices	User can access the application using the browser on any device	High	Sprint-4
	Uploading Image	USN-2	As a user, I should be able to upload images to predict the digits	User can upload images	High	Sprint-3
	Viewing the Results	USN-3	As a user, I should be able to view the results	The result of the prediction is displayed	High	Sprint-3
	Viewing Other Prediction	USN-4	As a user, I should be able to see other close predictions	The accuracy of other values must be displayed	Medium	Sprint-4
	Usage Instruction	USN-5	As a user, I should have a usage instruction to know how to use the application	The usage instruction is displayed on the home page	Medium	Sprint-4

6. PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning &Estimation

Title	Description	Status
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application	COMPLETED
Functional Requirement	Prepare the Functional Requirement Document	COMPLETED

Data Flow Diagram	Draw the data flow diagrams & submit for review	COMPLETED
Technology Architecture	Prepare the Technology Architecture Diagram	COMPLETED

Project Planning Phase

Title	Description	Status
Prepare Project Planning & Sprint Delivery Plan	Prepare the Product Backlog, Sprint planning, stories & Story points	COMPLETED
Prepare Milestone & Activity List	Prepare the Milestones & activity list of the project	COMPLETED

Ideation Phase

Title	Description	Status
Literature survey	Literature Survey on the selected projects & gathering information by referring the Technical papers etc.	COMPLETED
Brainstorm and Idea prioritization	List the Idea's by organising the brainstorming session & prioritize the top 4 Ideas based on the Feasibility & Importance	COMPLETED

6.2 Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Image Processing.	USN-1	As a user, I can retrieve useful information about the images.	1	Low	GOKUL S MURALIDHARAN R BALAVIGNESWAR E HIRTHIK R
Sprint-2	Model Building for Fruit Disease Prediction.	USN-2	As a user, I can able to predict fruit disease using this model.	1	Medium	GOKUL S MURALIDHARAN R BALAVIGNESWAR E HIRTHIK R
Sprint-2	Model Building for Vegetable Disease Prediction.	USN-3	As a user, I can able to predict vegetable disease using this model.	2	Medium	GOKUL S MURALIDHARAN R BALAVIGNESWAR E HIRTHIK R
Sprint-3	Application Building.	USN-4	As a user, I can see a web page for Fertilizers Recommendation System for Disease Prediction	2	High	GOKUL S MURALIDHARAN R BALAVIGNESWAR E HIRTHIK R
Sprint-4	Train The Model on IBM Cloud.	USN-5	As a user, I can save the information about Fertilizers and crops on IBM cloud	2	High	GOKUL S MURALIDHARAN R BALAVIGNESWAR E HIRTHIK R

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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	26 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	30 Oct 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	05 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	10 Nov 2022

7. CODING AND SOLUTIONING

Main.py 3 ●

D: > Fertilizers KerasCnn > Main.py > ...

```
    Label(testing_screen, text="").pack()
    Label(testing_screen, text="").pack()
    Label(testing_screen, text="").pack()
    Button(testing_screen, text='''Upload Image''', font=(
        'Verdana', 15), height="2", width="30", command=imgtest).pack()


global affect
def imgtest():


    import_file_path = filedialog.askopenfilename()

    image = cv2.imread(import_file_path)
    print(import_file_path)
    filename = 'Output/Out/Test.jpg'
    cv2.imwrite(filename, image)
    print("After saving image:")
    #result()

    #import_file_path = filedialog.askopenfilename()
    print(import_file_path)
    fnm = os.path.basename(import_file_path)
    print(os.path.basename(import_file_path))

   # file_sucess()

    print("\n********************\nImage : " + fnm + "\n********************")
    img = cv2.imread(import_file_path)
    if img is None:
        print('no data')
```

Main.py 3 ●

D: > Fertilizers KerasCnn > Main.py > ...

```
import tensorflow as tf
import numpy as np

from tkinter import *
import os
from tkinter import filedialog
import cv2
import time
from matplotlib import pyplot as plt
from tkinter import messagebox

def endprogram():
    print ("\nProgram terminated!")
    sys.exit()

def fulltraining():
    import model as mm

def testing():
    global testing_screen
    testing_screen = Toplevel(main_screen)
    testing_screen.title("Testing")
    # login_screen.geometry("400x300")
    testing_screen.geometry("600x450+650+150")
    testing_screen.minsize(120, 1)
    testing_screen.maxsize(1604, 881)
    testing_screen.resizable(1, 1)
    testing_screen.configure(bg='green')
    # login_screen.title("New Toplevel")

    Label(testing_screen, text='''Upload Image''', disabledforeground="#a3a3a3",
          foreground="#000000", width="300", height="2", font=("Calibri", 16)).pack()
```

Main.py 3 ●

D: > Fertilizers KerasCnn > Main.py > ...

```
60      print("\n*********************\nImage : " + fnm + "\n*********************")
61      img = cv2.imread(import_file_path)
62      if img is None:
63          print('no data')
64
65      img1 = cv2.imread(import_file_path)
66      print(img.shape)
67      img = cv2.resize(img, ((int)(img.shape[1] / 5), (int)(img.shape[0] / 5)))
68      original = img.copy()
69      neworiginal = img.copy()
70      cv2.imshow('original', img1)
71      gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
72
73      img1S = cv2.resize(img1, (960, 540))
74
75      cv2.imshow('Original image', img1S)
76      grayS = cv2.resize(gray, (960, 540))
77      cv2.imshow('Gray image', grayS)
78
79      dst = cv2.fastNlMeansDenoisingColored(img1, None, 10, 10, 7, 21)
80      cv2.imshow("Nosie Removal", dst)
81
82      thresh = 127
83      im_bw = cv2.threshold(grayS, thresh, 255, cv2.THRESH_BINARY)[1]
84      #cv2.imshow("affect Removal", im_bw)
85      number_of_black_pix = np.sum(im_bw == 0)
86      #print(number_of_black_pix)
87      #if(number_of_black_pix<5000):
88          #affect =
89
90
91      result()
```

8. TESTING

8.1 TEST CASES

A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly. A test case is a set of instructions on "HOW" to validate a particular test objective/target, which when followed will tell us if the expected behavior of the system is satisfied or not.

