FERTILIZERS RECOMMENDATION SYSTEM FOR DISEASE PREDICTION PROJECT REPORT

SUBMITTED BY TEAM ID: PNT2022TMID26099

SRIPRIYA.R(211519106155)

SRIDEVI (211519106154)

SWARANA.C (211519106166)

MANJU.M (211519106081)

BACHELORS OF ENGINEERING IN ELECTRONICS AND COMMUNICATION ENGINEERING

PANIMALAR INSTITUTE OF TECHNOLOGY

CHENNAI

INDEX

1. INTRODUCTION
Project Overview
Purpose

2. LITERATURE SURVEY
Existing problem
References
Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION
Empathy Map Canvas
Ideation & Brainstorming
Proposed Solution
Problem Solution fit

4. REQUIREMENT ANALYSIS
Functional requirement
Non-Functional requirements

5. PROJECT DESIGN
Data Flow Diagrams
Solution & Technical Architecture
User Stories

6. PROJECT PLANNING & SCHEDULING
Sprint Planning & Estimation
Sprint Delivery Schedule
Reports from JIRA

7. CODING & SOLUTIONING
Feature 1
Feature 2

8. TESTING
Test Cases
User Acceptance Testing

9. RESULTS

Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX
Source Code
GitHub & Project Demo Link

INTRODUCTION

Farmers face several challenges when growing crops like uncertain irrigation, poor soil quality, etc. Especially in India, a major fraction of farmers do not have the knowledge to select appropriate crops and fertilizers. Moreover, crop failure due to disease causes a significant loss to the farmers, as well as the consumers. While there have been recent developments in the automated detection of these diseases using Machine Learning techniques, the utilization of Deep Learning has not been fully explored. Additionally, such models are not easy to use because of the high-quality data used in their training, lack of computational power, and poor generalizability of the models. To this end, we create an open-source easy-to-use web application to address some of these issues which may help improve crop production. The study of plant diseases is important as they cause loss to the plant as well as plant produce. The various types of losses occur in the field, in storage or any time between sowing and consumption of produce. The diseases are responsible for direct monitory loss and material loss.

OVERVIEW: In this project, two datasets name fruit dataset and vegetable dataset are collected. The collected datasets are trained and tested with deep learning neural network named Convolutional Neural Networks (CNN). First, the fruit dataset is trained and then tested with CNN. It has 6 classes and all the classes are trained and tested. Second, the vegetable dataset is trained and tested. The software used for training and testing of datasets is Python. All the Python codes are first written in Jupyter notebook supplied along with Anaconda Python and then the codes are tested in IBM cloud. Finally, a web-based framework is designed with help Flask a Python library. There are 2 html files are created in templates folder along with their associated files in static folder. The Python program 'app.py' used to interface with these two webpages is written in Spyder-Anaconda python and tested. 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.

LITERATURE SURVEY

Existing problem proposed a method for leaf disease detection and suggest fertilizers to cure leaf diseases. But the method involves less number of train and test sets which results in poor accuracy. a simple prediction method for soil-based fertilizer recommendation system for predicted crop diseases. This method gives

less accuracy and prediction. & another proposed an IoT based system for leaf disease detection and fertilizer recommendation which is based on Machine Learning techniques yields less 80 percentage accuracies. 2.2 Proposed solution In this project work, a deep learning based neural network is used to train the collected datasets and test the same. The deep learning based neural network is CNN which gives more than 90% classification accuracies. By increasing the more number of dense layers and by modifying hyperparameters such as number of epochs, batch size, the accuracy rate can be increased to 95% to 98%.

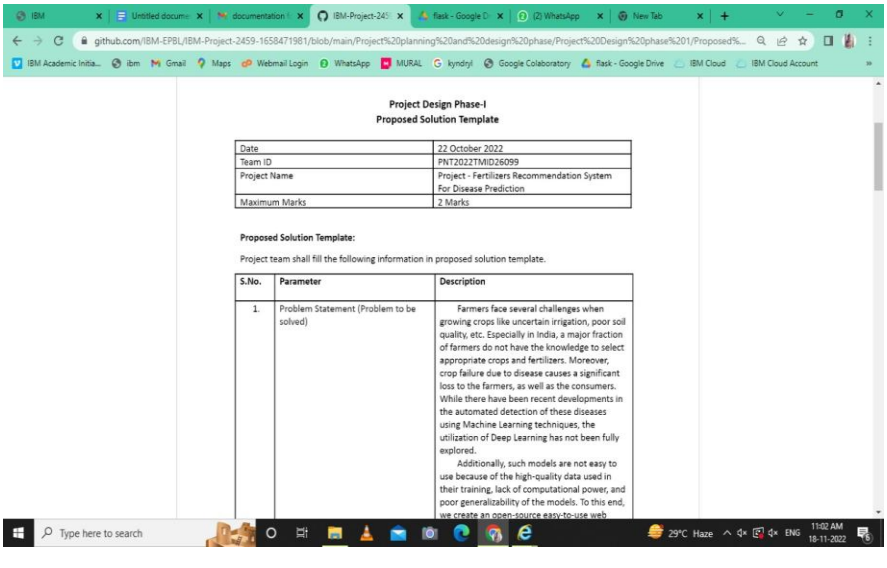
REFERENCES:

[1]. R Indumathi Leaf Disease Detection and Fertilizer Suggestion", IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), 29-30 March 2019, DOI: 10.1109/ICSCAN.2019.8878781. [2]. P. Pandi Selvi, P. Poornima, "Soil Based Fertilizer Recommendation System for Crop Disease Prediction System", International Journal of Engineering Trends and Applications (IJETA) – Volume 8 Issue 2, Mar-Apr 2021. [3]. H Shiva reddy, Ganesh hedge, Prof. DR Chinnaya3, "IoT based Leaf Disease Detection and Fertilizer Recommendation", International Research Journal of Engineering and Technology (IRJET), Volume: 06 Issue: 11, Nov 2019, e-ISSN: 2395- 0056.

PROBLEM STSTEMENT:

Agriculture is the most important sector in today's life. Most plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and a major threat to food security. Hence, early and accurate identification of plant diseases is essential to ensure high quantity and best quality. In recent years, the number of diseases on plants and the degree of harm caused has increased due to the variation in pathogen varieties, changes in cultivation methods, and inadequate plant protection techniques. So to overcome such cases An automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves This project aims at providing a system to support the cultivators in choosing the right fertilizers for their plants to counter the deficiency of nutrients that cause various infections and diseases. The below blocks define the problems

faced by the different users and the solutions that are provided by the system.



Project Design Phase-I
Proposed Solution Template

Date	22 October 2022
Team ID	PNT2022TMID26099
Project Name	Project - Fertilizers Recommendation System For Disease Prediction
Maximum Marks	2 Marks

Proposed Solution Template:

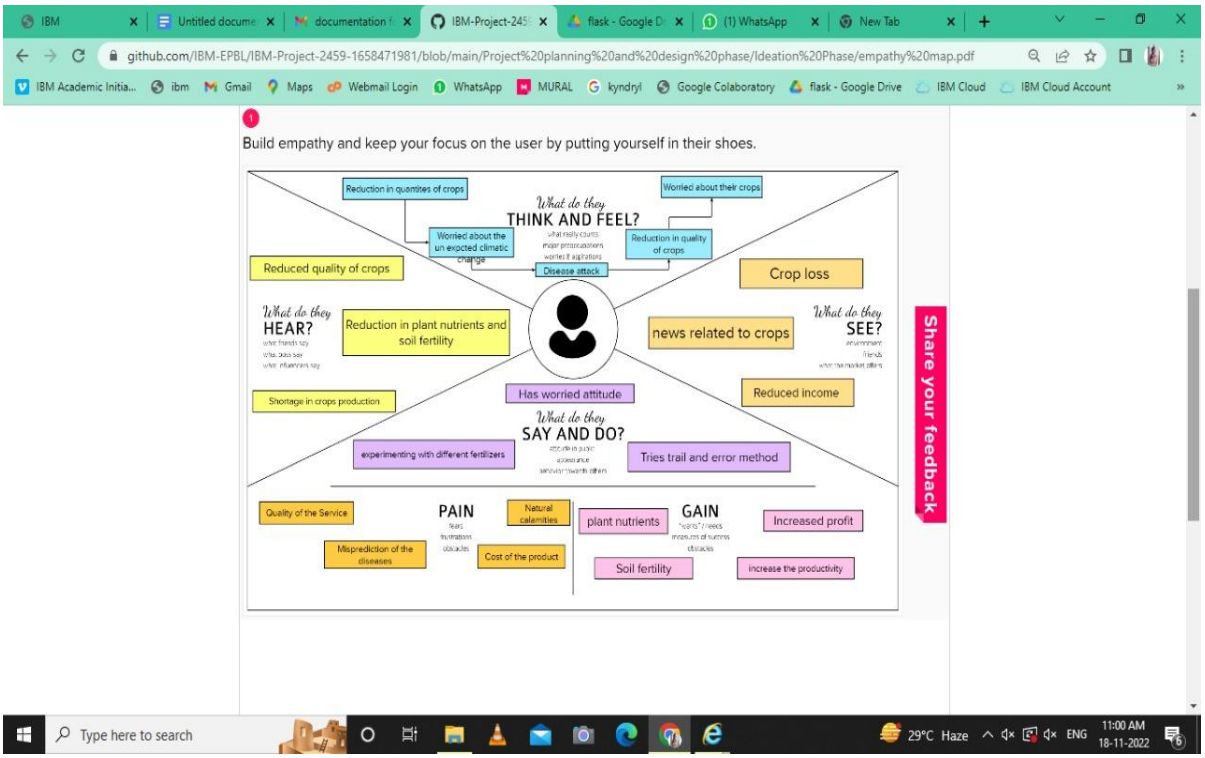
Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Farmers face several challenges when growing crops like uncertain irrigation, poor soil quality, etc. Especially in India, a major fraction of farmers do not have the knowledge to select appropriate crops and fertilizers. Moreover, crop failure due to disease causes a significant loss to the farmers, as well as the consumers. While there have been recent developments in the automated detection of these diseases using Machine Learning techniques, the utilization of Deep Learning has not been fully explored. Additionally, such models are not easy to use because of the high-quality data used in their training, lack of computational power, and poor generalizability of the models. To this end, we create an open-source easy-to-use web

IDEATION & PROPOSED SOLUTION :

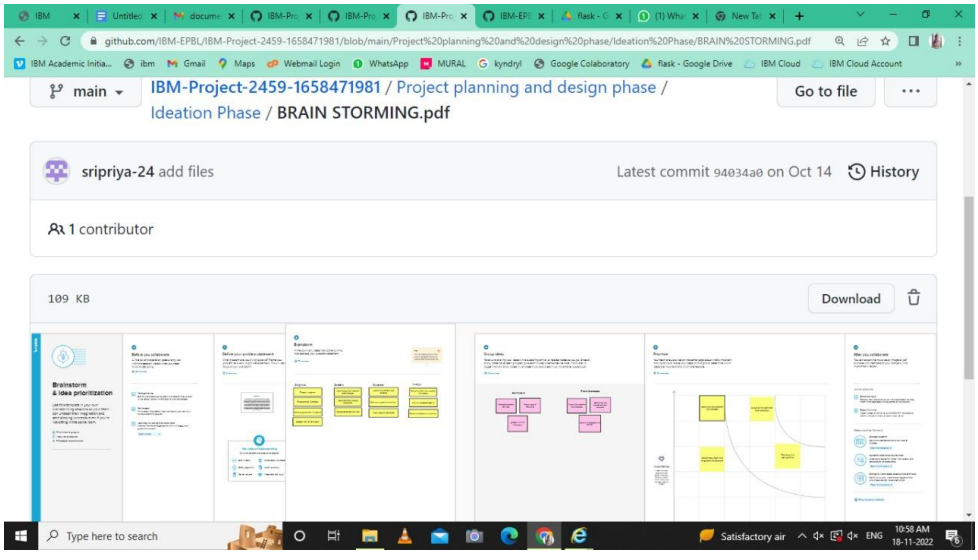
EMPATHY MAP CANVAS

An empathy map is used to gain deeper insights on the customer's interaction with the system. It gives an idea on what the user feels and experiences while using the system, what fears the user has respective to the system, etc. It also specifies how supportive the system environment is and what the users are likely to hear from the people around them regarding the usage of the system.



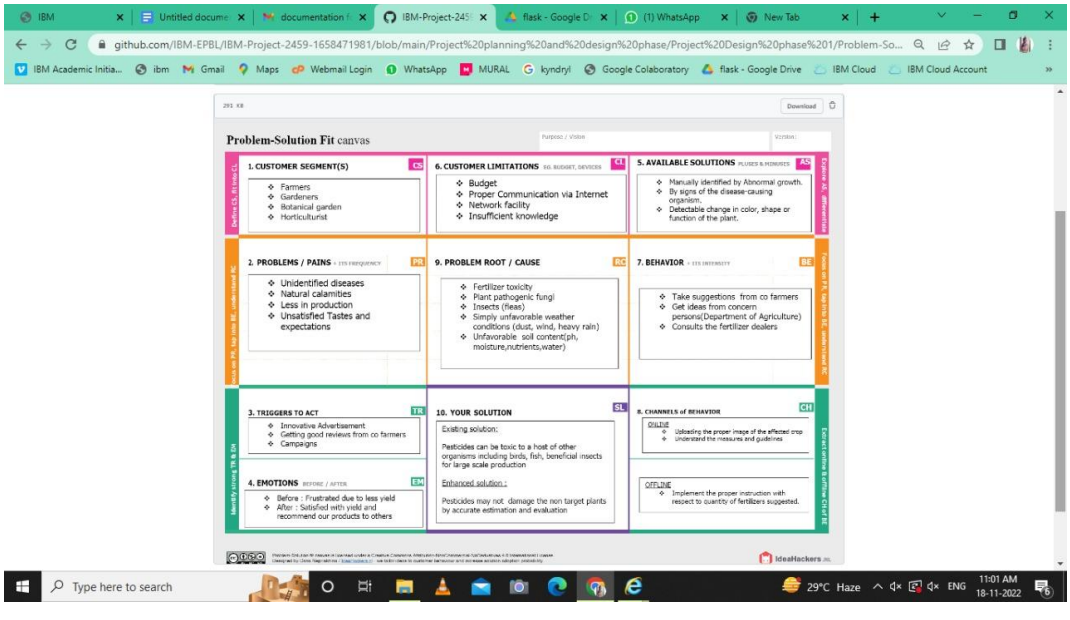
IDEATION & BRAINSTROMING

Ideation and Brainstorming are performed to generate ideas and solutions. Brainstorming is a group activity unlike ideation.



PROPOSED SOLUTION & PROBLEM SOLUTION FIT

An automated system that takes the images of plant parts as input identifies different diseases on plants by checking the symptoms shown on the leaves of the plant is built . Deep learning techniques are used to identify the diseases and suggest the fertilizes that can help cure the disease. The user need not consult any specialist for identification of diseases that affected the leaves or for the recommendation of the fertilizers.



REQUIREMENT ANALYSIS:

FUNCTIONAL REQUIREMENTS &NON FUNCTIONAL REQUIREMENTS

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registering through Gmail
FR-2	User confirmation	Confirmation is done through Email
FR-2	Image Capture	Take a picture of a leaf and verify that the leaf was captured using the specified criteria.
FR-3	Image Processing	Upload the image of the leaf for detecting the diseases that is present in the leaf.
FR-4	Leaf Prediction	Determine the parameter that should be taken into account for disease identification for identifying the leaf and predicting the disease in it.
FR-5	Image Description	Show the prescribed fertilizer that has to be used for the diseased leaf
FR-6	Providing Dataset	Training the datasets Testing the datasets
FR-7	Adding Datasets	Datasets for fruits and vegetables are added.

Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Data sets can be prepared according to the leaf .Leaf datasets can be used for detection of all kind of leaf's Datasets can be reusable to detect diseases present in leaf.
NFR-2	**Security**	User information and leaf data are secured The employed algorithms are more secure.
NFR-3	**Reliability**	The leaf quality is more for predicting the disease in leaf. The datasets and image capture consistently performs well.
NFR-4	**Performance**	The leaf problem is specified when the leaf is detected. Performs well according to the quality of the leaf and provides a specific cure to it by showing recommendation of fertilizer.
NFR-5	**Availability**	The quality of the leaf will be used again for detection. Datasets will be made available and easily accessible. It is available to all users to predict plant disease.
NFR-6	**Scalability**	Increasing the accuracy of disease prediction in the leaf.

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 Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Communication	Communication via Messages Communication via email
FR-4	User Training	Training via Online Classes Training via Seminars,Worshops etc

Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Easy to Use & Can easily get practiced & Adapted to the practice.
NFR-2	**Security**	No possibilities of any external attacks
NFR-3	**Reliability**	Fixing the Problems is quiet easy and Does not require more time
NFR-4	**Performance**	Can perform without any defects
NFR-5	**Availability**	The required Components & Software is easily available & Cost efficient
NFR-6	**Scalability**	Can function efficiently giving the Required target performance to the customer

THEORITICAL ANALYSIS:

step is the image dataset collection followed by image preprocessing. The third step is the training of image datasets with initializing different hyper parameters. Then build the model and save the model file with .h5 format. The final stage is the testing of existing or new datasets using the trained model. 3.2 Hardware/Software designing The software used for training and testing the dataset is Python. The Jupyter notebook (Notebook of IBM cloud also) is used for python programming. The neural network used for training and testing the model is Convolutional Neural Network (CNN).