Characteristics of a good test case:

- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.
- Reusable: Can be reused if necessary

S.NO	FUNCTION	DESCRIPTION	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
1	Framework construction	Generate the GUI for admin and user	Individual page for admin and user	Individual page for admin and user	Success
2	Read the comments	Comments analysis	Comments in text format	Comments in text format	Success
3	Classification	Classify the datasets	Negative comments	Negative comments	Success
4	Rules implementation	Block the comments and friends	Block the users	Block the users	Success

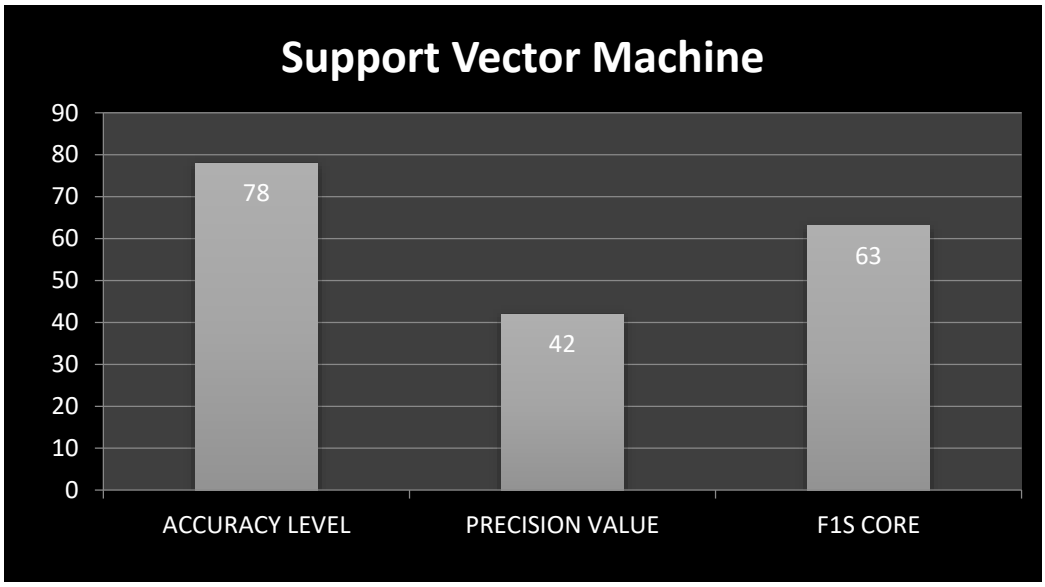
8.2 USER ACCEPTANCE TESTING

Acceptance testing can be defined in many ways, but a simple definition is the succeeds when the software functions in a manner that can be reasonable expected by the customer. After the acceptance test has been conducted, one of the two possible conditions exists. This is to fine whether the inputs are accepted by the database or other validations. For example accept only numbers in the numeric field, date format data in the date field. Also the null check for the not null fields. If any error occurs then show the error messages. The function of performance characteristics to specification and is accepted. A deviation from specification is uncovered and a deficiency list is created. User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

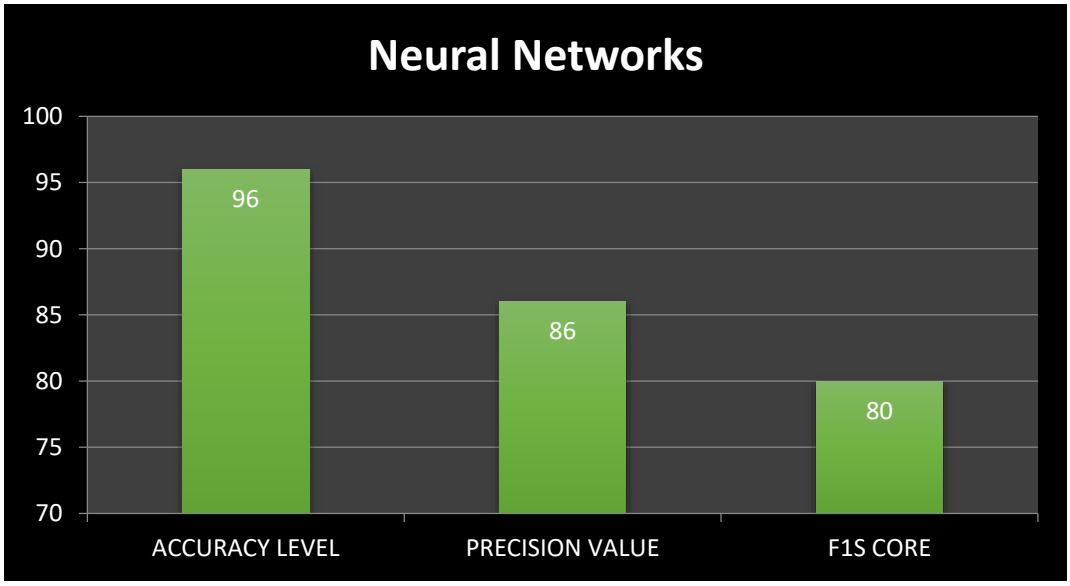
9.RESULTS

9.1 PERFORMANCE METRICS

Existing Algorithm



Proposed Algorithm



10. ADVANTAGES & DISADVANTAGES

DISADVANTAGES

- Suffer in the local minima problems.
- Dimensionality is high to produce large number of irrelevant features.
- User defined segmentation can be done.
- Does not recommend the fertilizers to leaves diseases.
- Manual approach is used

ADVANTAGES

- Segmentation can be done easily to spilt the tree parts.
- Classify the affected parts in leaves.
- Eliminate redundant features of images.
- Provide improved accuracy rate.

11. CONCLUSION

We presented a machine learning approach for crop yield prediction, which demonstrated superior performance in Crop Challenge using large datasets of products. The approach used deep neural networks to make yield predictions (including yield, check yield, and yield difference) based on genotype and environment data. The carefully designed deep neural networks were able to learn nonlinear and complex relationships between genes, environmental conditions, as well as their interactions from historical data and make reasonably accurate predictions of yields for new hybrids planted in new locations with known weather conditions. Performance of the model was found to be relatively sensitive to the quality of weather prediction, which suggested the importance of weather prediction techniques. We trained two deep neural networks, one for yield and the other for check yield, and then used the difference of their outputs as the prediction for yield difference. This model structure was found to be more effective than using one single neural network for yield difference, because the genotype and environment effects are more directly related to the yield and check yield than their difference. In modern era, the deep neural network is the prominent tool in agricultural industry for providing support to farmers in monitoring crop yield based on multiple parameters. Thus, the machine learning model provides high accuracy in detecting the suitable crop identification compared to other methodologies.

12. FUTURE SCOPE

The proposed model in this project work can be extended to image recognition. The entire model can be converted to application software using python to exe software. The real time image classification, image recognition and video processing are possible with help OpenCV python library. This project work can be extended for security applications such as figure print recognition, iris recognition and face recognition.

13.APPENDIX

SOURCE CODE

```
import tensorflow as tf
import time
import numpy as np
import os

start = time.time()
#try:
# Total iterations
final_iter = 1000

# Assign the batch value
batch_size = 20

# 20% of the data will automatically be used for validation
validation_size = 0.2
img_size = 128
num_channels = 3
train_path = r'data\Train'

# Prepare input data
if not os.path.exists(train_path):
print("No such directory")
raise Exception
classes = os.listdir(train_path)
num_classes = len(classes)

# We shall load all the training and validation images and labels into memory
using openCV and use that during training
data = dataset.read_train_sets(train_path, img_size, classes,
validation_size=validation_size)

# Display the stats
print("Complete reading input data. Will Now print a snippet of it")
print("Number of files in Training-set:\t\t{}".format(len(data.train.labels)))
print("Number of files in Validation-set:\t{}".format(len(data.valid.labels)))
session = tf.compat.v1.Session()
x = tf.compat.v1.placeholder(tf.float32, shape=[None, img_size, img_size,
num_channels], name='x')

## labels
y_true = tf.compat.v1.placeholder(tf.float32, shape=[None, num_classes],
name='y_true')
y_true_cls = tf.argmax(y_true, dimension=1)

##Network graph params
filter_size_conv1 = 3
num_filters_conv1 = 32

filter_size_conv2 = 3
num_filters_conv2 = 32

filter_size_conv3 = 3
num_filters_conv3 = 64
```

```
fc_layer_size = 128


def create_weights(shape):
return tf.Variable(tf.random.truncated_normal(shape, stddev=0.05))


def create_biases(size):
return tf.Variable(tf.constant(0.05, shape=[size]))



def make_generator_model(input,
                                num_input_channels,
                                conv_filter_size,
                                num_filters):
## We shall define the weights that will be trained using create_weights function.
weights = create_weights(shape=[conv_filter_size, conv_filter_size,
num_input_channels, num_filters])
## We create biases using the create_biases function. These are also trained.
biases = create_biases(num_filters)

## Creating the convolutional layer
layer = tf.nn.conv2d(input=input,
filter=weights,
strides=[1, 1, 1, 1],
padding='SAME')

    layer += biases

## We shall be using max-pooling.
layer = tf.nn.max_pool(value=layer,
ksize=[1, 2, 2, 1],
strides=[1, 2, 2, 1],
padding='SAME')
## Output of pooling is fed to Relu which is the activation function for us.
layer = tf.nn.relu(layer)

return layer


# Function to create a Flatten Layer
def create_flatten_layer(layer):
# We know that the shape of the layer will be [batch_size img_size img_size
num_channels]
    # But let's get it from the previous layer.
layer_shape = layer.get_shape()

## Number of features will be img_height * img_width* num_channels. But we shall
calculate it in place of hard-coding it.
num_features = layer_shape[1:4].num_elements()

## Now, we Flatten the layer so we shall have to reshape to num_features
layer = tf.reshape(layer, [-1, num_features])

return layer


# Function to create a Fully - Connected Layer
```

```
def create_fc_layer(input,
                    num_inputs,
                    num_outputs,
                    use_relu=True):
# Let's define trainable weights and biases.
weights = create_weights(shape=[num_inputs, num_outputs])
    biases = create_biases(num_outputs)

# Fully connected layer takes input x and produces wx+b.Since, these are matrices,
we use matmul function in Tensorflow
layer = tf.matmul(input, weights) + biases
if use_relu:
        layer = tf.nn.relu(layer)

return layer


# Create all the layers
layer_conv1 = make_generator_model(input=x,
num_input_channels=num_channels,
conv_filter_size=filter_size_conv1,
num_filters=num_filters_conv1)
layer_conv2 = make_generator_model(input=layer_conv1,
num_input_channels=num_filters_conv1,
conv_filter_size=filter_size_conv2,
num_filters=num_filters_conv2)

layer_conv3 = make_generator_model(input=layer_conv2,
num_input_channels=num_filters_conv2,
conv_filter_size=filter_size_conv3,
num_filters=num_filters_conv3)

layer_flat = create_flatten_layer(layer_conv3)

layer_fc1 = create_fc_layer(input=layer_flat,
num_inputs=layer_flat.get_shape()[1:4].num_elements(),
num_outputs=fc_layer_size,
use_relu=True)

layer_fc2 = create_fc_layer(input=layer_fc1,
num_inputs=fc_layer_size,
num_outputs=num_classes,
use_relu=False)

y_pred = tf.nn.softmax(layer_fc2, name='y_pred')

y_pred_cls = tf.argmax(y_pred, dimension=1)
session.run(tf.compat.v1.global_variables_initializer())
cross_entropy = tf.nn.softmax_cross_entropy_with_logits_v2(logits=layer_fc2,
labels=y_true)
cost = tf.reduce_mean(cross_entropy)
optimizer = tf.compat.v1.train.AdamOptimizer(learning_rate=1e-4).minimize(cost)
correct_prediction = tf.equal(y_pred_cls, y_true_cls)
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

session.run(tf.compat.v1.global_variables_initializer())


# Display all stats for every epoch
```

```
def show_progress(epoch, feed_dict_train, feed_dict_validate, val_loss,
total_epochs):
    acc = session.run(accuracy, feed_dict=feed_dict_train)
    val_acc = session.run(accuracy, feed_dict=feed_dict_validate)
    msg = "Training Epoch {0}/{4} --- Training Accuracy: {1:>6.1%}, Validation
Accuracy: {2:>6.1%},  Validation Loss: {3:.3f}"
print(msg.format(epoch + 1, acc, val_acc, val_loss, total_epochs))


total_iterations = 0

saver = tf.compat.v1.train.Saver()

print("")


# Training Function
def train(num_iteration):
global total_iterations

for i in range(total_iterations,
                   total_iterations + num_iteration):

        x_batch, y_true_batch, _, cls_batch = data.train.next_batch(batch_size)
        x_valid_batch, y_valid_batch, _, valid_cls_batch =
data.valid.next_batch(batch_size)

        feed_dict_tr = {x: x_batch,
                        y_true: y_true_batch}
        feed_dict_val = {x: x_valid_batch,
                         y_true: y_valid_batch}

        session.run(optimizer, feed_dict=feed_dict_tr)

if i % int(data.train.num_examples / batch_size) == 0:
            val_loss = session.run(cost, feed_dict=feed_dict_val)
            epoch = int(i / int(data.train.num_examples / batch_size))
# print(data.train.num_examples)
            # print(batch_size)
            # print(int(data.train.num_examples/batch_size))
            # print(i)

total_epochs = int(num_iteration / int(data.train.num_examples / batch_size)) + 1
show_progress(epoch, feed_dict_tr, feed_dict_val, val_loss, total_epochs)
            saver.save(session, 'trained_model')

    total_iterations += num_iteration


train(num_iteration=final_iter)


#except Exception as e:
    #print("Exception:",e)

# Calculate execution time
end = time.time()
dur = end-start
print("")
```

```
if dur<60:
print("Execution Time:",dur,"seconds")
elif dur>60 and dur<3600:
    dur=dur/60
print("Execution Time:",dur,"minutes")
else:
    dur=dur/(60*60)
print("Execution Time:",dur,"hours")
from flask import Flask, render_template, flash, request, session,send_file
from flask import render_template, redirect, url_for, request
import warnings
import datetime
import cv2
import tensorflow as tf
import numpy as np

from tkinter import *
import os


app = Flask(__name__)
app.config['DEBUG']
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'

@app.route("/")
def homepage():

return render_template('index.html')




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



@app.route("/train", methods=['GET', 'POST'])
def train():
if request.method == 'POST':
import model as model

return render_template('Tranning.html')




@app.route("/testimage", methods=['GET', 'POST'])
def testimage():
if request.method == 'POST':


        file = request.files['fileupload']
        file.save('data/alien_test/Test.jpg')
```

```
img = cv2.imread('data/alien_test/Test.jpg')



        train_path = r'data\train'
if not os.path.exists(train_path):
print("No such directory")
raise Exception
# Path of testing images
dir_path = r'data\alien_test'
if not os.path.exists(dir_path):
print("No such directory")
raise Exception

# Walk though all testing images one by one
for root, dirs, files in os.walk(dir_path):
for name in files:

print("")
                image_path = name
                filename = dir_path + '\\' + image_path
print(filename)
                image_size = 128
num_channels = 3
images = []

if os.path.exists(filename):

# Reading the image using OpenCV
image1 = cv2.imread(filename)

                    import_file_path = filename

                    image = cv2.imread(import_file_path)
                    fnm = os.path.basename(import_file_path)
                    filename = 'Test.jpg'
cv2.imwrite(filename, image)
# print("After saving image:")

print("\n**********************\nImage : " + fnm + "\n**********************")
                    img = cv2.imread(import_file_path)
if img is None:
print('no data')

                    img1 = cv2.imread(import_file_path)
print(img.shape)
                    img = cv2.resize(img, ((int)(img.shape[1] / 5),
(int)(img.shape[0] / 5)))
                    original = img.copy()
                    neworiginal = img.copy()
                    cv2.imshow('original', img1)
                    gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

                    cv2.imshow('Original image', img1)
                    orimage = 'static/Out/Test.jpg'
cv2.imwrite(orimage, img1)
```

```
                    cv2.imshow('Gray image', gray)

                    gry = 'static/Out/gry.jpg'

cv2.imwrite(gry, gray)


                    p = 0
for i in range(img.shape[0]):

for j in range(img.shape[1]):
                            B = img[i][j][0]
                            G = img[i][j][1]
                            R = img[i][j][2]
if (B >110 and G >110 and R >110):
                                p += 1

totalpixels = img.shape[0] * img.shape[1]
                    per_white = 100 * p / totalpixels
if per_white >10:
                        img[i][j] = [500, 300, 200]
                        cv2.imshow('color change', img)