The CNN has following layers:

1. Convolutional layer (32x32 kernal (3x3)) Image dataset collection Image Preprocessing Image dataset training Build & Save Mode Predict the test dataset
2. Max-pool layer (kernel(2x2))
3. Flatten layer
4. Dense layer (different layers with different size)

5. Drop out layer (optional)
6. Final output dense layer(size 6x1 for fruit dataset and 9x1 for Vegetable dataset) In the preprocessing step, images are normalized to 1 and then resized to 128x128. The images are arranged in different batch sizes. Then train set and test set are formed from the collected datasets.In order to do the above steps in Python, the following Python libraries must be imported before starting the process:

1. NumPy
2. TensorFlow
3. Keras
4. Matplotlib (optional for data visualization)

The following activation functions used in the CNN training

1. RELU at the end of convolution layer and Max Pool layer
2. SoftMax at the end of output dense layer
3. For testing the dataset argmax is used, its an optional

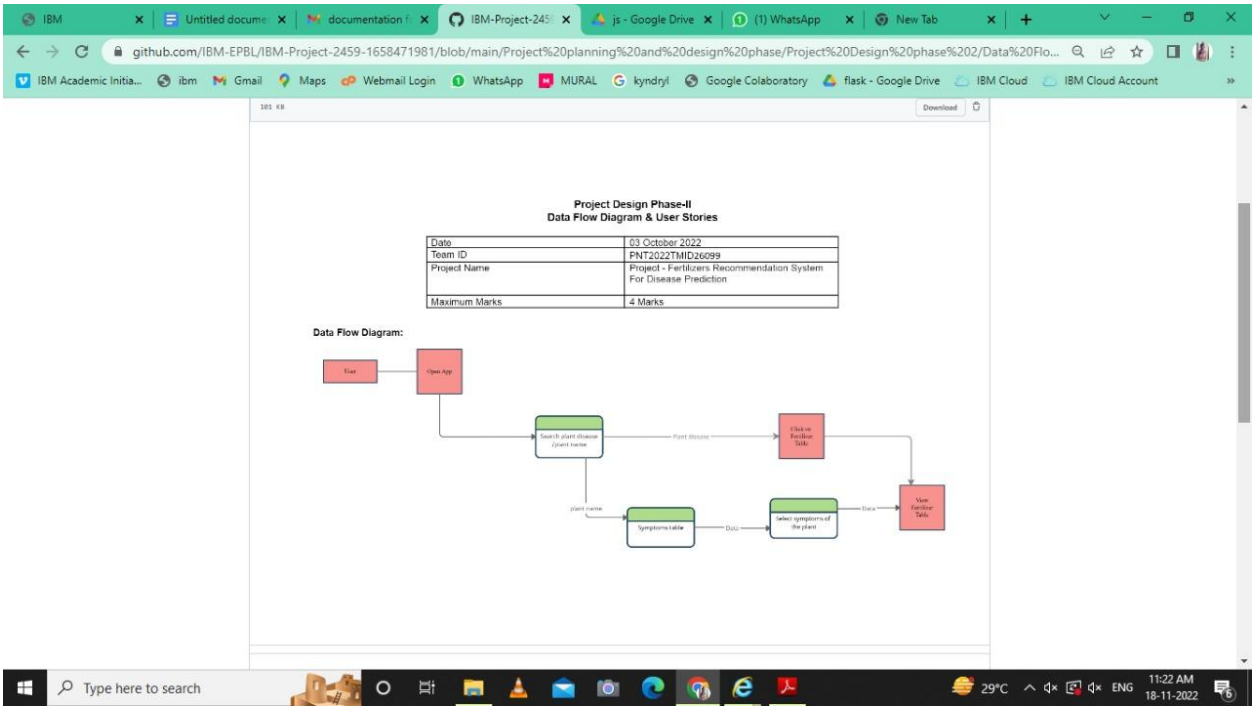
EXPERIMENTAL INVESTIGATIONS:

Analysis made while working on the solution The batch sizes are varied and tested. For different batch sizes, the CNN gives different accuracies. The batch size determines the number of iterations per epoch. Another important hyper parameter is the number of epochs. This determines accuracy and it has high influence on accuracy compared to other hyper parameters. The accuracy can be varied from 80% to 90% in vegetable dataset and 95% to 98% in the case of fruit dataset by increasing the number of epochs. The size of test dataset and train dataset also has very high influence on accuracies. The accuracy can be increased by using more number of images in train dataset. The computational time for model building is increased when the size of the train dataset increased and also number of epochs increased. The batch size of train dataset and test dataset also play a vital role in computational time. The Neural Network complexity is increased when more number of convolutional layers increased. If the number of layers increased, better accuracy result will obtain. At the same increasing the number of layers in CNN leads to more training time and also requires more time to build a model. The model .h5 size depends on the size of train datasets. But the memory requirement depends on the size of train dataset and CNN architecture complexity

PROJECT DESIGN

DATA FLOW DAIGRAMS

A data flow diagram or DFD(s) maps out the flow of information for any process or system. DFDs help you better understand process or system operation to discover potential problems, improve efficiency, and develop better processes.



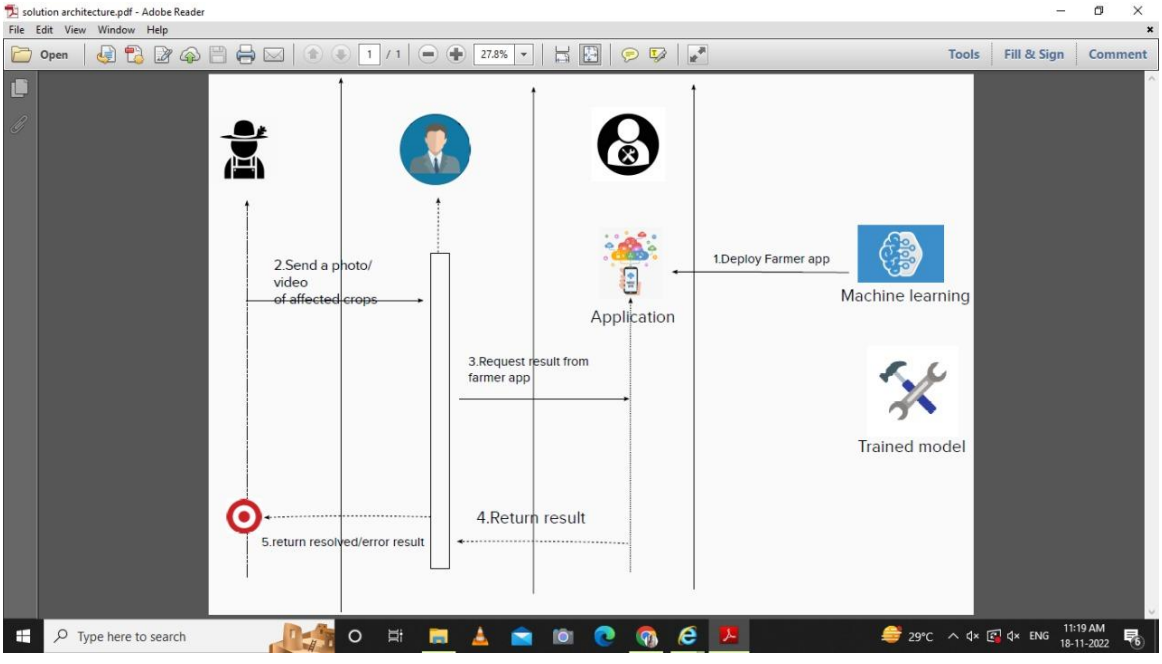
Project Design Phase-II
Data Flow Diagram & User Stories

Date	03 October 2022
Team ID	PNT2022TMID26099
Project Name	Project - Fertilizers Recommendation System For Disease Prediction
Maximum Marks	4 Marks

Data Flow Diagram:

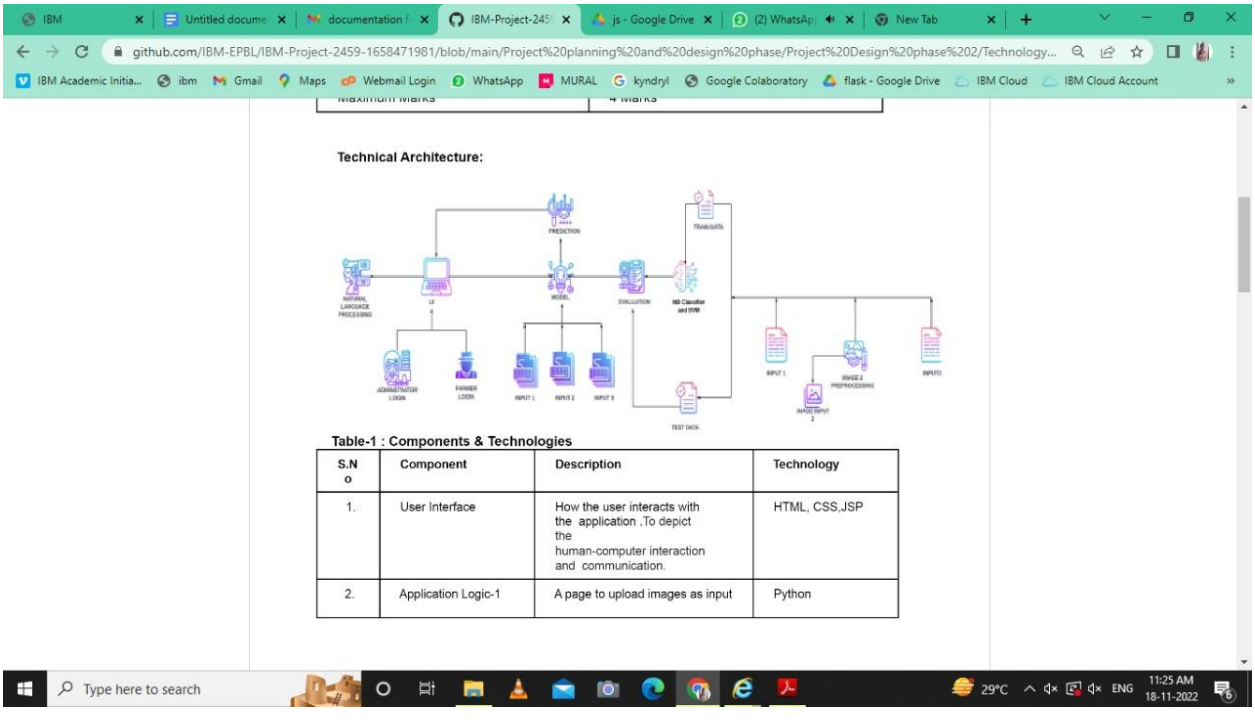
SOLUTION ARCHITECTURE:

Solution architecture is the process of developing solutions based on predefined processes, guidelines and best practices with the objective that the developed solution fits within the enterprise architecture in terms of information architecture, system portfolios, integration requirements, etc.



TECHNICAL ARCHITECTURE

Technical architecture involves the development of a technical blueprint regarding the arrangement, interaction, and interdependence of all elements so that system-relevant requirements are met.



Technical Architecture:

Table-1 : Components & Technologies

S.No	Component	Description	Technology
1.	User Interface	How the user interacts with the application .To depict the human-computer interaction and communication.	HTML, CSS,JSP
2.	Application Logic-1	A page to upload images as input	Python

USER STORIES

An informal, generic explanation of a software feature written from the viewpoint of the end user is known as a user story. Its objective is to explain how a software feature will benefit the user.

USER STORIES:

Use the below template to list all the user stories for the product.

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 providing my email address, password, and confirming my password .	I have access to my profile/dashboard.	High	Sprint-1
		USN-2	Once I have registered for the application, I will receive a confirmation email.	I can receive a confirmation email and click the confirm button.	High	Sprint-1
		USN-3	As a user, I can sign up for the application using Gmail.	I can use Gmail to access the application.	Medium	Sprint-1
	Login	USN-4	As a user, I can access the application by entering my email address and password.	I can make use of the Application for Disease Prediction	High	Sprint-1
Customer (Web user)	Registration	USN-5	As a Web user, I can register on the System with a User ID.	I can access the app like a website.	High	Sprint-1
Customer Care Executive	Customer Support	USN-6	As a supporter, I can see how customers use the product.	I can develop Customer Guidelines and Practices.	Low	Sprint-2
Administrator	Analyst	USN-7	As an admin, I can update several datasets about plant diseases.	I can store a significant amount of data.	High	Sprint-1
Customer Purpose	Prediction	USN-8	It use artificial intelligence to identify plant diseases in captured photographs and provides a live view of prediction.	I can predict plant disease.	High	Sprint-1

PROJECT PLANNING & SCHEDULING

The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole team.

SPRINT DELIVERY SCHEDULE

Agile sprints typically last from one week to one month. The goal of sprints is to put pressure on teams to innovate and deliver more quickly, hence the shorter the sprint, the better.

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	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	19 Nov 2022

Velocity:

Sprint 1 Average Velocity:
Average Velocity = 20/2 = 10

Sprint 2 Average Velocity:
Average Velocity = 20/2 = 10

Sprint 3 Average Velocity:
Average Velocity = 20/1 = 20

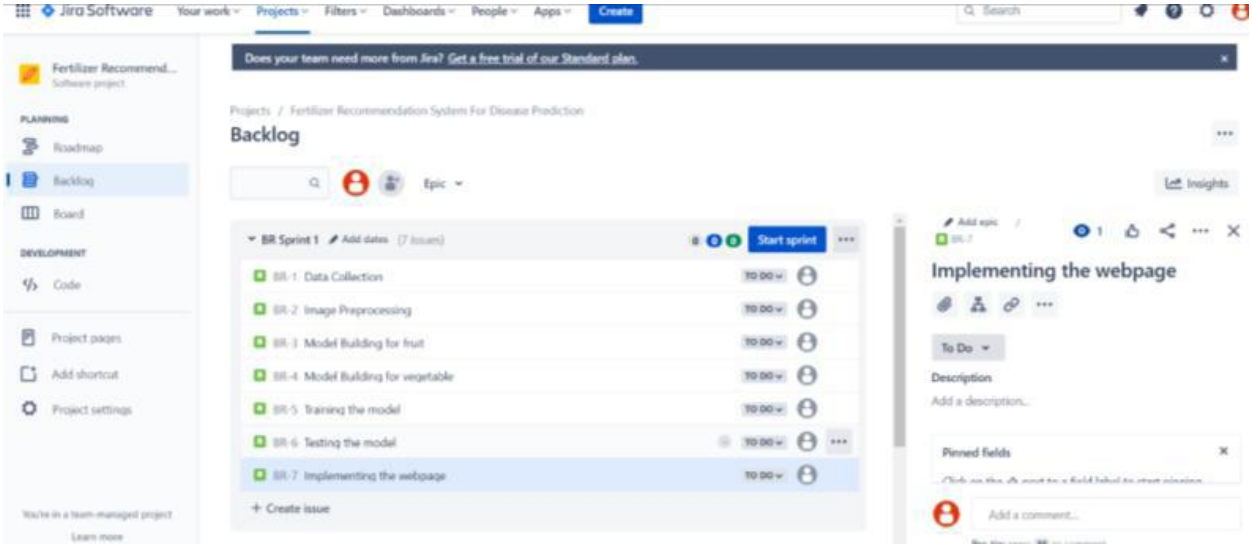
Sprint 4 Average Velocity:
Average Velocity = 20/2 = 10

Burndown Chart:

REPORTS FROM JIRA

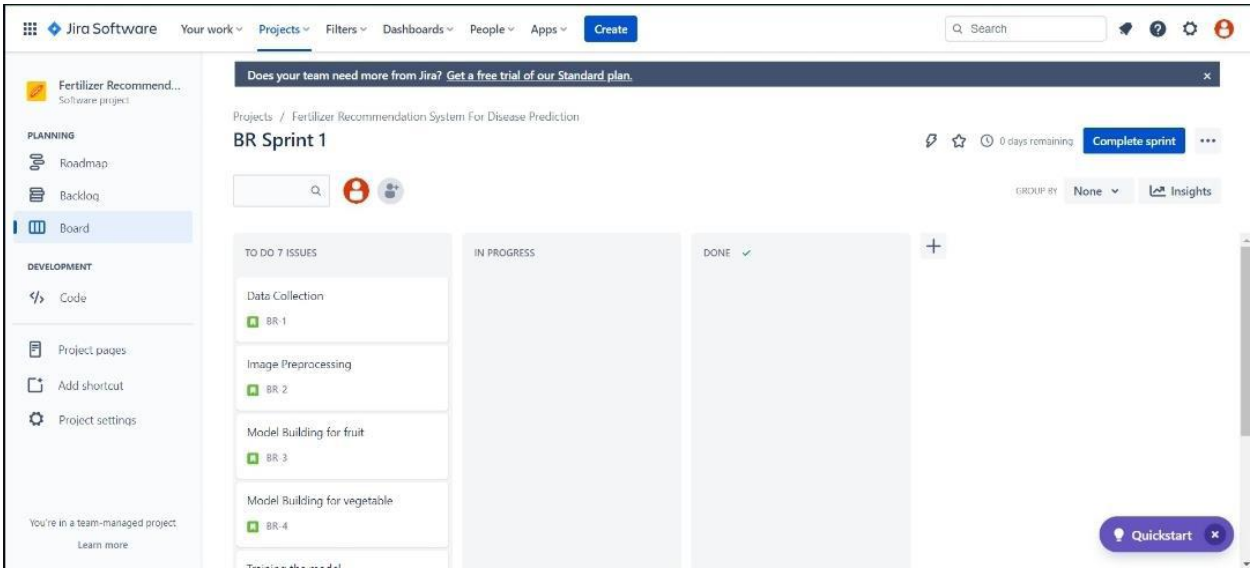
BACKLOG:

A backlog is a list of issues that's related to the project and the functions of the system. It makes it simple to make, store, manage a variety of problems including the ones the team is working on.



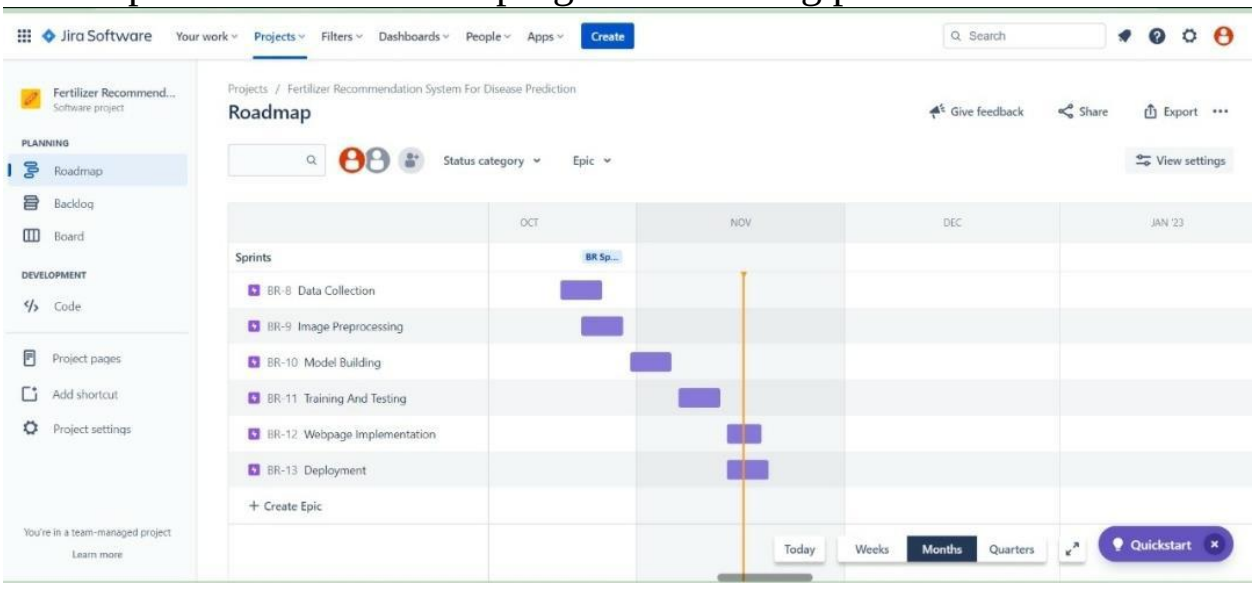
BOARD:

A board reflects your team's process, tracking the status of work. The columns on the board represent the status of your team's issues. The visual representation of the work helps in discussing and tracking of the progress of the project from start to finish.



ROADMAP:

A roadmap offers quick and easy planning that helps teams better manage their dependencies and track progress on the big picture in real-time.



CODING & SOLUTIONING

Python – app.py:

```
import os
import numpy as np
import pandas as pd
from tensorflow.keras.models import load_model
# from tensorflow.keras.preprocessing import image
from werkzeug.utils import secure_filename
from flask import Flask, render_template, request
app = Flask(__name__)
#load both the vegetable and fruit models
model = load_model("vegetable.h5")
model1=load_model("fruit.h5")
#home@app.route('/')

def home():
return render_template('home.html')
#prediction page
@app.route('/prediction')
def prediction():
```

```
return render_template('predict.html')
@app.route('/predict',methods=['POST'])
def predict():
if request.method == 'POST':
# Get the file from post request
f = request.files['image']
# Save the file to ./uploads
basepath = os.path.dirname(__file__)
file_path = os.path.join(
basepath, 'uploads', secure_filename(f.filename))
f.save(file_path)
img = image.load_img(file_path, target_size=(128, 128))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
plant=request.form['plant']
print(plant)
if(plant=="vegetable"):
preds = model.predict(x)
preds=np.argmax(preds)
print(preds)
df=pd.read_excel('precautions - veg.xlsx')
print(df.iloc[preds]['caution'])
else:
preds = model1.predict(x)
preds=np.argmax(preds)
df=pd.read_excel('precautions - fruits.xlsx')
print(df.iloc[preds]['caution'])
return df.iloc[preds]['caution']
if __name__ == "__main__":
app.run(debug=False)
s
```

Feature 1:
home.html:

```
<!DOCTYPE html>
<html >
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title> Plant Disease Prediction</title>
<link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
<link href='https://fonts.googleapis.com/css?family=Merriweather'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat' rel='stylesheet'>
<script type="text/javascript" src="https://gc.kis.v2.scr.kaspersky-
labs.com/FD126C42-EBFA-4E12-B309-
BB3FDD723AC1/main.js?attr=AMFGethlf4Q6r2IdpTrTqcDQGNLDU5Cbc3diYnUdLkg5mQrVB_td22OH
UAsBJSd0oo8OR0zM3rIPeFWfnEY4XCxQu4KOxMSqlshEoIBOzvYw0SsMYpyUv4fnvKEjmJoj_Y6cI4ov-
6AMOkz3Sh3epkfq0gltfnAPvvQBRdXqRmdqePVjlvvqL28ONZCiS0Qr5t0XGxJ0bSiWVT-
rH3cqaKCk05eP1Dx04mieTcjsA_TtFLx15PUu0ed6soaj-FOO6-
1d4OQxbJYBXUBefiUhzmOYCpsGIs1OyQvA0huo8AUYwYB72dvs07U3O2hq8BmYBv98h13sSo8iXKxyKx4F
UsOMkixjxYP6hu0wwi7yv1E2rei3GHtPl5YwHkWioQIPqAmrlmaPtFZmF-
jE4_UUCi9IEKws8IduDiqQIFkxfO3YT_sUC9gWmxKSpGbiebwCgV-
wvdGEnbUxY18p9Db6jC6FVKRhqdMBianq63qv-
zZRMZbEpjzQT0DQAH3Yho4o4A00FIW2004q8Q80xt2kV928P_nBgS9HOgHI5EZxenbjfqANTs1rh8GGhBd
7RJaE8-
2AaqT6zbLf2tILJ8j4fk3bV1qsdw0fPmp6foJbDu4343XH36a0VGHsMLeVqcc30PSsE1pJbGE4_C_ExQd0
_uRSA40mRjnFwHdLo9SJc1qghyc5YGQil_utG48olMy9cC6z-iyKg1EeLKB43u-
q4SlUimRnuUsZW7drNWaijSfJPDmkm7lUJ0POwQXPfnLa2_spc3FisWCOZ7dFuIgDciIu0yF8rio2X0Pz6
pZkGQW4Fwl6vWKrLplmHagJElKXg58YSWwAT2DILilBjuSPiTwCHR9Ya_mAXW4C03v7xzJlaSK9jneECqc
tvKnH3RFgDS8ocfDcY65lXNRkq6v1hrcdv5sM2ek4Kjq4OFgX-wijr-0JdpSDpZlbIK00sPb4-
u1B8c7MaCqBcbJAhfmg4utLU67fn5GLoCX_-5TAWV0ID-_sC1Vs9glWRPkKmmktJMbVy98XqC5-
DhtE3yd5I9ZM1SEH1gGYLlRjxwzPjWwHE-YH1Nx9lm-
Esq27TK7M86uT8iAe7LgtviO2YsCB0buShHWmjh3RzwMGqNqeymFSxPRK_sDmTFoVjcaYpGa0kaMwhmmF
9AtPwGmFaGglv3rryVg0X0bGoXRetnrPpDG7jUoq5zQuXQSedBf9hmNwEqWsSZtI4zNTxjiEkxU0djhPXq
ByZbnelp_3z6pqqniLzqj9jzAkvX6wDOW7ZycfDzOt-
zNgTxWdtf41P6ZjVu8EWSf65Wqgen5jD4IPXgXGtxkjrSbrqiX-
NxxxfKVJUOoOcEO0F6n3DWD0BMWS8UGOQO8gZZeXCfpuTIGYTD6okyD91kLk5AmhaNTJVKjkHO-
dHZqMHxikVhdK6C2PIfg4lEY0yuE3Fjj_5NNX5ZalIpOl3LN6YQ8Jqis_UmC_OXmjW2F5Y4p8VRRKc1HW2
DFaUxBrEgfSwe_keyaofodrjde_pfPuDQDryEgGy9DNIhpGUV_bQJ8jlPxRL7WSpmPU7-
IZ1mVN_onhqq2oI-WTl7ep-8w0GsJH3OhSRyyJC0XC9xtetqVjIHzcbKYFsxOaXT-
LLe7U9oHaXHzjDK3hn-ZNFYwzV_aoq8180eb" charset="UTF-8"></script><style>
.header {
```

```
top:0;
margin:0px;
left: 0px;
right: 0px;
position: fixed;
background-color: #28272c;
color: white;
box-shadow: 0px 8px 4px grey;
overflow: hidden;
padding-left:20px;
font-family: 'Josefin Sans';
font-size: 2vw;
width: 100%;
height:8%;
text-align: center;
}
.topnav {
overflow: hidden;
background-color: #333;
}
.topnav-right a {
float: left;
color: #f2f2f2;
text-align: center;
padding: 14px 16px;
text-decoration: none;
font-size: 18px;
}
.topnav-right a:hover {
background-color: #ddd;
color: black;
}
.topnav-right a.active {
background-color: #565961;
color: white;
}
.topnav-right {
float: right;
padding-right:100px;
}
body {
background-color:#ffffff;
background-repeat: no-repeat;
background-size:cover;
background-position: 0px 0px;
}
.button {
background-color: #28272c;
border: none;
color: white;
padding: 15px 32px;
text-align: center;
text-decoration: none;
```

```
display: inline-block;
font-size: 16px;
border-radius: 12px;
}
.button:hover {
box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}
input[type=text], input[type=password] {
width: 100%;
padding: 12px 20px;
display: inline-block;
margin-bottom:18px;
border: 1px solid #ccc;
box-sizing: border-box;
}
button {
background-color: #28272c;
color: white;
padding: 14px 20px;
```

```
margin-bottom:8px;
border: none;
cursor: pointer;
width: 15%;
border-radius:4px;
}
```

```
button:hover {
opacity: 0.8;
}
.cancelbtn {
width: auto;
padding: 10px 18px;
background-color: #f44336;
}
.imgcontainer {
text-align: center;
margin: 24px 0 12px 0;
}
img.avatar {
width: 30%;
border-radius: 50%;
}
.container {
padding: 16px;
}
span.psw {
float: right;
padding-top: 16px;
}
/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
span.psw {
display: block;
float: none;
}
.cancelbtn {
width: 100%;
}
}
.home{
margin:80px;
width: 84%;
height: 500px;
padding-top:10px;
padding-left: 30px;
}
.login{
margin:80px;
box-sizing: content-box;
width: 84%;
height: 420px;
padding: 30px;
border: 10px solid blue;
}
.left,.right{
box-sizing: content-box;
height: 400px;
margin:20px;
border: 10px solid blue;
```

```
}
.mySlides {display: none;}
img {vertical-align: middle;}
/* Slideshow container */
.slideshow-container {
max-width: 1000px;
position: relative;
margin: auto;
}
/* Caption text */
.text {
color: #f2f2f2;
font-size: 15px;
padding: 8px 12px;
position: absolute;
bottom: 8px;
width: 100%;
text-align: center;
}
/* The dots/bullets/indicators */
.dot {
height: 15px;
width: 15px;
margin: 0 2px;
background-color: #bbb;
border-radius: 50%;
display: inline-block;
transition: background-color 0.6s ease;
}
.active {
background-color: #717171;
}
/* Fading animation */
.fade {
-webkit-animation-name: fade;
-webkit-animation-duration: 1.5s;
animation-name: fade;
animation-duration: 1.5s;
}
@-webkit-keyframes fade {
from {opacity: .4}
to {opacity: 1}
}
@keyframes fade {
from {opacity: .4}
to {opacity: 1}
}
/* On smaller screens, decrease text size */
@media only screen and (max-width: 300px) {
.text {font-size: 11px}
}
</style>
</head>
```

```
<body style="font-family:'Times New Roman', Times, serif;background-
color:#C2C5A8;">
<div class="header">
<div style="width:50%;float:left;font-size:2vw;text-align:left;color:white;
padding-top:1%">Plant Disease Prediction</div>
<div class="topnav-right"style="padding-top:0.5%;">
<a class="active" href="{{ url_for('home')}}">Home</a>
<a href="{{ url_for('prediction')}}">Predict</a>
</div>
</div>
<div style="background-color:#ffffff;">
<div style="width:60%;float:left;">
<div style="font-size:50px;font-family:Montserrat;padding-left:20px;text-
align:center;padding-top:10%;">
<b>Detect if your plant<br> is infected!!</b></div><br>
<div style="font-size:20px;font-family:Montserrat;padding-left:70px;padding-
right:30px;text-align:justify;">Agriculture is one of the major sectors worls
wide. Over the years it has developed and the use of new technologies and
equipment replaced almost all the traditional methods of farming. The plant
diseases effect the production. Identification of diseases and taking necessary
precautions is all done through naked eye, which requires labour and laboratries.
This application helps farmers in detecting the diseases by observing the spots on
the leaves, which inturn saves effort and labor costs.</div><br><br>
</div>
</div>
<div style="width:40%;float:right;"><br><br>
<img src="{{url_for('static',filename='images/12456.png')}}" style="max-
height:100%;max-width:100%;">
</div>
</div>
<div class="home">
<br>
</div>
<script>
var slideIndex = 0;
showSlides();
function showSlides() {
var i;
var slides = document.getElementsByClassName("mySlides");
var dots = document.getElementsByClassName("dot");
for (i = 0; i < slides.length; i++) {
slides[i].style.display = "none";
}
slideIndex++;
if (slideIndex > slides.length) {slideIndex = 1}
for (i = 0; i < dots.length; i++) {
dots[i].className = dots[i].className.replace(" active", "");
}
slides[slideIndex-1].style.display = "block";
dots[slideIndex-1].className += " active";
setTimeout(showSlides, 2000); // Change image every 2 seconds
}
```

```
</script>
</body>
</html>
```

Feature 2 :
Predict.html:

```
<!DOCTYPE html>
<html >
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title> Plant Disease Prediction</title>
<link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>
<link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css"
rel="stylesheet">
<script type="text/javascript" src="https://gc.kis.v2.scr.kaspersky-
labs.com/FD126C42-EBFA-4E12-B309-
BB3FDD723AC1/main.js?attr=3wvf44XdejigWHFj22ANQmgfA-L5oa67wZhZwPtEITSot6t8o-
DPZwNcHRFhpa2tgGpDJGis4-1IHYyxyIAN2GE0-kSZKkCLRkbKttCLVN9mKhGFVtGJ3auoiiByn_jJ-
mA447x4TmdjGgz8XvMdLSPF4Gu5xwt0joGxWDXuOEF18Sa5usZGgj4TdDiTfDHpElX3P1eH-
lsevFhUJQEZe3981VXjRKYRn2FrxsYwXGSMBn0sRR9IYup35XYNQkvA6DLQV1lwLc4XuAo0BlJYAfI75R4
O5LwTWuT-uaft0DEQeuV_f3rKvkrcBkalcpWnyXVLeLyjMz5CqpZ1aSCy1MgVAzWxGb-
GX3eQb0F5qOksANddV_vhz1Ai4RgptuAfB8mVyuz0nWZzpmwam34lc4NL4tfyWGncKz2taMyGfsK4Mrn0z
fPlY9_n9FP0lMlAX0IQ8TfbVp4B1vbwnA-
RVJq8mxoTjgMgqhKhp6NQY_8gZULkbqqA0pqUMvfL3_fZC1PFipLNjCyCGe9YOaU9L7QF4CXeKsRhJXmI8
98FhpxB1oI7z0xvndsDLPRsqbNuse_eGL9tz0Te5HLGhtoXSn5O8pHC99_XHYofrlismcByzZlmVqVkCNf
mbnMjaD9IQf6xAACyjkQ927AOvyDVCZKr-
tV6wRZyv_z7Z1J9AG7SGSLoB34AkMytkYXvpgGn21pGFNhvl3YSmyKYc2XJs89zHbp5fSyXsfasogSEYLb
pxCmuvzZKO4haaqouKDcLwBGMFp_Br095f-
AlhhWOdPDx1ezvTMx1NgS4QO97OmbyQCqHUFWWZLYNgjQ8zpfdBXB17L_v_lfmrUWhUiUVc9tRcJy-
lpchFJe8Gz7TUOKCRDjbIWtiqXryDeENrJgQ31laXp-
VVYpOI1L55pek2fgk5OCGNzVges5oG4PpMyCIXtJpv32E5rlPTktG4hD8eXmYQECVU1HvSmEiKvuY6T6i9
wdpqg_AnycRzUXmYdahFT3W7zToIn2RXzNfdOU0zbYBvtJ70TpR4PjfU75lJ0FsnphDuCnero3UYOak7vY
vGYD9YV2md5v-3AmP-eOor2m55JZRH_Hxpn28x-nDNCOHqVBC6leYuYFBVV_vL5l-
E8n92uWUqwMEzdZPZtAyRaCfz3D2Y0IYn-
ZrnfNTg2M_zVJePmUu1xdjYh7d1dx7nwclm7wJrBPb3JnX2kvEGYs9SM17MlwzoY1VJq4UzJ2D6oEvhQwH
vG4e1etlS6iLWzhy8RVMfBlTa4DPDOHmTlHhsKbn0UaMyFFCppe79rtVRctcomnVmQysUwUOhjzlAq30-
hXJCTqdCWJe2xnxjAuUHVqHSiHiZllZaoOWNCV5Ypx_eqzn-KyZS3u-
2_hGLHHNA2AVBWn_hF3Gz16dw6zA4QSmWZSfDUcNObLJGOSTaDS3Z8jPTloYPFmu8oES6TL1dLlEK5YhcS
GaX4iv6o95drsZGb6bBcWgT7sNFHW6dVE9wdjoDFuBergPIAm0sKaZQ2Ex6j1OWCbE6UaPg-
VNfziA2FEPpJaI9hEPI2gdaSuHqovlEOt5mjuFBBOxpK0t8kOZRtsVzqUuJw3VcLjaP6SfG_KZfgX_g8TP
s6CcFhlLRz63oXMQFPW6AA7eudWfygndazedq5B-
6DqSkOT04GTUJNqLcElg6KEEWqxd88BzoQoK28jrAf-xWHNIZv5HmQQYEnyX0U_cW8HX-
```

```
hde54TuY_fY3e5QYu4be-JxTkA4JxWLEagSa7-zs" charset="UTF-8"></script><script
src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
<script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
<script
src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Merriweather'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat' rel='stylesheet'>
<link href="{{ url_for('static', filename='css/final.css') }}" rel="stylesheet">
<style>
.header {
top:0;
margin:0px;
left: 0px;
right: 0px;
position: fixed;
background-color: #28272c;
color: white;
box-shadow: 0px 8px 4px grey;
overflow: hidden;
padding-left:20px;
font-family: 'Josefin Sans';
```

```
font-size: 2vw;
width: 100%;
height:8%;
text-align: center;
}
.topnav {
overflow: hidden;
background-color: #333;
}
```

```
.topnav-right a {
float: left;
color: #f2f2f2;
text-align: center;
padding: 14px 16px;
text-decoration: none;
font-size: 18px;
}
.topnav-right a:hover {
background-color: #ddd;
color: black;
}
.topnav-right a.active {
background-color: #565961;
color: white;
}
.topnav-right {
float: right;
padding-right:100px;
}
.login{
margin-top:-70px;
}
body {
background-color:#ffffff;
background-repeat: no-repeat;
background-size:cover;
background-position: 0px 0px;
}
.login{ margin-top:100px;
}
.container {
margin-top:40px;
padding: 16px;
}
select {
width: 100%;
margin-bottom: 10px;
background: rgba(255,255,255,255);
border: none;
outline: none;
padding: 10px;
font-size: 13px;
color: #000000;
text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
border: 1px solid rgba(0,0,0,0.3);
border-radius: 4px;
box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px
rgba(255,255,255,0.2);
-webkit-transition: box-shadow .5s ease;
-moz-transition: box-shadow .5s ease;
-o-transition: box-shadow .5s ease;
-ms-transition: box-shadow .5s ease;
```

```
transition: box-shadow .5s ease;
}
</style>
</head>
<body style="font-family:Montserrat;overflow:scroll;">
<div class="header">
<div style="width:50%;float:left;font-size:2vw;text-align:left;color:white;
padding-top:1%">Plant Disease Prediction</div>
<div class="topnav-right" style="padding-top:0.5%;">
</div>
</div>
<div class="container">
<div id="content" style="margin-top:2em">
<div class="container">
<div class="row">
<div class="col-sm-6 bd" >
<br>
<img src="{{url_for('static',filename='images/789.jpg')}}"
style="height:450px;width:550px"class="img-rounded" alt="Gesture">
</div>
<div class="col-sm-6">
<div>
<h4>Drop in the image to get the prediction </h4>
<form action = "" id="upload-file" method="post" enctype="multipart/form-data">
<select name="plant">
<option value="select" selected>Select plant type</option>
<option value="fruit">Fruit</option>
<option value="vegetable">Vegetable</option>
</select><br>
<label for="imageUpload" class="upload-label" style="background: #28272c;">
Choose...
</label>
<input type="file" name="image" id="imageUpload" accept=".png, .jpg, .jpeg">
</form>
<div class="image-section" style="display:none;">
<div class="img-preview">
<div id="imagePreview">
</div>
</div>
<div>
<button type="button" class="btn btn-info btn-lg " id="btn-predict"
style="background: #28272c;">Predict!</button>
</div>
</div>
<div class="loader" style="display:none;"></div>
<h3>
<span id="result" style="font-size:17px; "> </span>
</h3>
</div>
```

```
</div>
</div>
</div>
</div>
</div>
</body>
<footer>
<script src="{{ url_for('static', filename='js/main.js') }}"
type="text/javascript"></script>
</footer>
</html>
```