# Guassian blur
blur1 = cv2.GaussianBlur(img, (3, 3), 1)
# mean-shift algo
newimg = np.zeros((img.shape[0], img.shape[1], 3), np.uint8)
                    criteria = (cv2.TERM_CRITERIA_EPS +
cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
                    img = cv2.pyrMeanShiftFiltering(blur1, 20, 30, newimg, 0,
criteria)
                    cv2.imshow('means shift image', img)

                    noise = 'static/Out/noise.jpg'

cv2.imwrite(noise, img)


# Guassian blur
blur = cv2.GaussianBlur(img, (11, 11), 1)

                    blur = cv2.GaussianBlur(img, (11, 11), 1)
# Canny-edge detection
canny = cv2.Canny(blur, 160, 290)
                    canny = cv2.cvtColor(canny, cv2.COLOR_GRAY2BGR)
# contour to find leafs
bordered = cv2.cvtColor(canny, cv2.COLOR_BGR2GRAY)
                    contours, hierarchy = cv2.findContours(bordered,
cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
                    maxC = 0
for x in range(len(contours)):
if len(contours[x]) > maxC:
                            maxC = len(contours[x])
                            maxid = x
perimeter = cv2.arcLength(contours[maxid], True)
# print perimeter
Tarea = cv2.contourArea(contours[maxid])
                    cv2.drawContours(neworiginal, contours[maxid], -1, (0, 0,
255))
                    cv2.imshow('Contour', neworiginal)
```

```
# cv2.imwrite('Contour complete leaf.jpg',neworiginal)
                    # Creating rectangular roi around contour
height, width, _ = canny.shape
                    min_x, min_y = width, height
                    max_x = max_y = 0
frame = canny.copy()
# computes the bounding box for the contour, and draws it on the frame,
for contour, hier in zip(contours, hierarchy):
                        (x, y, w, h) = cv2.boundingRect(contours[maxid])
                        min_x, max_x = min(x, min_x), max(x + w, max_x)
                        min_y, max_y = min(y, min_y), max(y + h, max_y)
if w >80 and h >80:
# cv2.rectangle(frame, (x,y), (x+w,y+h), (255, 0, 0), 2)   #we do not draw the
rectangle as it interferes with contour later on
roi = img[y:y + h, x:x + w]
                            originalroi = original[y:y + h, x:x + w]
if (max_x - min_x >0 and max_y - min_y >0):
                        roi = img[min_y:max_y, min_x:max_x]
                        originalroi = original[min_y:max_y, min_x:max_x]
                        cv2.rectangle(frame, (min_x, min_y), (max_x, max_y), (255,
0, 0),
2)  # we do not draw the rectangle as it interferes with contour
cv2.imshow('ROI', frame)

                    roi12 = 'static/Out/roi.jpg'

cv2.imwrite(roi12, frame)



                    cv2.imshow('rectangle ROI', roi)
img = roi
# Changing colour-space
                    # imghsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
imghls = cv2.cvtColor(roi, cv2.COLOR_BGR2HLS)
                    cv2.imshow('HLS', imghls)
                    imghls[np.where((imghls == [30, 200, 2]).all(axis=2))] = [0,
200, 0]
                    cv2.imshow('new HLS', imghls)
# Only hue channel
huehls = imghls[:, :, 0]
                    cv2.imshow('img_hue hls', huehls)
# ret, huehls = cv2.threshold(huehls,2,255,cv2.THRESH_BINARY)
huehls[np.where(huehls == [0])] = [35]
                    cv2.imshow('img_hue with my mask', huehls)
# Thresholding on hue image
ret, thresh = cv2.threshold(huehls, 28, 255, cv2.THRESH_BINARY_INV)
                    cv2.imshow('thresh', thresh)
# Masking thresholded image from original image
mask = cv2.bitwise_and(originalroi, originalroi, mask=thresh)
                    cv2.imshow('masked out img', mask)

# Resizing the image to our desired size and preprocessing will be done exactly as
done during training
image = cv2.resize(image1, (image_size, image_size), 0, 0, cv2.INTER_LINEAR)
                    images.append(image)
                    images = np.array(images, dtype=np.uint8)
                    images = images.astype('float32')
                    images = np.multiply(images, 1.0 / 255.0)
```

```
# The input to the network is of shape [None image_size image_size num_channels].
Hence we reshape.
x_batch = images.reshape(1, image_size, image_size, num_channels)

# Let us restore the saved model
sess = tf.compat.v1.Session()
# Step-1: Recreate the network graph. At this step only graph is created.
saver = tf.compat.v1.train.import_meta_graph('models/trained_model.meta')
# Step-2: Now let's load the weights saved using the restore method.
saver.restore(sess, tf.train.latest_checkpoint('./models/'))

# Accessing the default graph which we have restored
graph = tf.compat.v1.get_default_graph()

# Now, let's get hold of the op that we can be processed to get the output.
                    # In the original network y_pred is the tensor that is the
prediction of the network
y_pred = graph.get_tensor_by_name("y_pred:0")

## Let's feed the images to the input placeholders
x = graph.get_tensor_by_name("x:0")
                    y_true = graph.get_tensor_by_name("y_true:0")
                    y_test_images = np.zeros((1, len(os.listdir(train_path))))

# Creating the feed_dict that is required to be fed to calculate y_pred
feed_dict_testing = {x: x_batch, y_true: y_test_images}
                    result = sess.run(y_pred, feed_dict=feed_dict_testing)
# Result is of this format [[probabiliy_of_classA probability_of_classB ....]]
print(result)

# Convert np.array to list
a = result[0].tolist()
                    r = 0

# Finding the maximum of all outputs
max1 = max(a)
                    index1 = a.index(max1)
                    predicted_class = None

# Walk through directory to find the label of the predicted output
count = 0
for root, dirs, files in os.walk(train_path):
for name in dirs:
if count == index1:
                                predicted_class = name
                            count += 1

# If the maximum confidence output is largest of all by a big margin then
                    # print the class or else print a warning
for i in a:
if i != max1:
if max1 - i < i:
                                r = 1

out = ''

pre = ""
if r == 0:
```

```
print(predicted_class)

if (predicted_class == "Black spot"):
                            out = predicted_class

                            pre = 'Griffin  Fertilizer  reducing the fungus'

elif (predicted_class == "canker"):
                            out = predicted_class
                            pre = 'sprayed with Bordeaux mixture 1.0 per cent.'


elif (predicted_class == "greening"):
                            out =  predicted_class
                            pre =  'Mn-Zn-Fe-B micronutrient fertilizer'

elif (predicted_class == "healthy"):
                            out =  predicted_class
# messagebox.showinfo("Uses", '')
elif (predicted_class == "Melanose"):
                            out = predicted_class
                            pre =  'strobilurin fungicide'




else:

                        out = 'Could not classify with definite confidence'


else:
print("File does not exist")







        org = 'static/Out/Test.jpg'
gry ='static/Out/gry.jpg'
noise = 'static/Out/noise.jpg'
roi12 = 'static/Out/roi.jpg'




return
render_template('Test.html',result=out,org=org,gry=gry,inv=noise,noi=roi12,fer=pre
)



def sendmsg(targetno,message):
import requests

requests.post("http://smsserver9.creativepoint.in/api.php?username=fantasy&passwor
```

```
d=596692&to=" + targetno + "&from=FSSMSS&message=Dear user  your msg is " +
message + " Sent By FSMSG
FSSMSS&PEID=1501563800000030506&templateid=1507162882948811640")









if __name__ == '__main__':
    app.run(debug=True, use_reloader=True)
import cv2
import os
import glob
from sklearn.utils import shuffle
import numpy as np


def load_train(train_path, image_size, classes):
    images = []
    labels = []
    img_names = []
    cls = []

print('Going to read training images')
for fields in classes:
        index = classes.index(fields)
print('Now going to read {} files (Index: {})'.format(fields, index))
        path = os.path.join(train_path, fields, '*g')
        files = glob.glob(path)
for fl in files:
            image = cv2.imread(fl)
            image = cv2.resize(image, (image_size, image_size),0,0,
cv2.INTER_LINEAR)
            image = image.astype(np.float32)
            image = np.multiply(image, 1.0 / 255.0)
            images.append(image)
            label = np.zeros(len(classes))
            label[index] = 1.0
labels.append(label)
            flbase = os.path.basename(fl)
            img_names.append(flbase)
            cls.append(fields)
    images = np.array(images)
    labels = np.array(labels)
    img_names = np.array(img_names)
    cls = np.array(cls)

return images, labels, img_names, cls


class DataSet(object):

def __init__(self, images, labels, img_names, cls):
```

```
self._num_examples = images.shape[0]

self._images = images
self._labels = labels
self._img_names = img_names
self._cls = cls
self._epochs_done = 0
self._index_in_epoch = 0

@property
def images(self):
return self._images

@property
def labels(self):
return self._labels

@property
def img_names(self):
return self._img_names

@property
def cls(self):
return self._cls

@property
def num_examples(self):
return self._num_examples

@property
def epochs_done(self):
return self._epochs_done

def next_batch(self, batch_size):
"""Return the next `batch_size` examples from this data set."""
start = self._index_in_epoch
self._index_in_epoch += batch_size

if self._index_in_epoch >self._num_examples:
# After each epoch we update this
self._epochs_done += 1
start = 0
self._index_in_epoch = batch_size
assert batch_size <= self._num_examples
    end = self._index_in_epoch

return self._images[start:end], self._labels[start:end],
self._img_names[start:end], self._cls[start:end]


def read_train_sets(train_path, image_size, classes, validation_size):
class DataSets(object):
pass
data_sets = DataSets()

  images, labels, img_names, cls = load_train(train_path, image_size, classes)
  images, labels, img_names, cls = shuffle(images, labels, img_names, cls)

if isinstance(validation_size, float):
```

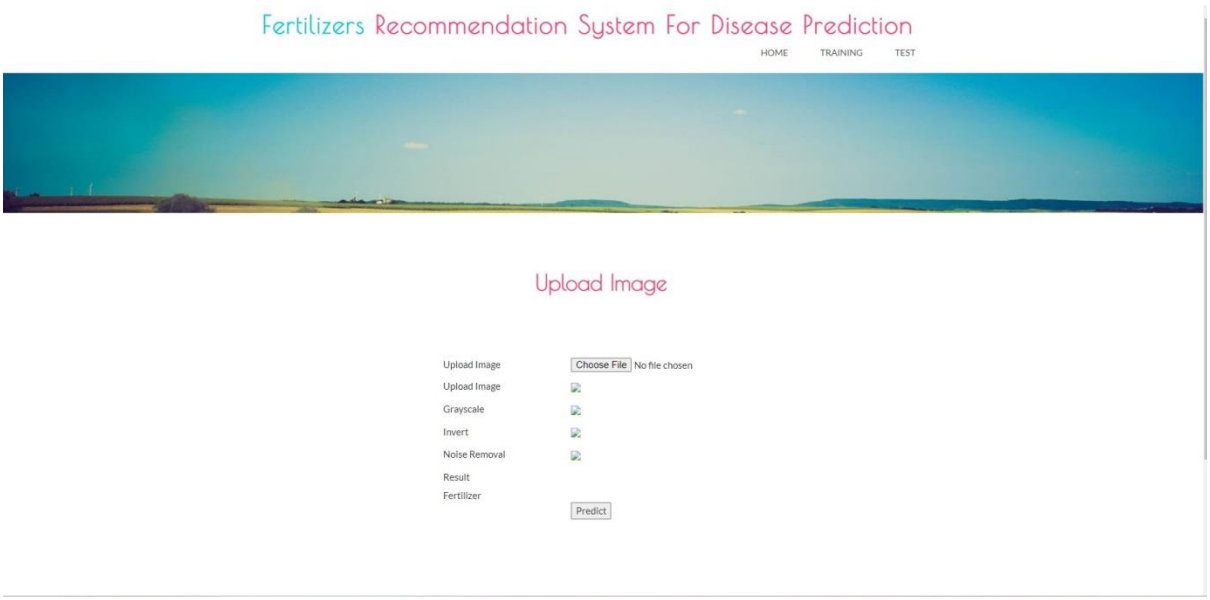
```
    validation_size = int(validation_size * images.shape[0])

  validation_images = images[:validation_size]
  validation_labels = labels[:validation_size]
  validation_img_names = img_names[:validation_size]
  validation_cls = cls[:validation_size]

  train_images = images[validation_size:]
  train_labels = labels[validation_size:]
  train_img_names = img_names[validation_size:]
  train_cls = cls[validation_size:]

  data_sets.train = DataSet(train_images, train_labels, train_img_names,
train_cls)
  data_sets.valid = DataSet(validation_images, validation_labels,
validation_img_names, validation_cls)

return data_sets
import tensorflow as tf
import numpy as np

from tkinter import *
import os
from tkinter import filedialog
import cv2
import time
from matplotlib import pyplot as plt
from tkinter import messagebox




def endprogram():
print ("\nProgram terminated!")
   sys.exit()









def training():

import Training as tr




def imgtraining():
    import_file_path = filedialog.askopenfilename()

    image = cv2.imread(import_file_path)
```