final.css:

```
.img-preview {
width: 256px;
height: 256px;
position: relative;
border: 5px solid #F8F8F8;
box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
margin-top: 1em;
margin-bottom: 1em;
}
.img-preview>div {
width: 100%;
height: 100%;
background-size: 256px 256px;
background-repeat: no-repeat;
background-position: center;
}
input[type="file"] {
display: none;
}
.upload-label{
display: inline-block;
padding: 12px 30px;
background: #28272c;
color: #fff;
font-size: 1em;
transition: all .4s;
cursor: pointer;
}
.upload-label:hover{
background: #C2C5A8;
color: #39D2B4;
}
.loader {
border: 8px solid #f3f3f3; /* Light grey */
border-top: 8px solid #28272c; /* Blue */
border-radius: 50%;
width: 50px;
height: 50px;
```

```
animation: spin 1s linear infinite;
}
@keyframes spin {
0% { transform: rotate(0deg); }
100% { transform: rotate(360deg); }
}
```

main.js:

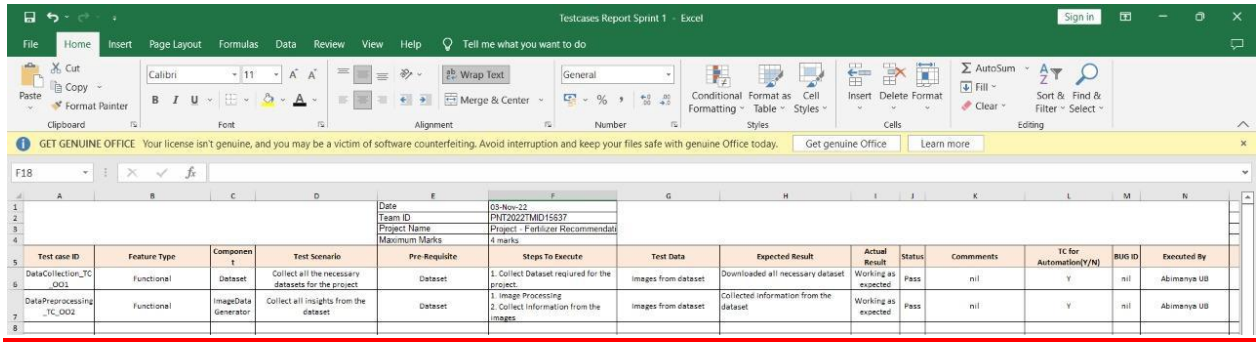
```
$(document).ready(function () {
// Init
$('.image-section').hide();
$('.loader').hide();
$('#result').hide();
// Upload Preview
function readURL(input) {
if (input.files && input.files[0]) {
var reader = new FileReader();
reader.onload = function (e) {
$('#imagePreview').css('background-image', 'url(' + e.target.result + ')');
$('#imagePreview').hide();
$('#imagePreview').fadeIn(650);
}
reader.readAsDataURL(input.files[0]);
}
}
$("#imageUpload").change(function () {
$('.image-section').show();
$('#btn-predict').show();
$('#result').text('');
$('#result').hide();
readURL(this);
});
// Predict
$('#btn-predict').click(function () {
var form_data = new FormData($('#upload-file')[0]);
// Show loading animation
$(this).hide();
$('.loader').show();
// Make prediction by calling api /predict
$.ajax({
type: 'POST',
url: '/predict',
data: form_data,
contentType: false,
cache: false,
processData: false,
async: true,
success: function (data) {
// Get and display the result
$('.loader').hide();
$('#result').fadeIn(600);
```

```
$('#result').text('Prediction: '+data);
console.log('Success!');
},
});
});
});
```

TESTING

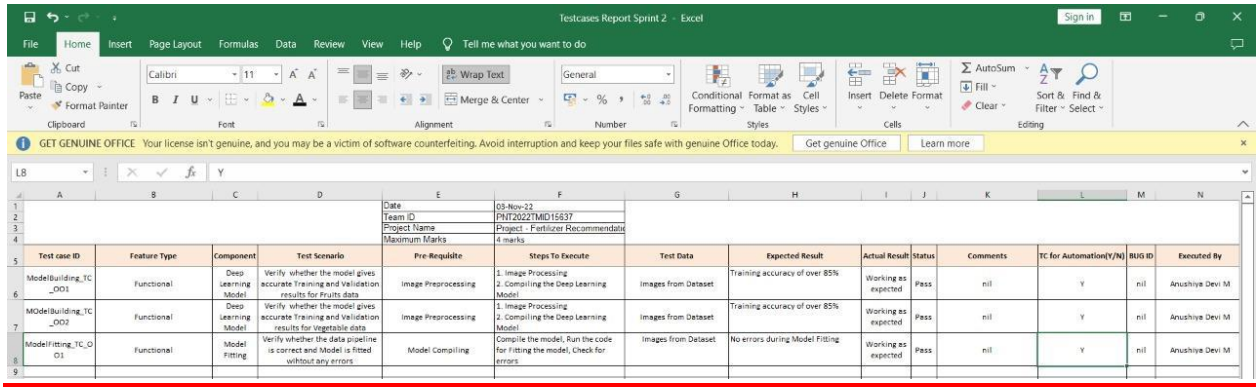
TEST CASES

Test cases are a set of actions performed on a system to determine if it satisfies software requirements and functions correctly as it claimed to perform.



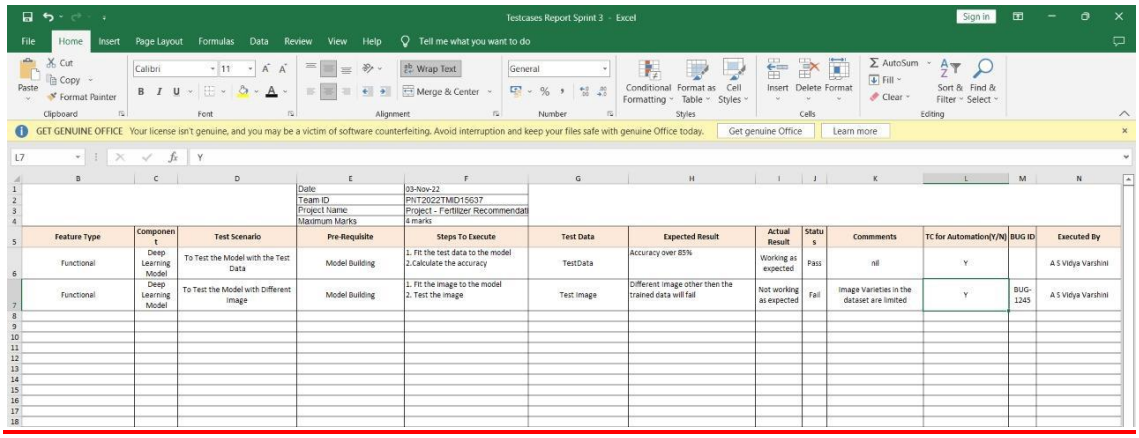
Date	03-Nov-22
Team ID	PNT2022TMID15637
Project Name	Project - Fertilizer Recommendati
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commments	TC for Automation(Y/N)	BUG ID	Executed By
DataCollection_TC_OO1	Functional	Dataset	Collect all the necessary datasets for the project	Dataset	1. Collect Dataset required for the project.	Images from dataset	Downloaded all necessary dataset	Working as expected	Pass	nil	Y	nil	Abimanya UB
DataPreprocessing_TC_OO2	Functional	ImageData Generator	Collect all insights from the dataset	Dataset	1. Image Processing 2. Collect Information from the images	Images from dataset	Collected information from the dataset	Working as expected	Pass	nil	Y	nil	Abimanya UB



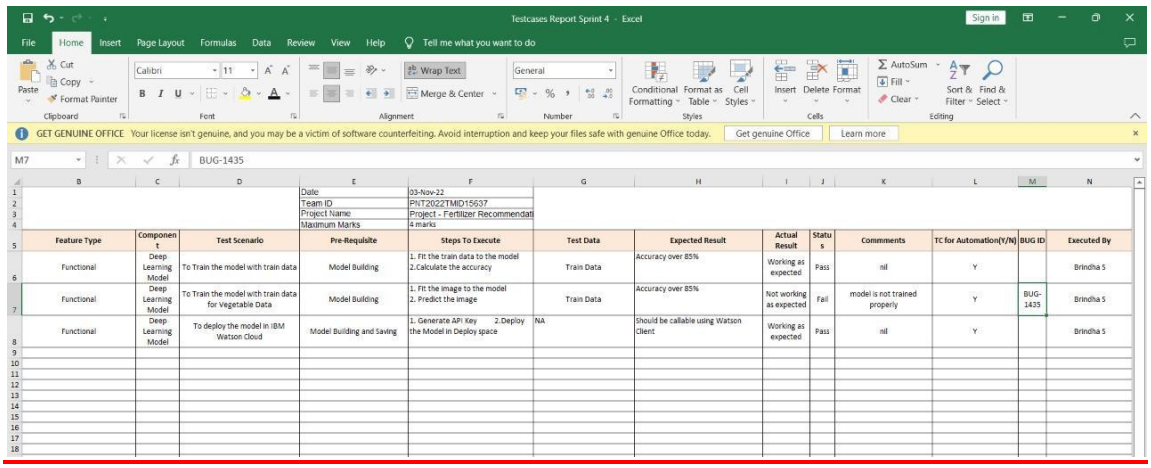
Date	03-Nov-22
Team ID	PNT2022TMID15637
Project Name	Project - Fertilizer Recommendati
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
ModelBuilding_TC_OO1	Functional	Deep Learning Model	Verify whether the model gives accurate Training and Validation results for Fruits data	Image Preprocessing	1. Image Processing 2. Compiling the Deep Learning Model	Images from Dataset	Training accuracy of over 85%	Working as expected	Pass	nil	Y	nil	Anushiya Devi M
MOdelBuilding_TC_OO2	Functional	Deep Learning Model	Verify whether the model gives accurate Training and Validation results for Vegetable data	Image Preprocessing	1. Image Processing 2. Compiling the Deep Learning Model	Images from Dataset	Training accuracy of over 85%	Working as expected	Pass	nil	Y	nil	Anushiya Devi M
ModelFitting_TC_OO1	Functional	Model Fitting	Verify whether the data pipeline is correct and Model is fitted without any errors	Model Compiling	Compile the model, Run the code for Fitting the model, Check for errors	Images from Dataset	No errors during Model Fitting	Working as expected	Pass	nil	Y	nil	Anushiya Devi M



Date	03-Nov-22
Team ID	PNT2022TMID15637
Project Name	Project - Fertilizer Recommendati
Maximum Marks	4 marks

Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commments	TC for Automation(Y/N)	BUG ID	Executed By
Functional	Deep Learning Model	To Test the Model with the Test Data	Model Building	1. Fit the test data to the model 2.Calculate the accuracy	TestData	Accuracy over 85%	Working as expected	Pass	nil	Y		A S Vidya Varshini
Functional	Deep Learning Model	To Test the Model with Different Image	Model Building	1. Fit the image to the model 2. Test the image	Test Image	Different image other then the trained data will fail	Not working as expected	Fail	Image Varieties in the dataset are limited	Y	BUG-1243	A S Vidya Varshini



Date	03-Nov-22
Team ID	PNT2022TMID15637
Project Name	Project - Fertilizer Recommendati
Maximum Marks	4 marks

Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commments	TC for Automation(Y/N)	BUG ID	Executed By
Functional	Deep Learning Model	To Train the model with train data	Model Building	1. Fit the train data to the model 2.Calculate the accuracy	Train Data	Accuracy over 85%	Working as expected	Pass	nil	Y		Brindha S
Functional	Deep Learning Model	To Train the model with train data for Vegetable Data	Model Building	1. Fit the image to the model 2. Predict the image	Train Data	Accuracy over 85%	Not working as expected	Fail	model is not trained properly	Y	BUG-1435	Brindha S
Functional	Deep Learning Model	To deploy the model in IBM Watson Cloud	Model Building and Saving	1. Generate API Key 2.Deploy the Model in Deploy space	NA	Should be callable using Watson Client	Working as expected	Pass	nil	Y		Brindha S

USER ACCEPTANCE TESTING

Before deploying the software application to a production environment the end user or client performs a type of testing known as user acceptance testing, or UAT to ensure whether the software functionalities serve the purpose of development.

Acceptance Testing
UAT Execution & Report Submission

Date	17 November 2022
Team ID	PNT2022TMID15637
Project Name	Fertilisers recommendation System for disease prediction
Maximum Marks	4 Marks

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Fertilizers recommendation system for disease prediction project at the time of the release to User Acceptance Testing (UAT).

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Fertilizers recommendation system for disease prediction project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
Leaf spots	10	4	2	3	19
Mosaic leaf pattern	9	6	3	6	24
Blights	4	5	2	1	12
Yellow leaves	11	4	3	20	38
Fruit rots	3	2	1	0	6
Misshapen leaves	2	7	0	1	10
Fruit spots	5	4	1	1	11
Totals	44	31	13	32	120

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Leaf spots	18	0	0	18
Fruit spots	5	0	0	5
Mosaic leaf pattern	43	0	0	43
Blights	2	0	0	2
Misshapen leaves	25	0	0	25
Yellow leaves	7	0	0	7
Fruit rots	9	0	0	9

RESULTS

Performance Metrics

Perform metrics are a baseline for performance tests. Monitoring the correct parameters will help you detect areas that require increased attention and find ways

to improve them. Final findings (output) of the project given below in the form of screenshot: Training and Testing of Fruit dataset

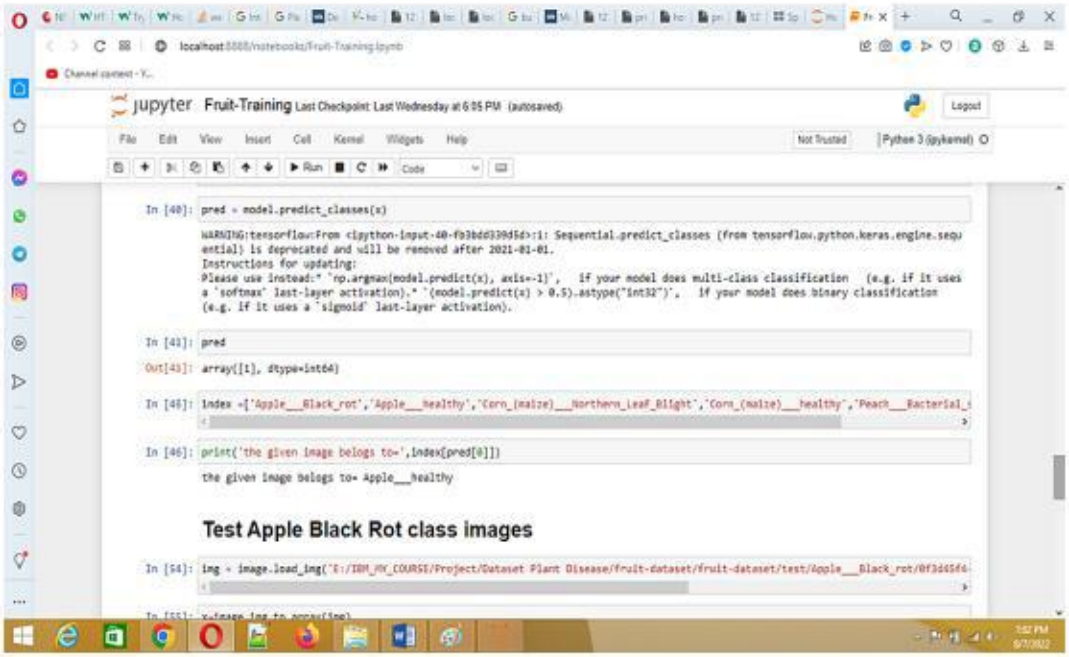


Figure.6.2 Test the Fruit dataset

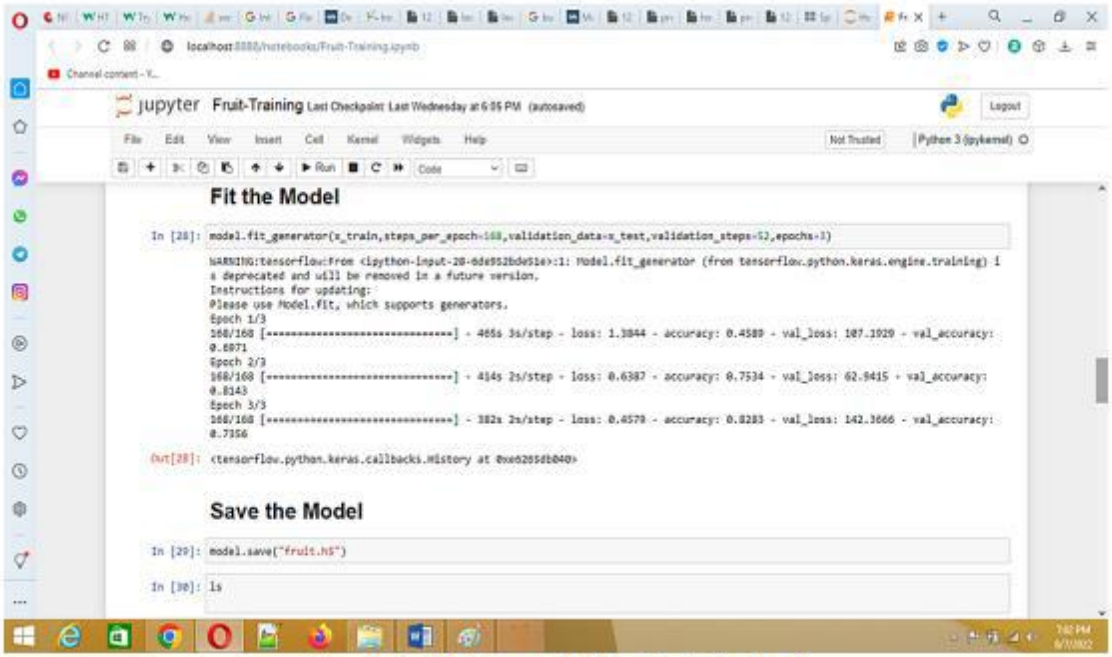


Figure.6.1. Fit a model for Fruit dataset

Train and Test Vegetable dataset

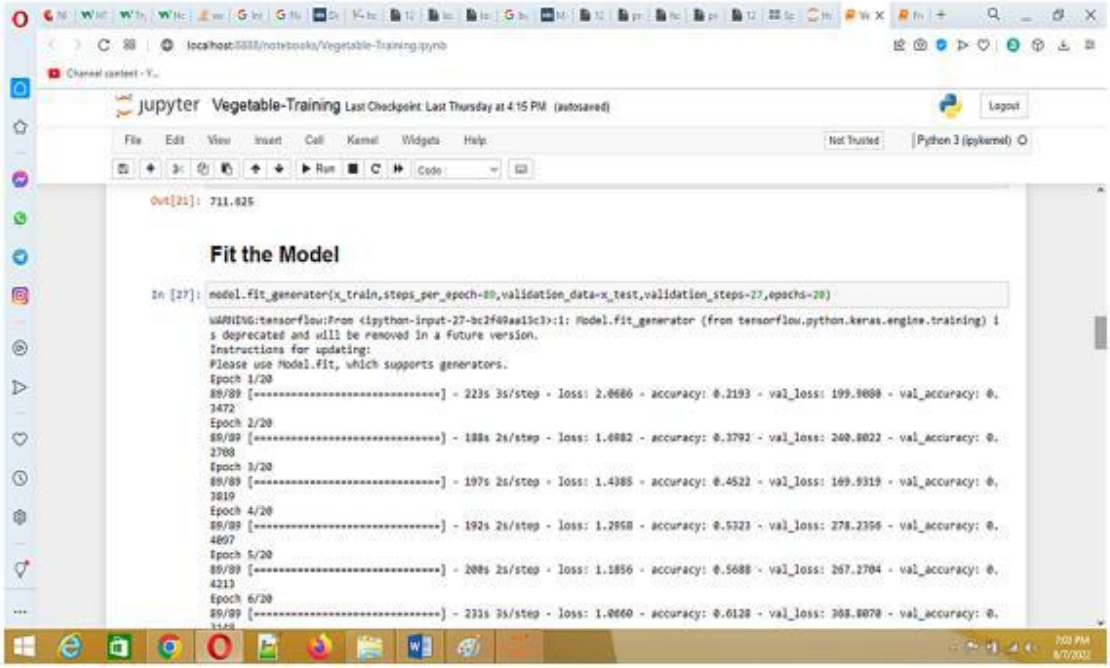


Figure.6.3. Train the Vegetable dataset

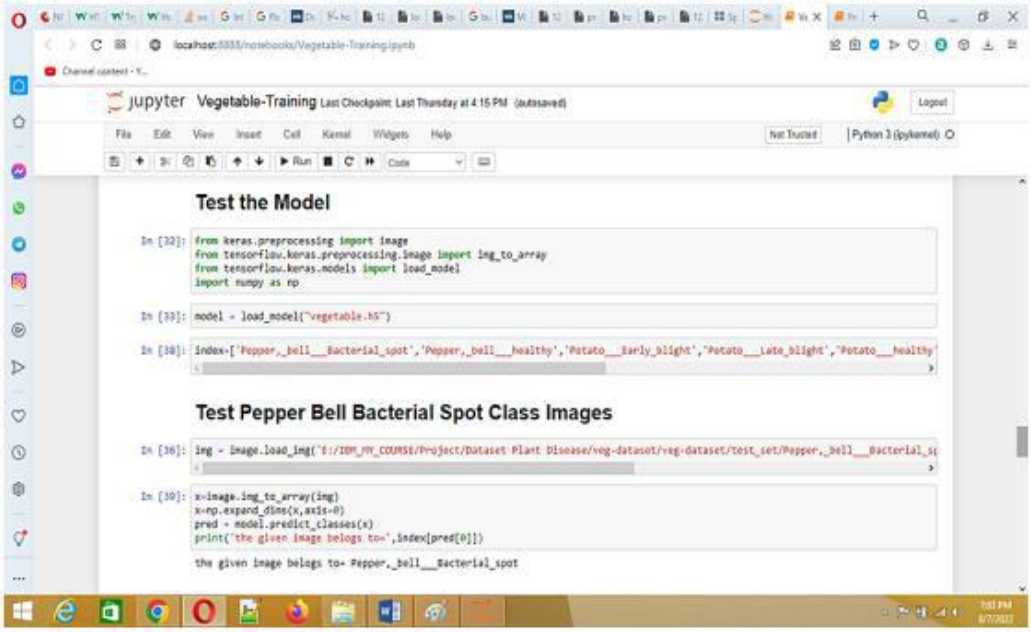
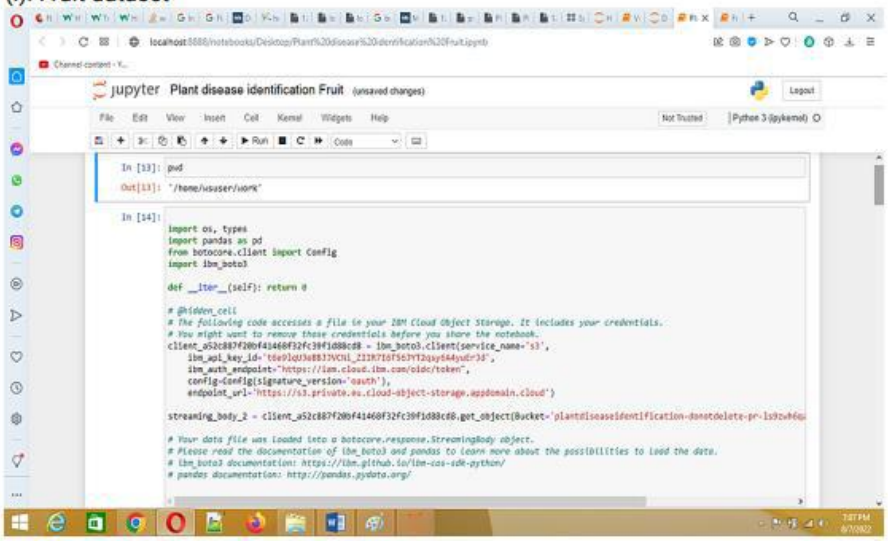


Figure.6.4. Test the Vegetable dataset

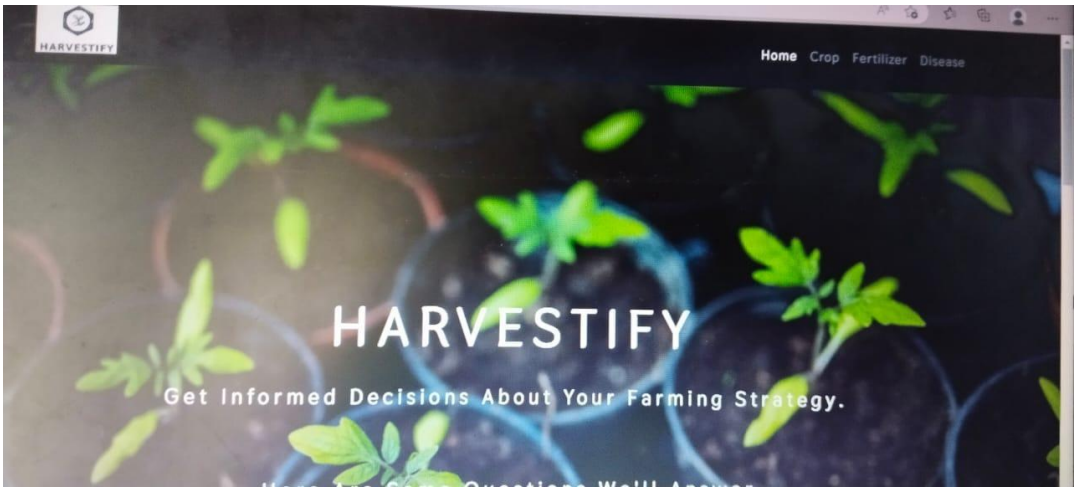
Train and Test Vegetable dataset IBM Cloud

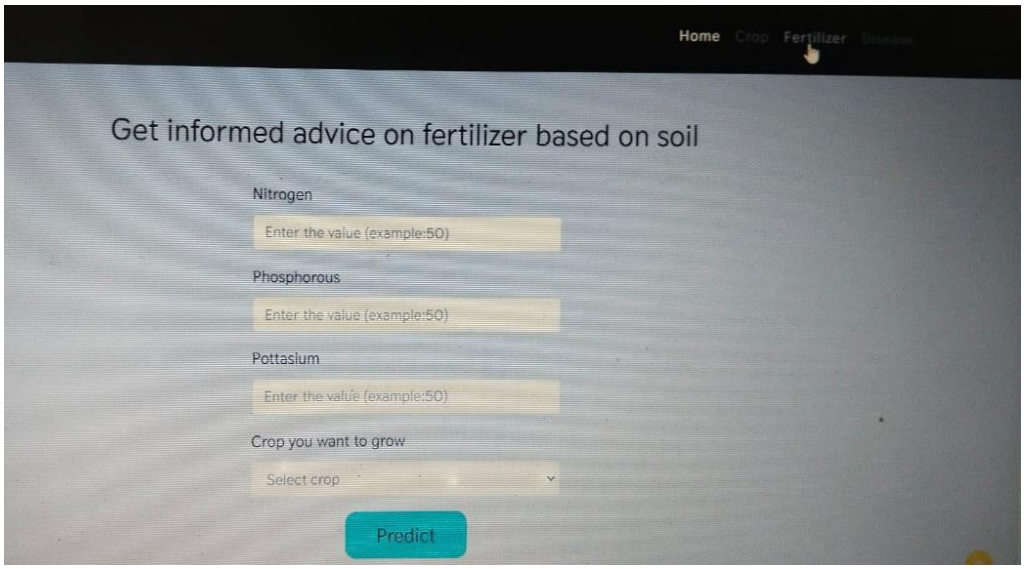
Due to CUH limit exceeds, I have downloaded the notebooks and opened in Jupyter notebook