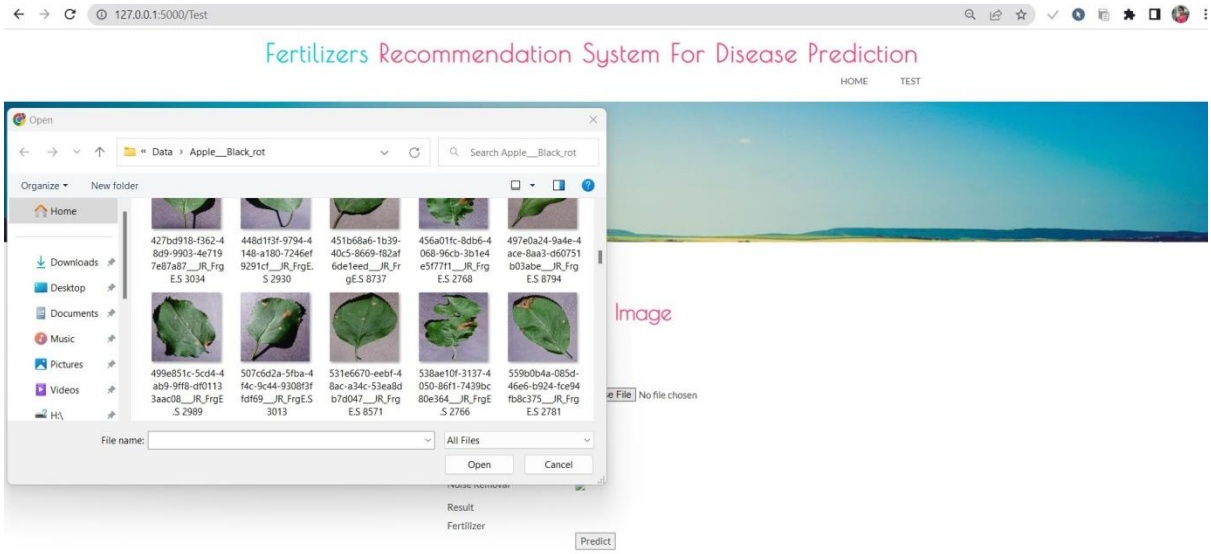
```
    filename = 'Test.jpg'
cv2.imwrite(filename, image)
print("After saving image:")

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    cv2.imshow('Original image', image)
    cv2.imshow('Gray image', gray)
# import_file_path = filedialog.askopenfilename()
print(import_file_path)
    fnm = os.path.basename(import_file_path)
print(os.path.basename(import_file_path))

from PIL import Image, ImageOps

    im = Image.open(import_file_path)
    im_invert = ImageOps.invert(im)
    im_invert.save('lena_invert.jpg', quality=95)
    im = Image.open(import_file_path).convert('RGB')
    im_invert = ImageOps.invert(im)
    im_invert.save('tt.png')
    image2 = cv2.imread('tt.png')
    cv2.imshow("Invert", image2)

"""----------------------------------------------"""

img = image

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    cv2.imshow('Original image', img)
#cv2.imshow('Gray image', gray)
dst = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 21)
    cv2.imshow("Nosie Removal", dst)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)



print("\n********************\nImage : " + fnm + "\n********************")
    img = cv2.imread(import_file_path)
if img is None:
print('no data')

    img1 = cv2.imread(import_file_path)
print(img.shape)
    img = cv2.resize(img, ((int)(img.shape[1] / 5), (int)(img.shape[0] / 5)))
original = img.copy()
neworiginal = img.copy()
    cv2.imshow('original', img1)
    gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

    cv2.imshow('Original image', img1)
# cv2.imshow('Gray image', gray)
p = 0
for i in range(img.shape[0]):

for j in range(img.shape[1]):
            B = img[i][j][0]
            G = img[i][j][1]
```

```
            R = img[i][j][2]
if (B >110 and G >110 and R >110):
                p += 1

totalpixels = img.shape[0] * img.shape[1]
    per_white = 100 * p / totalpixels
if per_white >10:
        img[i][j] = [500, 300, 200]
        cv2.imshow('color change', img)
# Guassian blur
blur1 = cv2.GaussianBlur(img, (3, 3), 1)
# mean-shift algo
newimg = np.zeros((img.shape[0], img.shape[1], 3), np.uint8)
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
    img = cv2.pyrMeanShiftFiltering(blur1, 20, 30, newimg, 0, criteria)
    cv2.imshow('means shift image', img)
# Guassian blur
blur = cv2.GaussianBlur(img, (11, 11), 1)
    cv2.imshow('Noise Remove', blur)
    corners = cv2.goodFeaturesToTrack(gray, 27, 0.01, 10)
    corners = np.int0(corners)

# we iterate through each corner,
    # making a circle at each point that we think is a corner.
for i in corners:
        x, y = i.ravel()
        cv2.circle(image, (x, y), 3, 255, -1)

    plt.imshow(image), plt.show()




def testing():
global testing_screen
    testing_screen = Toplevel(main_screen)
    testing_screen.title("Testing")
# login_screen.geometry("400x300")
testing_screen.geometry("600x450+650+150")
    testing_screen.minsize(120, 1)
    testing_screen.maxsize(1604, 881)
    testing_screen.resizable(1, 1)
# login_screen.title("New Toplevel")

Label(testing_screen, text='''Upload Image''', background="#d9d9d9",
disabledforeground="#a3a3a3",
foreground="#000000", bg="turquoise", width="300", height="2", font=("Calibri",
16)).pack()
    Label(testing_screen, text="").pack()
    Label(testing_screen, text="").pack()
    Label(testing_screen, text="").pack()
    Button(testing_screen, text='''Upload Image''', font=(
'Verdana', 15), height="2", width="30", command=imgtest).pack()

def imgtest():
    import_file_path = filedialog.askopenfilename()

    image = cv2.imread(import_file_path)
```