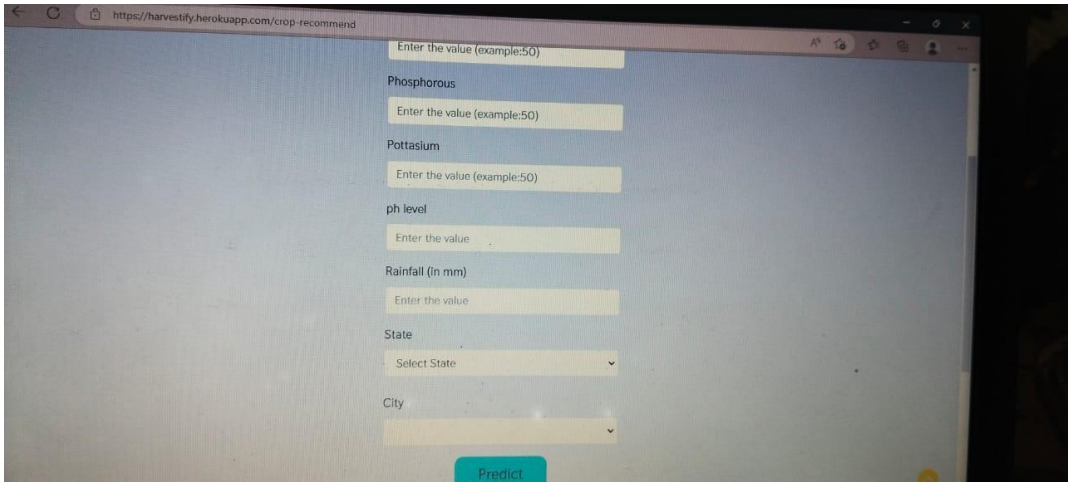
(i). Fruit dataset

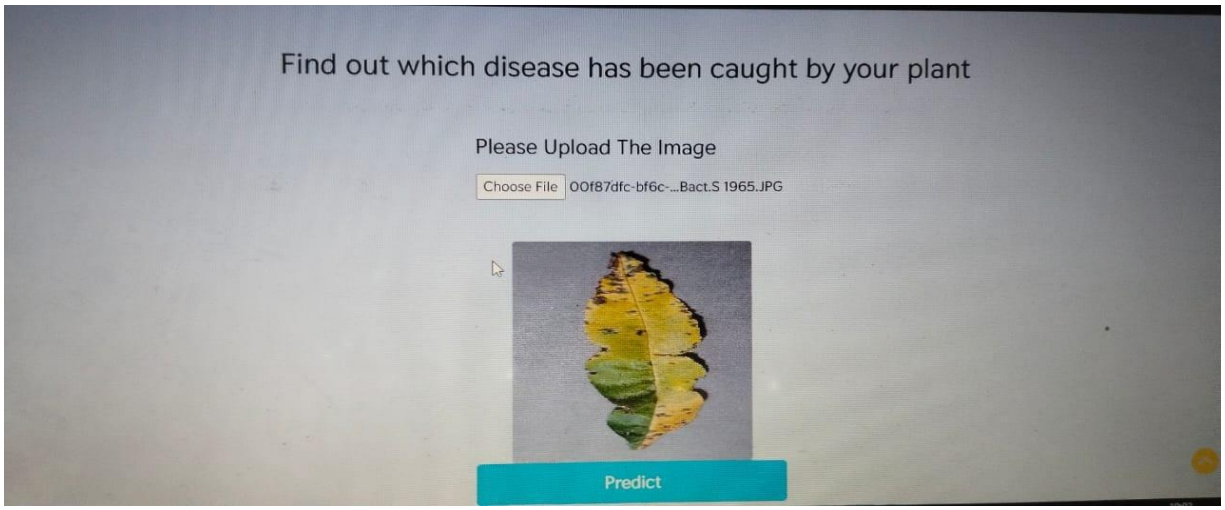


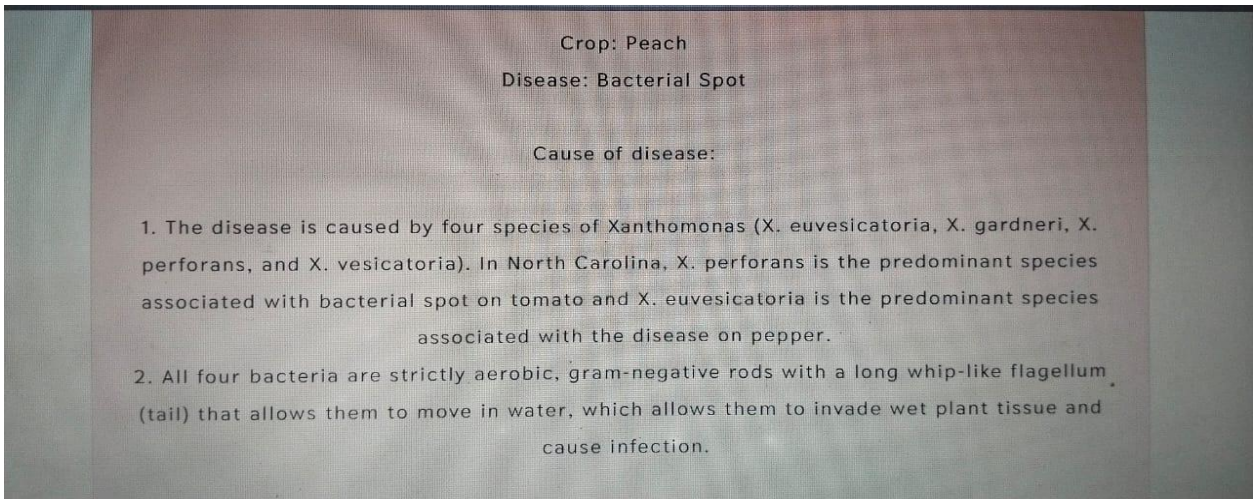
OUTPUT

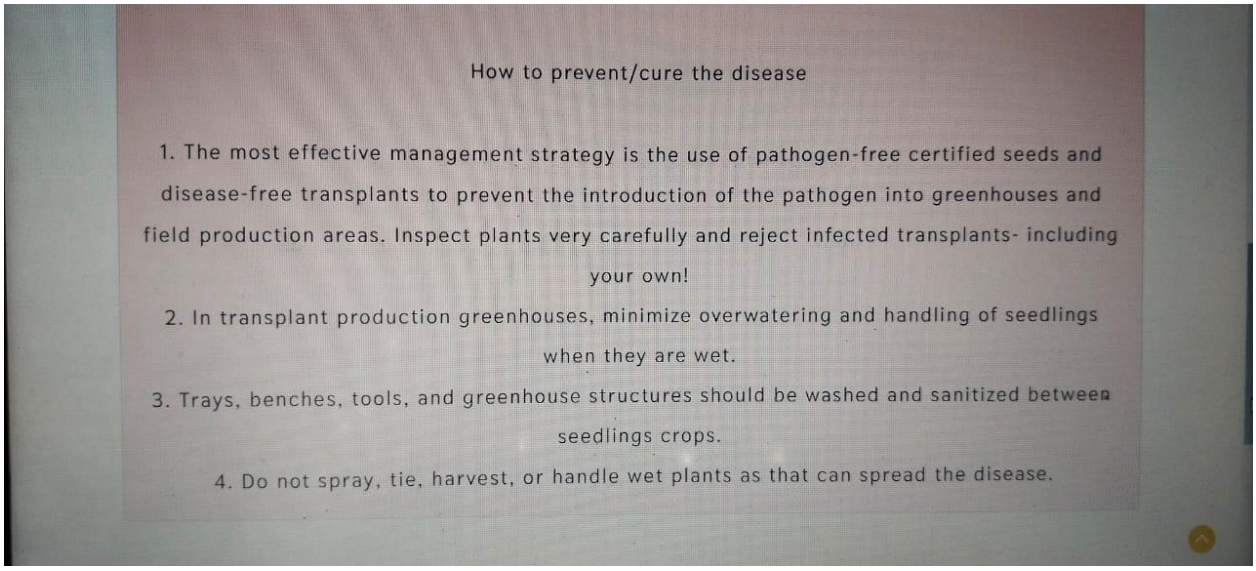



Home
Crop
Fertilizer
Disease
Get informed advice on fertilizer based on soil
Nitrogen
Enter the value (example:50)
Phosphorous
Enter the value (example:50)
Pottasium
Enter the value (example:50)
Crop you want to grow
Select crop
Predict


https://harvestify.herokuapp.com/crop-recommend
Enter the value (example:50)
Phosphorous
Enter the value (example:50)
Pottasium
Enter the value (example:50)
ph level
Enter the value
Rainfall (in mm)
Enter the value
State
Select State
City
Predict


Find out which disease has been caught by your plant
Please Upload The Image
Choose File
00f87dfc-bf6c-...Bact.S 1965.JPG
Predict





ADVANTAGES & DISADVANTAGES

LIST OF ADVANTAGES

- The proposed model here produces very high accuracy of classification.
- Very large datasets can also be trained and tested.
- Images of very high can be resized within the proposed itself.

LIST OF DISADVANTAGES

- For training and testing, the proposed model requires very high computational time.
- The neural network architecture used in this project work has high complexity.

APPLICATIONS

1. The trained network model used to classify the image patterns with high accuracy.

2. The proposed model not only used for plant disease classification but also for other image pattern classification such as animal classification.

3. This project work application involves not only image classification but also for pattern recognition.

CONCLUSION

The model proposed here involves image classification of fruit datasets and vegetable datasets. The following points are observed during model testing and training: The accuracy of classification increased by increasing the number of epochs. For different batch sizes, different classification accuracies are obtained The accuracies are increased by increasing more convolution layers. The accuracy of classification also increased by varying dense layers.Different accuracies are obtained by varying the size of kernel used in the convolution layer output. Accuracies are different while varying the size of the train and test datasets. Detection and recognition of plant diseases using machine learning are very efficient in providing symptoms of identifying diseases at its earliest. Plant pathologists can analyze 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 of the regions of agriculture.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.agriculture plays a vital role in our daily life providing us the basic essential that is the food , It is important to ensure that the farmers who practice agriculture should be benefited and gain the profits in their field without any loss or emotional breakout.

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.

APPENDIX

A. Source Code (Jupyter notebook python code)

fruit.ipynb (due to limited page size the code vegetable.ipynb uploaded in github)

```
#!/usr/bin/env python
# coding: utf-8
# In[1]: pwd
# In[2]: cd E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruit-dataset/fruit-dataset
# # Apply ImageDataGenerator functionality to Train and Test set
# # Preprocessing # In[3]: from keras.preprocessing.image import ImageDataGenerator train_datagen =
ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_fli p=True) test_datagen =
ImageDataGenerator(rescale=1) # In[4]: pwd
# In[5]: x_train = train_datagen.flow_from_directory('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruit-
dataset/fruitdataset/train',target_size=(128,128),batch_size=32,class_mode='categorical')
# In[6]: x_test=test_datagen.flow_from_directory('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruit-
dataset/fruit-dataset/test',target_size=(128,128), batch_size=32,class_mode='categorical') # # Import the models
# In[7]: from tensorflow.keras.models import Sequential from tensorflow.keras.layers import
Dense,Convolution2D,MaxPool2D,Flatten
# # Initializing the models 10
# In[8]: model=Sequential()
# # Add CNN Layers
# In[9]: model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
# In[10]: x_train.class_indices
# # Add Pooling layer # In[11]: model.add(MaxPool2D(pool_size=(2,2)))
# # Add Flatten layer # In[12]: model.add(Flatten())
# # Add Dense Layer
# In[21]: model.add(Dense(40, kernel_initializer='uniform',activation='relu')) model.add(Dense(20,
kernel_initializer='random_uniform',activation='relu'))
# # Add Output Layer # In[24]: model.add(Dense(6,activation='softmax', kernel_initializer='random_uniform'))
# # Compile the model # In[25]:
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy' ]) # In[26]: len(x_train)
# In[27]: 5384/32
# # Fit the Model
# In[28]: model.fit_generator(x_train,steps_per_epoch=168,validation_data=x_test,validation_st eps=52,epochs=3)
# # Save the Model
# In[29]: model.save("fruit.h5")
# In[30]: ls
# # Test the Model
# In[32]: from keras.preprocessing import image from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model import numpy as np
# In[33]: model = load_model("fruit.h5")
# # Test Apple_Healthy Class images
# In[37]: img = image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruitdataset/fruit-
dataset/test/Apple___healthy/00fca0da-2db3-481b-b98a9b67bb7b105c___RS_HL 7708.JPG',target_size=(128,128))
11
# In[39]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0)
```

In[40]: pred = model.predict_classes(x)
In[41]: pred
In[45]: index =['Apple___Black_rot','Apple___healthy','Corn_(maize)___Northern_Leaf_Blight','Corn_(maize)___healthy','Peach___Bacterial_spot','Peach___healthy']
In[46]: print('the given image belogs to=',index[pred[0]])
Test Apple Black Rot class images # In[54]: img = image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruitdataset/fruit-dataset/test/Apple___Black_rot/0f3d45f4-e121-42cd-a5b6-be2f866a0574___JR_FrgE.S 2870.JPG',target_size=(128,128))
In[55]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred = model.predict_classes(x) print('the given image belogs to=',index[pred[0]])
Test Corn Northern leaf Blight class images
In[56]: img = image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruitdataset/fruitdataset/test/Corn_(maize)___Northern_Leaf_Blight/00a14441-7a62- 4034-bc40-b196aeab2785___RS_NLB 3932.JPG',target_size=(128,128))
In[57]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred = model.predict_classes(x) print('the given image belogs to=',index[pred[0]])
Test Corn Healthy class images # In[58]: img = image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruitdataset/fruit-dataset/test/Corn_(maize)___healthy/0a68ef5a-027c-41ae-b227-159dae77d3dd___R.S_HL 7969 copy.jpg',target_size=(128,128))
In[59]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred = model.predict_classes(x) print('the given image belogs to=',index[pred[0]]) # # Test Peach Bacterial spot class images # In[60]: img = image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruitdataset/fruit-dataset/test/Peach___Bacterial_spot/00ddc106-692e-4c67-b2e8- 569c924caf49___Rutg._Bact.S 1228.JPG',target_size=(128,128)) 12 # In[61]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred = model.predict_classes(x) print('the given image belogs to=',index[pred[0]])
Test Peach Healthy class images
In[62]: img = image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruitdataset/fruit-dataset/test/Peach___healthy/1a07ce54-f4fd-41cf-b088- 144f6bf71859___Rutg._HL 3543.JPG',target_size=(128,128))
In[63]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred = model.predict_classes(x) print('the given image belogs to=',index[pred[0]])

DEPLOYMENT MODEL CODE:

Fruit model:

ls
sample_data/
pwd
'/home/wsuser/work'
!pip install keras==2.7.0
!pip install tensorflow==2.5.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab wheels/public/simple/
Requirement already satisfied: keras==2.7.0 in /usr/local/lib/python3.7/dist-packages (2.7.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab wheels/public/simple/
Requirement already satisfied: tensorflow==2.5.0 in /usr/local/lib/python3.7/dist-packages (2.5.0)
Requirement already satisfied: h5py~=3.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (3.1.0)
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (3.19.6)
Requirement already satisfied: typing-extensions~=3.7.4 in /usr/local/lib/python3.7/dist packages (from tensorflow==2.5.0) (3.7.4.3)
Requirement already satisfied: keras-nightly~=2.5.0.dev in /usr/local/lib/python3.7/dist packages (from tensorflow==2.5.0) (2.5.0.dev2021032900)

Requirement already satisfied: flatbuffers~=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (1.12)
Requirement already satisfied: gast==0.4.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (0.4.0)
Requirement already satisfied: absl-py~=0.10 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (0.15.0)
Requirement already satisfied: astunparse~=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (1.6.3)
Requirement already satisfied: tensorflow-estimator<2.6.0,>=2.5.0rc0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (2.5.0) Requirement already satisfied: tensorboard~=2.5 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (2.9.1)
Requirement already satisfied: opt-einsum~=3.3.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (3.3.0)
Requirement already satisfied: six~=1.15.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (1.15.0)
Requirement already satisfied: google-pasta~=0.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (0.2.0)
Requirement already satisfied: grpcio~=1.34.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (1.34.1)
Requirement already satisfied: wrapt~=1.12.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (1.12.1)
Requirement already satisfied: termcolor~=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (1.1.0)
Requirement already satisfied: keras-preprocessing~=1.1.2 in /usr/local/lib/python3.7/dist packages (from tensorflow==2.5.0) (1.1.2)
Requirement already satisfied: wheel~=0.35 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (0.38.3)
Requirement already satisfied: numpy~=1.19.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (1.19.5)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages (from h5py~=3.1.0->tensorflow==2.5.0) (1.5.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist packages (from tensorboard~=2.5->tensorflow==2.5.0) (2.14.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow==2.5.0) (0.6.1) Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow==2.5.0) (1.8.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow==2.5.0) (0.4.6)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow==2.5.0) (1.0.1)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow==2.5.0) (3.4.1)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist packages (from tensorboard~=2.5->tensorflow==2.5.0) (2.23.0)
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow==2.5.0) (57.4.0)

Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboard~=2.5->tensorflow==2.5.0) (4.9) Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist packages (from google-auth<3,>=1.6.3->tensorboard~=2.5->tensorflow==2.5.0) (0.2.8) Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.7/dist packages (from google-auth<3,>=1.6.3->tensorboard~=2.5->tensorflow==2.5.0) (5.2.0) Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.5->tensorflow==2.5.0) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist packages (from markdown>=2.6.8->tensorboard~=2.5->tensorflow==2.5.0) (4.13.0) Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard~=2.5->tensorflow==2.5.0) (3.10.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard~=2.5- >tensorflow==2.5.0) (0.4.8)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.5->tensorflow==2.5.0) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.5->tensorflow==2.5.0) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.5->tensorflow==2.5.0) (3.0.4) Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.5->tensorflow==2.5.0) (2022.9.24) Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.5- >tensorflow==2.5.0) (3.2.2)

Image Augmentation

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True,v
ertical_flip=False)
test_datagen=ImageDataGenerator(rescale=1./255)
ls
pwd
/content
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3
def __iter__(self): return 0
# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your crede ntials.
# You might want to remove those credentials before you share the notebook.
client_4ff9f1114db24196a9abd4f5c1f0b60a = ibm_boto3.client(service_name='s3',
ibm_api_key_id='j4lNXssktSSxQiDx3pbNR_eFi1SMCDE6MFnBQ_EmNCDM',
ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
config=Config(signature_version='oauth'),
```

endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')
streaming_body_1 = client_4ff9f1114db24196a9abd4f5c1f0b60a.get_object(Bucket='trainm odel-donotdelete-pr-cbqe37eh8gzesa', Key='fruit-dataset.zip')['Body']
# Your data file was loaded into a botocore.response.StreamingBody object. # Please read the documentation of ibm_boto3 and pandas to learn more about the possibil ities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/ # pandas documentation: http://pandas.pydata.org/
from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_1.read()), "r")
file_paths = unzip.namelist()
for path in file_paths:
unzip.extract(path)
pwd
'/home/wsuser/work'
import os
filenames = os.listdir('/home/wsuser/work/fruit-dataset/train')
x_train=train_datagen.flow_from_directory("/home/wsuser/work/fruit dataset/train",target_size=(128,128),class_mode='categorical',batch_size=24) Found 5384 images belonging to 6 classes.
x_test=test_datagen.flow_from_directory(r"/home/wsuser/work/fruit dataset/test",target_size=(128,128),
class_mode='categorical',batch_size=24)
Found 1686 images belonging to 6 classes.
x_train.class_indices
{'Apple___Black_rot': 0, 'Apple___healthy': 1, 'Corn_(maize)___Northern_Leaf_Blight': 2, 'Corn_(maize)___healthy': 3, 'Peach___Bacterial_spot': 4, 'Peach___healthy': 5}

CNN

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Convolution2D,MaxPooling2D,Flatten
model=Sequential()
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.summary()
Model: "sequential_1"
_____ Layer (type)
Output Shape Param #
==========
conv2d_1 (Conv2D) (None, 126, 126, 32) 896
max_pooling2d (MaxPooling2D (None, 63, 63, 32) 0
)
flatten (Flatten) (None, 127008) 0
========== Total
params: 896
Trainable params: 896
Non-trainable params: 0
32*(3*3*3+1)

896
#Hidden Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))

Output Layers

model.add(Dense(6,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy']) len(x_train)
225
1238/24
51.583333333333336
model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validatio
n_steps=len(x_test),epochs=10)
/tmp/wsuser/ipykernel_164/1582812018.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation
_steps=len(x_test),epochs=10)
Epoch 1/10
225/225 [==============================] - 118s 520ms/step - loss: 0.8920 - accuracy:
0.8094 - val_loss: 0.2273 - val_accuracy: 0.9235
Epoch 2/10
225/225 [==============================] - 116s 515ms/step - loss: 0.2367 - accuracy:
0.9179 - val_loss: 0.2056 - val_accuracy: 0.9324
Epoch 3/10
225/225 [==============================] - 116s 517ms/step - loss: 0.1970 - accuracy:
0.9337 - val_loss: 0.4972 - val_accuracy: 0.8754
Epoch 4/10
225/225 [==============================] - 117s 521ms/step - loss: 0.1688 - accuracy:
0.9422 - val_loss: 0.2279 - val_accuracy: 0.9217
Epoch 5/10
225/225 [==============================] - 116s 516ms/step - loss: 0.1438 - accuracy:
0.9487 - val_loss: 0.1685 - val_accuracy: 0.9484
Epoch 6/10
225/225 [==============================] - 117s 518ms/step - loss: 0.1362 - accuracy:
0.9556 - val_loss: 0.1176 - val_accuracy: 0.9662
Epoch 7/10
225/225 [==============================] - 116s 515ms/step - loss: 0.1282 - accuracy:
0.9590 - val_loss: 0.5466 - val_accuracy: 0.8387
Epoch 8/10
225/225 [==============================] - 116s 514ms/step - loss: 0.1282 - accuracy:
0.9597 - val_loss: 0.1194 - val_accuracy: 0.9620
Epoch 9/10
225/225 [==============================] - 116s 514ms/step - loss: 0.1141 - accuracy:
0.9616 - val_loss: 0.1478 - val_accuracy: 0.9508
Epoch 10/10
225/225 [==============================] - 116s 516ms/step - loss: 0.0927 - accuracy:
0.9695 - val_loss: 0.0772 - val_accuracy: 0.9751

<keras.callbacks.History at 0x7f71e8184070>

Saving Model

ls
fruit-dataset/
model.save('fruit.h5')
!tar -zcvf Train-model_new.tgz fruit.h5
fruit.h5
ls -1
fruit-dataset/
fruit.h5
Train-model_new.tgz

IBM Cloud Deployment Model

!pip install watson-machine-learning-client –upgrade
Collecting watson-machine-learning-client
Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
|[redacted]| 538 kB 21.2 MB/s eta 0:00:01
Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (4.62.3)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (2022.9.24)

Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (2.26.0)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (0.8.9)
Requirement already satisfied: ibm-cos-sdk in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.11.0) Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (1.3.4)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: boto3 in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (1.18.21)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (1.26.7)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client) (0.10.0) Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client) (0.5.0) Requirement already satisfied: botocore<1.22.0,>=1.21.21 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client) (1.21.41) Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from botocore<1.22.0,>=1.21.21->boto3->watson machine-learning-client) (2.8.2)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from python-dateutil<3.0.0,>=2.1->botocore<1.22.0,>=1.21.21->boto3->watson machine-learning-client) (1.15.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machine-learning-client) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machine-learning-client) (2.11.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from requests->watson-machine-learning-client) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from requests->watson-machine-learning-client) (3.3) Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client) (2021.3) Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client) (1.19.5) Installing collected packages: watson-machine-learning-client
Successfully installed watson-machine-learning-client-1.0.391