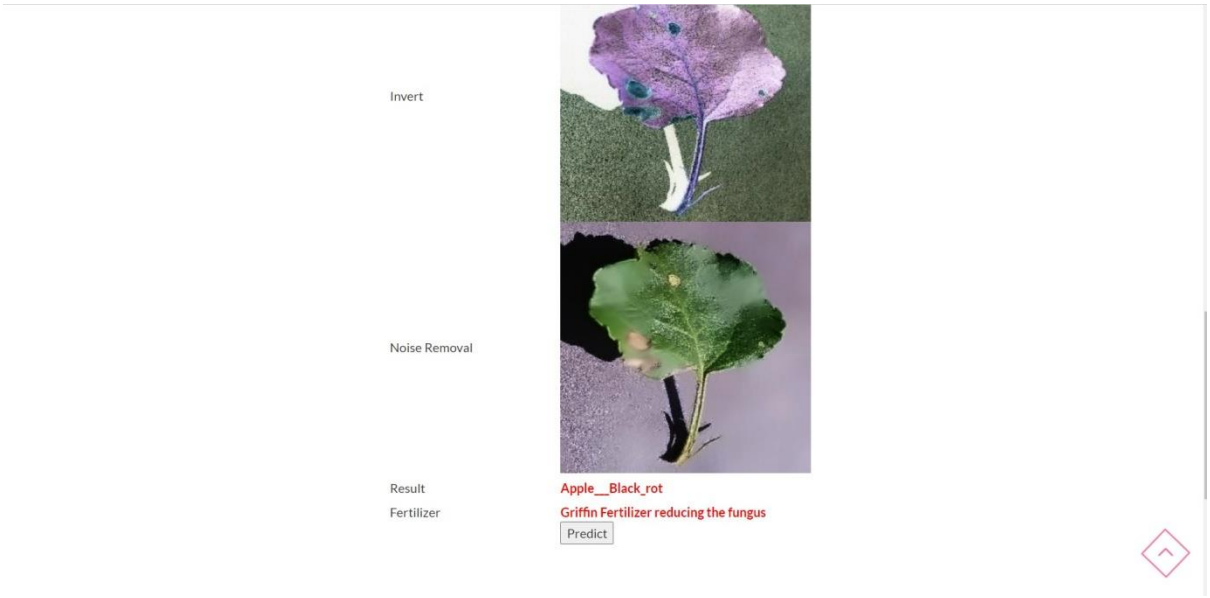
```
print(import_file_path)
    filename = 'data/alien_test/Test.jpg'
cv2.imwrite(filename, image)
print("After saving image:")

def main_account_screen():
from PIL import Image, ImageTk
global main_screen
    main_screen = Tk()
    width = 600
height = 600
screen_width = main_screen.winfo_screenwidth()
    screen_height = main_screen.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    main_screen.geometry("%dx%d+%d+%d" % (width, height, x, y))
    main_screen.resizable(0, 0)
# main_screen.geometry("300x250")
main_screen.title("Leaf Disease classification")

    Label(text="Leaf Disease classification", bg="turquoise", width="300",
height="5", font=("Calibri", 16)).pack()
    Label(text="").pack()
    Label(text="").pack()

    image = ImageTk.PhotoImage(Image.open('gui/12344.jpg'))

    Label(main_screen, text='Hello', image=image, compound='left', height="100",
width="200",).pack()

    Button(text="Training", font=(
'Verdana', 15), height="2", width="30", command=training,
highlightcolor="black").pack(side=TOP)
    Label(text="").pack()
    Button(text="Testing", font=(
'Verdana', 15), height="2", width="30", command=testing).pack(side=TOP)

    Label(text="").pack()

    main_screen.mainloop()


main_account_screen()
```

SCREENSHOTS:




127.0.0.1:5000/Test
Fertilizers Recommendation System For Disease Prediction
HOME
TEST
Open
Data › Apple__Black_rot
Search Apple__Black_rot
Organize
New folder
Home
Downloads
Desktop
Documents
Music
Pictures
Videos
427bd918-f362-48d9-9903-4e7197e87a87__JR_FrgE.S 3034
448d1f3f-9794-4148-a180-7246ef9291cf__JR_FrgE.S 2930
451b68a6-1b39-40c5-8669-f82af6de1eed__JR_FrgE.S 8737
456a01fc-8db6-4068-96cb-3b1e4e5f77f1__JR_FrgE.S 2768
497e0a24-9a4e-4ace-8aa3-d60751b03abe__JR_FrgE.S 8794
499e851c-5cd4-4ab9-9ff8-df01133aac08__JR_FrgE.S 2989
507c6d2a-5fba-4f4c-9c44-9308f3ffdf69__JR_FrgE.S 3013
531e6670-eebf-48ac-a34c-53ea8db7d047__JR_FrgE.S 8571
538ae10f-3137-4050-86f1-7439bc80e364__JR_FrgE.S 2766
559b0b4a-085d-46e6-b924-fce94fb8c375__JR_FrgE.S 2781
File name:
All Files
Open
Cancel
Image
No file chosen
Result
Fertilizer
Predict


Upload Image
Upload Image
Grayscale
Invert
Noise Removal
Result
Fertilizer
Potato__Early_blight
Griffin Fertilizer reducing the fungus
Predict



"E:\Project 2022\PYTHON\LeafWebsitePy\MultiClassificationImage\venv\Scripts\python.exe" "E:/Project 2022/PYTHON/LeafWebsitePy/MultiClassificationImage/Training.py"

Going to read training images

Now going to read Black spot files (Index: 0)

Now going to read canker files (Index: 1)

Now going to read greening files (Index: 2)

Now going to read healthy files (Index: 3)

Now going to read Melanose files (Index: 4)

Complete reading input data. Will Now print a snippet of it

Number of files in Training-set: 488

Number of files in Validation-set: 121

2022-11-01 14:01:22.019284: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2

WARNING:tensorflow:From E:/Project 2022/PYTHON/LeafWebsitePy/MultiClassificationImage/Training.py:40: calling argmax (from tensorflow.python.ops.math_ops) with dimension is deprecated and will be removed in a future version.