```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
"url": "https://us-south.ml.cloud.ibm.com",
"apikey":"0P3XkyCFYqABnc48BNG2ReoGAJy-oDXDRuULl4Y_zFxa"
}
client = APIClient(wml_credentials)
def guid_from_space_name(client, space_name):
```

```
space = client.spaces.get_details()
return(next(item for item in space['resources'] if item['entity']["name"]==space_name)['m
etadata']['id'])
space_uid = guid_from_space_name(client, 'Trainmodel')
print("Space UID = " + space_uid)
Space UID = 616c7d74-e99b-4c09-9922-27394a62c2d0
client.set.default_space(space_uid)
'SUCCESS'
client.software_specifications.list()
NAME ASSET_ID TYPE
default_py3.6 0062b8c9-8b7d-44a0-a9b9-46c416adcbd9 base kernel-spark3.2-scala2.12 020d69ce-
7ac1-5e68-ac1a-31189867356a base pytorch-onnx_1.3-py3.7-edt 069ea134-3346-5748-b513-
49120e15d288 base scikit-learn_0.20-py3.6 09c5a1d0-9c1e-4473-a344-eb7b665ff687 base spark-
mllib_3.0-scala_2.12 09f4cff0-90a7-5899-b9ed-1ef348aebdee base pytorch-onnx_rt22.1-py3.9
0b848dd4-e681-5599-be41-b5f6fccc6471 base ai-function_0.1-py3.6 0cdb0f1e-5376-4f4d-92dd-
da3b69aa9bda base shiny-r3.6 0e6e79df-875e-4f24-8ae9-62dcc2148306 base
tensorflow_2.4-py3.7-horovod 1092590a-307d-563d-9b62-4eb7d64b3f22 base pytorch_1.1-py3.6
10ac12d6-6b30-4ccd-8392-3e922c096a92 base tensorflow_1.15-py3.6-ddl 111e41b3-de2d-5422-
a4d6-bf776828c4b7 base runtime-22.1-py3.9 12b83a17-24d8-5082-900f-0ab31fbfd3cb base scikit-
learn_0.22-py3.6 154010fa-5b3b-4ac1-82af-4d5ee5abbc85 base default_r3.6 1b70aec3-ab34-4b87-
8aa0-a4a3c8296a36 base pytorch-onnx_1.3-py3.6 1bc6029a-cc97-56da-b8e0-39c3880dbbe7 base
kernel-spark3.3-r3.6 1c9e5454-f216-59dd-a20e-474a5cdf5988 base pytorch-onnx_rt22.1-py3.9-edt
1d362186-7ad5-5b59-8b6c-9d0880bde37f base tensorflow_2.1-py3.6 1eb25b84-d6ed-5dde-b6a5-
3fbdf1665666 base spark-mllib_3.2 20047f72-0a98-58c7-9ff5-a77b012eb8f5 base tensorflow_2.4-
py3.8-horovod 217c16f6-178f-56bf-824a-b19f20564c49 base runtime-22.1-py3.9-cuda 26215f05-
08c3-5a41-a1b0-da66306ce658 base do_py3.8 295addb5-9ef9-547e-9bf4-92ae3563e720 base autoai-
ts_3.8-py3.8 2aa0c932-798f-5ae9-abd6-15e0c2402fb5 base tensorflow_1.15-py3.6 2b73a275-7cbf-
420b-a912-eae7f436e0bc base kernel-spark3.3-py3.9 2b7961e2-e3b1-5a8c-a491-482c8368839a base
pytorch_1.2-py3.6 2c8ef57d-2687-4b7d-acce-01f94976dac1 base spark-mllib_2.3 2e51f700-bca0-
4b0d-88dc-5c6791338875 base pytorch-onnx_1.1-py3.6-edt 32983cea-3f32-4400-8965-
dde874a8d67e base spark-mllib_3.0-py37 36507ebe-8770-55ba-ab2a-eafe787600e9 base spark-
mllib_2.4 390d21f8-e58b-4fac-9c55-d7ceda621326 base xgboost_0.82-py3.6 39e31acd-5f30-41dc-
ae44-60233c80306e base pytorch-onnx_1.2-py3.6-edt 40589d0e-7019-4e28-8daa-fb03b6f4fe12 base
default_r36py38 41c247d3-45f8-5a71-b065-8580229facf0 base
autoai-ts_rt22.1-py3.9 4269d26e-07ba-5d40-8f66-2d495b0c71f7 base autoai-obm_3.0 42b92e18-
d9ab-567f-988a-4240ba1ed5f7 base pmml-3.0_4.3 493bcb95-16f1-5bc5-bee8-81b8af80e9c7 base
spark-mllib_2.4-r_3.6 49403dff-92e9-4c87-a3d7-a42d0021c095 base xgboost_0.90-py3.6 4ff8d6c2-
1343-4c18-85e1-689c965304d3 base pytorch-onnx_1.1-py3.6 50f95b2a-bc16-43bb-bc94-
b0bed208c60b base autoai-ts_3.9-py3.8 52c57136-80fa-572e-8728-a5e7cbb42cde base spark-
mllib_2.4-scala_2.11 55a70f99-7320-4be5-9fb9-9edb5a443af5 base spark-mllib_3.0 5c1b0ca2-4977-
5c2e-9439-ffd44ea8ffe9 base autoai-obm_2.0 5c2e37fa-80b8-5e77-840f-d912469614ee base spss-
modeler_18.1 5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b base cuda-py3.8 5d3232bf-c86b-5df4-a2cd-
7bb870a1cd4e base autoai-kb_3.1-py3.7 632d4b22-10aa-5180-88f0-f52dfb6444d7 base pytorch-
onnx_1.7-py3.8 634d3cdc-b562-5bf9-a2d4-
```

ea90a478456b base spark-mllib_2.3-r_3.6 6586b9e3-ccd6-4f92-900f-0f8cb2bd6f0c base tensorflow_2.4-py3.7 65e171d7-72d1-55d9-8ebb-f813d620c9bb base spss-modeler_18.2 687eddc9-028a-4117-b9dd-e57b36f1efa5 base ---------- ---------- ----
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
software_space_uid = client.software_specifications.get_uid_by_name("tensorflow_rt22.1- py3.9")
software_spec_uid
‘1eb25b84-d6ed-5dde-b6a5-3fbdf1665666'
ls
fruit-dataset/ fruit.h5 Train-model_new.tgz
model_details = client.repository.store_model(model= 'Train-model_new.tgz', meta_props={
client.repository.ModelMetaNames.NAME:"CNN",
client.repository.ModelMetaNames.TYPE:"tensorflow_2.7",
client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid})
model_id = client.repository.get_model_id(model_details)
model_id
'd0aeb6a2-e89c-4f8d-bf2f-a28ca4ea3cca'
ls
fruit-dataset/ fruit.h5 Train-model_new.tgz
Test The Model
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model('fruit.h5')
#@title
img=image.load_img(r"C:\Users\LENOVO\Desktop\fruit-dataset\fruit dataset\test\00fca0da-2db3-481b-b98a
9b67bb7b105c___RS_HL 7708.JPG",target_size=(128,128))
img
img=image.load_img(r"C:\Users\LENOVO\Desktop\ibm\Dataset Plant Disease\fruit dataset\fruit-dataset\test\Apple___healthy\0adc1c5b-8958-47c0-a152- f28078c214f1___RS_HL
7825.JPG",target_size=(128,128))
img
x=image.img_to_array(img)
X
array([[[99., 86., 106.],
[101., 88., 108.],
[118., 105., 125.],
...,
[92., 83., 102.],
[93., 84., 103.],
[89., 80., 99.]],
[[96., 83., 103.],
[87., 74., 94.],
[102., 89., 109.],
...,
[88., 79., 98.],
[89., 80., 99.],
[83., 74., 93.]],
[[86., 73., 93.],

```
[ 88., 75., 95.],
[ 98., 85., 105.],
...,
[107., 98., 117.],
[ 96., 87., 106.],
[ 96., 87., 106.]],
...,
[[172., 175., 194.],
[173., 176., 195.],
[175., 178., 197.],
...,
[179., 180., 198.],
[184., 185., 203.],
[179., 180., 198.]],
[[172., 175., 194.],
[170., 173., 192.],
[173., 176., 195.],
...,
[178., 179., 197.],
[182., 183., 201.],
[178., 179., 197.]],
[[169., 172., 191.],
[166., 169., 188.],
[168., 171., 190.],
...,
[187., 188., 206.],
[185., 186., 204.],
[186., 187., 205.]]], dtype=float32) x=np.expand_dims(x,axis=0)
X
array([[[[ 99., 86., 106.],
[101., 88., 108.],
[118., 105., 125.],
...,
[ 92., 83., 102.],
[ 93., 84., 103.],
[ 89., 80., 99.]],
[[ 96., 83., 103.],
[ 87., 74., 94.],
[102., 89., 109.],
...,
[ 88., 79., 98.],
[ 89., 80., 99.],
[ 83., 74., 93.]],
[[ 86., 73., 93.],
[ 88., 75., 95.],
[ 98., 85., 105.],
...,
[107., 98., 117.],
[ 96., 87., 106.],
```

```
[ 96., 87., 106.]],
...,
[[172., 175., 194.],
[173., 176., 195.],
[175., 178., 197.],
...,
[179., 180., 198.],
[184., 185., 203.],
[179., 180., 198.]],
[[172., 175., 194.],
[170., 173., 192.],
[173., 176., 195.],
...,
[178., 179., 197.],
[182., 183., 201.],
[178., 179., 197.]],
[[169., 172., 191.],
[166., 169., 188.],
[168., 171., 190.],
...,
[187., 188., 206.],
[185., 186., 204.],
[186., 187., 205.]]]], dtype=float32)
y=np.argmax(model.predict(x),axis=1)
1/1 [==============================] - 0s 105ms/step
x_train.class_indices
{'Apple___Black_rot': 0, 'Apple___healthy': 1, 'Corn_(maize)___Northern_Leaf_Blight': 2,
'Corn_(maize)___healthy': 3, 'Peach___Bacterial_spot': 4, 'Peach___healthy': 5}
index=['Apple___Black_rot','Apple___healthy','Corn_(maize)___Northern_Leaf_Blight','Corn
_(maize)___healthy','Peach___Bacterial_spot','Peach___healthy']
index[y[0]]
'Apple___healthy'
img=image.load_img(r"C:\LENOVO\Desktop\ibm\Dataset Plant Disease\fruit-dataset\fruit
dataset\test\Peach___healthy\0a2ed402-5d23-4e8d-bc98-
b264aea9c3fb___Rutg._HL 2471.JPG",target_size=(128,128))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
index=['Apple___Black_rot','Apple___healthy''Peach___Bacterial_spot','Peach___healthy']
index[y[0]]
1/1 [==============================] - 0s 26ms/step
'Peach___healthy'
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask,render_template,request
app=Flask(__name__)
model=load_model("fruit.h5")
@app.route('/')
```

```
def index():
return render_template("index.html")
@app.route('/predict',methods=['GET','POST'])
def upload():
if request.method=='POST':
f=request.files['image']
basepath=os.path.dirname('__file__')
filepath=os.path.join(basepath,'uploads',f.filename)
f.save(filepath)
img=image.load_img(filepath,target_size=(128,128))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=np.argmax(model.predict(x),axis=1)
index=['Apple___Black_rot','Apple___healthy',
,'Peach___Bacterial_spot','Peach___healthy']
text="The Classified Fruit disease is : " +str(index[pred[0]])
return text
if __name__=='__main__':
app.run(debug=False)
```

vegetable model :

```
ls
sample_data/
pwd
'/home/wsuser/work'
!pip install keras==2.7.0
!pip install tensorflow==2.5.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab wheels/public/simple/
Requirement already satisfied: keras==2.7.0 in /usr/local/lib/python3.7/dist-packages (2.7.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab wheels/public/simple/
Requirement already satisfied: tensorflow==2.5.0 in /usr/local/lib/python3.7/dist-packages (2.5.0)
Requirement already satisfied: h5py~=3.1.0 in /usr/local/lib/python3.7/dist-packages (from
tensorflow==2.5.0) (3.1.0)
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/dist-packages (from
tensorflow==2.5.0) (3.19.6)
Requirement already satisfied: typing-extensions~=3.7.4 in /usr/local/lib/python3.7/dist packages
(from tensorflow==2.5.0) (3.7.4.3)
Requirement already satisfied: keras-nightly~=2.5.0.dev in /usr/local/lib/python3.7/dist packages
(from tensorflow==2.5.0) (2.5.0.dev2021032900)
Requirement already satisfied: flatbuffers~=1.12.0 in /usr/local/lib/python3.7/dist-packages (from
tensorflow==2.5.0) (1.12)
Requirement already satisfied: gast==0.4.0 in /usr/local/lib/python3.7/dist-packages (from
tensorflow==2.5.0) (0.4.0)
Requirement already satisfied: absl-py~=0.10 in /usr/local/lib/python3.7/dist-packages (from
tensorflow==2.5.0) (0.15.0)
Requirement already satisfied: astunparse~=1.6.3 in /usr/local/lib/python3.7/dist-packages (from
tensorflow==2.5.0) (1.6.3)
```

Requirement already satisfied: tensorflow-estimator<2.6.0,>=2.5.0rc0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (2.5.0) Requirement already satisfied: tensorboard~=2.5 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (2.9.1)
Requirement already satisfied: opt-einsum~=3.3.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (3.3.0)
Requirement already satisfied: six~=1.15.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (1.15.0)
Requirement already satisfied: google-pasta~=0.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (0.2.0)
Requirement already satisfied: grpcio~=1.34.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (1.34.1)
Requirement already satisfied: wrapt~=1.12.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (1.12.1)
Requirement already satisfied: termcolor~=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (1.1.0)
Requirement already satisfied: keras-preprocessing~=1.1.2 in /usr/local/lib/python3.7/dist packages (from tensorflow==2.5.0) (1.1.2)
Requirement already satisfied: wheel~=0.35 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (0.38.3)
Requirement already satisfied: numpy~=1.19.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.5.0) (1.19.5)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages (from h5py~=3.1.0->tensorflow==2.5.0) (1.5.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist packages (from tensorboard~=2.5->tensorflow==2.5.0) (2.14.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow==2.5.0) (0.6.1) Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow==2.5.0) (1.8.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow==2.5.0) (0.4.6)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow==2.5.0) (1.0.1)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow==2.5.0) (3.4.1)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist packages (from tensorboard~=2.5->tensorflow==2.5.0) (2.23.0)
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow==2.5.0) (57.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboard~=2.5->tensorflow==2.5.0) (4.9) Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist packages (from google-auth<3,>=1.6.3->tensorboard~=2.5->tensorflow==2.5.0) (0.2.8) Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.7/dist packages (from google-auth<3,>=1.6.3->tensorboard~=2.5->tensorflow==2.5.0) (5.2.0) Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.5->tensorflow==2.5.0) (1.3.1)

Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist packages (from markdown>=2.6.8->tensorboard~=2.5->tensorflow==2.5.0) (4.13.0) Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard~=2.5->tensorflow==2.5.0) (3.10.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard~=2.5- >tensorflow==2.5.0) (0.4.8)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.5->tensorflow==2.5.0) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.5->tensorflow==2.5.0) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.5->tensorflow==2.5.0) (3.0.4) Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.5->tensorflow==2.5.0) (2022.9.24) Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.5- >tensorflow==2.5.0) (3.2.2)

Image Augmentation

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True,v
ertical_flip=False)
test_datagen=ImageDataGenerator(rescale=1./255)
ls
pwd
/content
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3
def __iter__(self): return 0
# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your crede ntials.
# You might want to remove those credentials before you share the notebook.
client_4ff9f1114db24196a9abd4f5c1f0b60a = ibm_boto3.client(service_name='s3',
ibm_api_key_id='j4lNXssktSSxQiDx3pbNR_eFi1SMCDE6MFnBQ_EmNCDM',
ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
config=Config(signature_version='oauth'),
endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')
streaming_body_1 = client_4ff9f1114db24196a9abd4f5c1f0b60a.get_object(Bucket='trainm odel-
donotdelete-pr-cbqe37eh8gzesa', Key='vegetable-dataset.zip')['Body']
# Your data file was loaded into a botocore.response.StreamingBody object. # Please read the
documentation of ibm_boto3 and pandas to learn more about the possibil ities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/ # pandas documentation:
http://pandas.pydata.org/
from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_1.read()), "r")
```

```
file_paths = unzip.namelist()
for path in file_paths:
unzip.extract(path)
pwd
'/home/wsuser/work'
import os
filenames = os.listdir('/home/wsuser/work/vegetable-dataset/train')
x_train=train_datagen.flow_from_directory("/home/wsuser/work/vegetable
dataset/train",target_size=(128,128),class_mode='categorical',batch_size=24) Found 5384 images
belonging to 6 classes.
x_test=test_datagen.flow_from_directory(r"/home/wsuser/work/vegetable
dataset/test",target_size=(128,128),
class_mode='categorical',batch_size=24)
Found 1686 images belonging to 6 classes.
x_train.class_indices
{'Tomato___Blight': 0, 'Tomato___healthy': 1, 'Corn_(maize)___Northern_Leaf_Blight': 2,
'Corn_(maize)___healthy': 3, 'Potato___Blight': 4, 'Potato___healthy': 5}
```