Instructions for updating:

Use the `axis` argument instead

WARNING:tensorflow:From E:/Project 2022/PYTHON/LeafWebsitePy/MultiClassificationImage/Training.py:82: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

Training Epoch 1/42 --- Training Accuracy: 45.0%, Validation Accuracy: 65.0%, Validation Loss: 1.396

Training Epoch 2/42 --- Training Accuracy: 20.0%, Validation Accuracy: 25.0%, Validation Loss: 1.316

Training Epoch 3/42 --- Training Accuracy: 20.0%, Validation Accuracy: 10.0%, Validation Loss: 1.259

Training Epoch 4/42 --- Training Accuracy: 30.0%, Validation Accuracy: 25.0%, Validation Loss: 1.241

Training Epoch 5/42 --- Training Accuracy: 35.0%, Validation Accuracy: 30.0%, Validation Loss: 1.229

Training Epoch 6/42 --- Training Accuracy: 40.0%, Validation Accuracy: 30.0%, Validation Loss: 1.207

Training Epoch 7/42 --- Training Accuracy: 40.0%, Validation Accuracy: 30.0%, Validation Loss: 1.191

Training Epoch 8/42 --- Training Accuracy: 40.0%, Validation Accuracy: 30.0%, Validation Loss: 1.211

Training Epoch 9/42 --- Training Accuracy: 40.0%, Validation Accuracy: 30.0%, Validation Loss: 1.198

Training Epoch 10/42 --- Training Accuracy: 40.0%, Validation Accuracy: 35.0%, Validation Loss: 1.148

Training Epoch 11/42 --- Training Accuracy: 40.0%, Validation Accuracy: 45.0%, Validation Loss: 1.107

Training Epoch 12/42 --- Training Accuracy: 40.0%, Validation Accuracy: 50.0%, Validation Loss: 1.065

Training Epoch 13/42 --- Training Accuracy: 40.0%, Validation Accuracy: 50.0%, Validation Loss: 1.041

Training Epoch 14/42 --- Training Accuracy: 50.0%, Validation Accuracy: 50.0%, Validation Loss: 0.996

Training Epoch 15/42 --- Training Accuracy: 55.0%, Validation Accuracy: 65.0%, Validation Loss: 0.931

Training Epoch 16/42 --- Training Accuracy: 55.0%, Validation Accuracy: 65.0%, Validation Loss: 0.880

Training Epoch 17/42 --- Training Accuracy: 55.0%, Validation Accuracy: 65.0%, Validation Loss: 0.849

Training Epoch 18/42 --- Training Accuracy: 55.0%, Validation Accuracy: 70.0%, Validation Loss: 0.811

Training Epoch 19/42 --- Training Accuracy: 65.0%, Validation Accuracy: 70.0%, Validation Loss: 0.787

Training Epoch 20/42 --- Training Accuracy: 65.0%, Validation Accuracy: 75.0%, Validation Loss: 0.762

Training Epoch 21/42 --- Training Accuracy: 65.0%, Validation Accuracy: 75.0%, Validation Loss: 0.741

Training Epoch 22/42 --- Training Accuracy: 65.0%, Validation Accuracy: 70.0%, Validation Loss: 0.718

Training Epoch 23/42 --- Training Accuracy: 65.0%, Validation Accuracy: 70.0%, Validation Loss: 0.698

Training Epoch 24/42 --- Training Accuracy: 70.0%, Validation Accuracy: 70.0%, Validation Loss: 0.679

Training Epoch 25/42 --- Training Accuracy: 70.0%, Validation Accuracy: 70.0%, Validation Loss: 0.655

Training Epoch 26/42 --- Training Accuracy: 70.0%, Validation Accuracy: 75.0%, Validation Loss: 0.635

Training Epoch 27/42 --- Training Accuracy: 75.0%, Validation Accuracy: 75.0%, Validation Loss: 0.607

Training Epoch 28/42 --- Training Accuracy: 75.0%, Validation Accuracy: 80.0%, Validation Loss: 0.586

Training Epoch 29/42 --- Training Accuracy: 80.0%, Validation Accuracy: 80.0%, Validation Loss: 0.560

Training Epoch 30/42 --- Training Accuracy: 85.0%, Validation Accuracy: 85.0%, Validation Loss: 0.532

Training Epoch 31/42 --- Training Accuracy: 85.0%, Validation Accuracy: 85.0%, Validation Loss: 0.515

Training Epoch 32/42 --- Training Accuracy: 90.0%, Validation Accuracy: 85.0%, Validation Loss: 0.507

Training Epoch 33/42 --- Training Accuracy: 90.0%, Validation Accuracy: 85.0%, Validation Loss: 0.494

Training Epoch 34/42 --- Training Accuracy: 90.0%, Validation Accuracy: 85.0%, Validation Loss: 0.478

Training Epoch 35/42 --- Training Accuracy: 90.0%, Validation Accuracy: 85.0%, Validation Loss: 0.471

Training Epoch 36/42 --- Training Accuracy: 90.0%, Validation Accuracy: 85.0%, Validation Loss: 0.452

Training Epoch 37/42 --- Training Accuracy: 90.0%, Validation Accuracy: 85.0%, Validation Loss: 0.453

Training Epoch 38/42 --- Training Accuracy: 90.0%, Validation Accuracy: 85.0%, Validation Loss: 0.433

Training Epoch 39/42 --- Training Accuracy: 90.0%, Validation Accuracy: 85.0%, Validation Loss: 0.437

Training Epoch 40/42 --- Training Accuracy: 90.0%, Validation Accuracy: 85.0%, Validation Loss: 0.417

Training Epoch 41/42 --- Training Accuracy: 90.0%, Validation Accuracy: 85.0%, Validation Loss: 0.433

Training Epoch 42/42 --- Training Accuracy: 90.0%, Validation Accuracy: 85.0%, Validation Loss: 0.401

Execution Time: 3.9937183459599814 minutes

Process finished with exit code 0

GITHUB

https://github.com/IBM-EPBL/IBM-Project-19529-1659699229.git

PROJECT DEMO

https://drive.google.com/drive/folders/1MormULX_0NiKhbMurMnxatbQi8VjTlf9?usp=sharing