CNN

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Convolution2D,MaxPooling2D,Flatten
model=Sequential()
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.summary()
Model: "sequential_1"
_____ Layer (type)
Output Shape Param #
==================================================================
conv2d_1 (Conv2D) (None, 126, 126, 32) 896
max_pooling2d (MaxPooling2D (None, 63, 63, 32) 0
)
flatten (Flatten) (None, 127008) 0
================================================================= Total
params: 896
Trainable params: 896
Non-trainable params: 0
32*(3*3*3+1)
896
#Hidden Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
```

Output Layers

```
model.add(Dense(6,activation='softmax'))
```

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy']) len(x_train)
225
1238/24
51.583333333333336
model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validatio
n_steps=len(x_test),epochs=10)
/tmp/wsuser/ipykernel_164/1582812018.py:1: UserWarning: `Model.fit_generator` is deprecated and
will be removed in a future version. Please use `Model.fit`, which supports generators.
model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation
_steps=len(x_test),epochs=10)
Epoch 1/10
225/225 [==============================] - 118s 520ms/step - loss: 0.8920 - accuracy:
0.8094 - val_loss: 0.2273 - val_accuracy: 0.9235
Epoch 2/10
225/225 [==============================] - 116s 515ms/step - loss: 0.2367 - accuracy:
0.9179 - val_loss: 0.2056 - val_accuracy: 0.9324
Epoch 3/10
225/225 [==============================] - 116s 517ms/step - loss: 0.1970 - accuracy:
0.9337 - val_loss: 0.4972 - val_accuracy: 0.8754
Epoch 4/10
225/225 [==============================] - 117s 521ms/step - loss: 0.1688 - accuracy:
0.9422 - val_loss: 0.2279 - val_accuracy: 0.9217
Epoch 5/10
225/225 [==============================] - 116s 516ms/step - loss: 0.1438 - accuracy:
0.9487 - val_loss: 0.1685 - val_accuracy: 0.9484
Epoch 6/10
225/225 [==============================] - 117s 518ms/step - loss: 0.1362 - accuracy:
0.9556 - val_loss: 0.1176 - val_accuracy: 0.9662
Epoch 7/10
225/225 [==============================] - 116s 515ms/step - loss: 0.1282 - accuracy:
0.9590 - val_loss: 0.5466 - val_accuracy: 0.8387
Epoch 8/10
225/225 [==============================] - 116s 514ms/step - loss: 0.1282 - accuracy:
0.9597 - val_loss: 0.1194 - val_accuracy: 0.9620
Epoch 9/10
225/225 [==============================] - 116s 514ms/step - loss: 0.1141 - accuracy:
0.9616 - val_loss: 0.1478 - val_accuracy: 0.9508
Epoch 10/10
225/225 [==============================] - 116s 516ms/step - loss: 0.0927 - accuracy:
0.9695 - val_loss: 0.0772 - val_accuracy: 0.9751
<keras.callbacks.History at 0x7f71e8184070>
```

Saving Model

```
ls
vegetable-dataset/
model.save('vegetable.h5')
!tar -zcvf Train-model_new.tgz vegetable.h5
```

vegetable.h5ls -1
vegetable-dataset/
vegetable.h5
Train-model_new.tgz

IBM Cloud Deployment Model

!pip install watson-machine-learning-client –upgrade
Collecting watson-machine-learning-client
Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
|[redacted]| 538 kB 21.2 MB/s eta 0:00:01
Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (4.62.3)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (2022.9.24)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (2.26.0)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (0.8.9)
Requirement already satisfied: ibm-cos-sdk in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.11.0) Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (1.3.4)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: boto3 in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (1.18.21)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from watson-machine-learning-client) (1.26.7)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client) (0.10.0) Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client) (0.5.0) Requirement already satisfied: botocore<1.22.0,>=1.21.21 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client) (1.21.41) Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from botocore<1.22.0,>=1.21.21->boto3->watson machine-learning-client) (2.8.2)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site packages (from python-dateutil<3.0.0,>=2.1->botocore<1.22.0,>=1.21.21->boto3->watson machine-learning-client) (1.15.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machine-learning-client) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machine-learning-client) (2.11.0)

Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from requests->watson-machine-learning-client) (2.0.4) Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from requests->watson-machine-learning-client) (3.3) Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client) (2021.3) Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python 3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client) (1.19.5) Installing collected packages: watson-machine-learning-client
Successfully installed watson-machine-learning-client-1.0.391

```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
"url": "https://us-south.ml.cloud.ibm.com",
"apikey":"0P3XkyCFYqABnc48BNG2ReoGAJy-oDXDRuULl4Y_zFxa"
}
client = APIClient(wml_credentials)
def guid_from_space_name(client, space_name):
space = client.spaces.get_details()
return(next(item for item in space['resources'] if item['entity']["name"]==space_name)['m
etadata']['id'])
space_uid = guid_from_space_name(client, 'Trainmodel')
print("Space UID = " + space_uid)
```

Space UID = 616c7d74-e99b-4c09-9922-27394a62c2d0

```
client.set.default_space(space_uid)
```

'SUCCESS'

```
client.software_specifications.list()
```

NAME ASSET_ID TYPE
default_py3.6 0062b8c9-8b7d-44a0-a9b9-46c416adcbd9 base kernel-spark3.2-scala2.12 020d69ce-7ac1-5e68-ac1a-31189867356a base pytorch-onnx_1.3-py3.7-edt 069ea134-3346-5748-b513-49120e15d288 base scikit-learn_0.20-py3.6 09c5a1d0-9c1e-4473-a344-eb7b665ff687 base spark-mllib_3.0-scala_2.12 09f4cff0-90a7-5899-b9ed-1ef348aebdee base pytorch-onnx_rt22.1-py3.9 0b848dd4-e681-5599-be41-b5f6fccc6471 base ai-function_0.1-py3.6 0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda base shiny-r3.6 0e6e79df-875e-4f24-8ae9-62dcc2148306 base tensorflow_2.4-py3.7-horovod 1092590a-307d-563d-9b62-4eb7d64b3f22 base pytorch_1.1-py3.6 10ac12d6-6b30-4ccd-8392-3e922c096a92 base tensorflow_1.15-py3.6-ddl 111e41b3-de2d-5422-a4d6-bf776828c4b7 base runtime-22.1-py3.9 12b83a17-24d8-5082-900f-0ab31fbfd3cb base scikit-learn_0.22-py3.6 154010fa-5b3b-4ac1-82af-4d5ee5abbc85 base default_r3.6 1b70aec3-ab34-4b87-8aa0-a4a3c8296a36 base pytorch-onnx_1.3-py3.6 1bc6029a-cc97-56da-b8e0-39c3880dbbe7 base kernel-spark3.3-r3.6 1c9e5454-f216-59dd-a20e-474a5cdf5988 base pytorch-onnx_rt22.1-py3.9-edt 1d362186-7ad5-5b59-8b6c-9d0880bde37f base tensorflow_2.1-py3.6 1eb25b84-d6ed-5dde-b6a5-3fbdf1665666 base spark-mllib_3.2 20047f72-0a98-58c7-9ff5-a77b012eb8f5 base tensorflow_2.4-py3.8-horovod 217c16f6-178f-56bf-824a-b19f20564c49 base runtime-22.1-py3.9-cuda 26215f05-08c3-5a41-a1b0-da66306ce658 base do_py3.8 295addb5-9ef9-547e-9bf4-92ae3563e720 base autoai-ts_3.8-py3.8 2aa0c932-798f-5ae9-abd6-15e0c2402fb5 base tensorflow_1.15-py3.6 2b73a275-7cbf-420b-a912-eae7f436e0bc base kernel-spark3.3-py3.9 2b7961e2-e3b1-5a8c-a491-482c8368839a base

pytorch_1.2-py3.6 2c8ef57d-2687-4b7d-acce-01f94976dac1 base spark-mllib_2.3 2e51f700-bca0-4b0d-88dc-5c6791338875 base pytorch-onnx_1.1-py3.6-edt 32983cea-3f32-4400-8965-dde874a8d67e base spark-mllib_3.0-py37 36507ebe-8770-55ba-ab2a-eafe787600e9 base spark-mllib_2.4 390d21f8-e58b-4fac-9c55-d7ceda621326 base xgboost_0.82-py3.6 39e31acd-5f30-41dc-ae44-60233c80306e base pytorch-onnx_1.2-py3.6-edt 40589d0e-7019-4e28-8daa-fb03b6f4fe12 base default_r36py38 41c247d3-45f8-5a71-b065-8580229facf0 base autoai-ts_rt22.1-py3.9 4269d26e-07ba-5d40-8f66-2d495b0c71f7 base autoai-obm_3.0 42b92e18-d9ab-567f-988a-4240ba1ed5f7 base pmml-3.0_4.3 493bcb95-16f1-5bc5-bee8-81b8af80e9c7 base spark-mllib_2.4-r_3.6 49403dff-92e9-4c87-a3d7-a42d0021c095 base xgboost_0.90-py3.6 4ff8d6c2-1343-4c18-85e1-689c965304d3 base pytorch-onnx_1.1-py3.6 50f95b2a-bc16-43bb-bc94-b0bed208c60b base autoai-ts_3.9-py3.8 52c57136-80fa-572e-8728-a5e7cbb42cde base spark-mllib_2.4-scala_2.11 55a70f99-7320-4be5-9fb9-9edb5a443af5 base spark-mllib_3.0 5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9 base autoai-obm_2.0 5c2e37fa-80b8-5e77-840f-d912469614ee base spss-modeler_18.1 5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b base cuda-py3.8 5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e base autoai-kb_3.1-py3.7 632d4b22-10aa-5180-88f0-f52dfb6444d7 base pytorch-onnx_1.7-py3.8 634d3cdc-b562-5bf9-a2d4-ea90a478456b base spark-mllib_2.3-r_3.6 6586b9e3-ccd6-4f92-900f-0f8cb2bd6f0c base tensorflow_2.4-py3.7 65e171d7-72d1-55d9-8ebb-f813d620c9bb base spss-modeler_18.2 687eddc9-028a-4117-b9dd-e57b36f1efa5 base ---------- ---------- ----

Note: Only first 50 records were displayed. To display more use 'limit' parameter.

software_space_uid = client.software_specifications.get_uid_by_name("tensorflow_rt22.1- py3.9")

software_spec_uid

‘1eb25b84-d6ed-5dde-b6a5-3fbdf1665666'

ls

vegetable-dataset/ vegetable.h5 Train-model_new.tgz

model_details = client.repository.store_model(model= 'Train-model_new.tgz', meta_props={
client.repository.ModelMetaNames.NAME:"CNN",
client.repository.ModelMetaNames.TYPE:"tensorflow_2.7",
client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid})

model_id = client.repository.get_model_id(model_details)

model_id

'd0aeb6a2-e89c-4f8d-bf2f-a28ca4ea3cca'

ls

vegetable-dataset/ vegetable.h5 Train-model_new.tgz

Test The Model

import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model('vegetable.h5')
#@title
img=image.load_img(r"C:\Users\LENOVO\Desktop\vegetable-dataset\vegetable dataset\test\00fca0da-2db3-481b-b98a 9b67bb7b105c___RS_HL 7708.JPG",target_size=(128,128))
img

```
img=image.load_img(r"C:\Users\LENOVO\Desktop\ibm\Dataset Plant Disease\vegetable
dataset\vegetable-dataset\test\Tomato___healthy\0adc1c5b-8958-47c0-a152-
f28078c214f1___RS_HL 7825.JPG",target_size=(128,128))
img
x=image.img_to_array(img)
X
array([[[ 99., 86., 106.],
[101., 88., 108.],
[118., 105., 125.],
...,
[ 92., 83., 102.],
[ 93., 84., 103.],
[ 89., 80., 99.]],
[[ 96., 83., 103.],
[ 87., 74., 94.],
[102., 89., 109.],
...,
[ 88., 79., 98.],
[ 89., 80., 99.],
[ 83., 74., 93.]],
[[ 86., 73., 93.],
[ 88., 75., 95.],
[ 98., 85., 105.],
...,
[107., 98., 117.],
[ 96., 87., 106.],
[ 96., 87., 106.]],
...,
[[172., 175., 194.],
[173., 176., 195.],
[175., 178., 197.],
...,
[179., 180., 198.],
[184., 185., 203.],
[179., 180., 198.]],
[[172., 175., 194.],
[170., 173., 192.],
[173., 176., 195.],
...,
[178., 179., 197.],
[182., 183., 201.],
[178., 179., 197.]],
[[169., 172., 191.],
[166., 169., 188.],
[168., 171., 190.],
...,
[187., 188., 206.],
[185., 186., 204.],
[186., 187., 205.]]], dtype=float32) x=np.expand_dims(x,axis=0)
```

```
X
array([[[[ 99., 86., 106.],
[101., 88., 108.],
[118., 105., 125.],
...,
[ 92., 83., 102.],
[ 93., 84., 103.],
[ 89., 80., 99.]],
[[ 96., 83., 103.],
[ 87., 74., 94.],
[102., 89., 109.],
...,
[ 88., 79., 98.],
[ 89., 80., 99.],
[ 83., 74., 93.]],
[[ 86., 73., 93.],
[ 88., 75., 95.],
[ 98., 85., 105.],
...,
[107., 98., 117.],
[ 96., 87., 106.],
[ 96., 87., 106.]],
...,
[[172., 175., 194.],
[173., 176., 195.],
[175., 178., 197.],
...,
[179., 180., 198.],
[184., 185., 203.],
[179., 180., 198.]],
[[172., 175., 194.],
[170., 173., 192.],
[173., 176., 195.],
...,
[178., 179., 197.],
[182., 183., 201.],
[178., 179., 197.]],
[[169., 172., 191.],
[166., 169., 188.],
[168., 171., 190.],
...,
[187., 188., 206.],
[185., 186., 204.],
[186., 187., 205.]]]], dtype=float32)
y=np.argmax(model.predict(x),axis=1)
1/1 [==============================] - 0s 105ms/step
x_train.class_indices
{'Tomato___Blight': 0, 'Tomato___healthy': 1, 'Corn_(maize)___Northern_Leaf_Blight': 2,
'Corn_(maize)___healthy': 3, 'Potato___Blight': 4, 'Potato___healthy': 5}
```

```
index=['Tomato___Blight','Tomato___healthy','Corn_(maize)___Northern_Leaf_Blight','Cor
n_(maize)___healthy','Potato___Blight','Potato___healthy']
index[y[0]]
'Tomato___healthy'
img=image.load_img(r"C:\LENOVO\Desktop\ibm\Dataset Plant Disease\vegetable
dataset\vegetable-dataset\test\Potato___healthy\0a2ed402-5d23-4e8d-bc98-
b264aea9c3fb___Rutg._HL 2471.JPG",target_size=(128,128))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
index=['Tomato___Blight','Tomato___healthy''Potato___Blight','Potato___healthy'] index[y[0]]
1/1 [==============================] - 0s 26ms/step
'Potato___healthy'
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask,render_template,request
app=Flask(__name__)
model=load_model("vegetable.h5")
@app.route('/')
def index():
return render_template("index.html")
@app.route('/predict',methods=['GET','POST'])
def upload():
if request.method=='POST':
f=request.files['image']
basepath=os.path.dirname('__file__')
filepath=os.path.join(basepath,'uploads',f.filename)
f.save(filepath)
img=image.load_img(filepath,target_size=(128,128))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=np.argmax(model.predict(x),axis=1)
index=['Tomato___Blight','Tomato___healthy', ,'Potato___Blight','Potato___healthy'] text="The
Classified Vegetable disease is : " +str(index[pred[0]]) return text
if __name__=='__main__':
app.run(debug=False)
```

ibmapp.py

```
import requests
from tensorflow.keras.preprocessing import image from tensorflow.keras.models import load_model
import numpy as np
import pandas as pd
import tensorflow as tf
from flask import Flask, request, render_template, redirect, url_for import os
from werkzeug.utils import secure_filename
app = Flask(_name_)
#load both the vegetable and fruit models
```

```
model = load_model("IBM-vegetable.h5")
model1=load_model("IBM-fruit.h5")
#home page
@app.route('/')
def home():
return render_template('home.html')
#prediction page
@app.route('/prediction')
def prediction():
return render_template('predict.html')
@app.route('/predict',methods=['POST'])
def predict():
if request.method == 'POST':
# Get the file from post request
f = request.files['image']
# Save the file to ./uploads
basepath = os.path.dirname(_file_)
file_path = os.path.join(
basepath, 'uploads', secure_filename(f.filename)) f.save(file_path)
img = image.load_img(file_path, target_size=(128, 128))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
plant=request.form['plant']
print(plant)
if(plant=="vegetable"):
preds = model.predict(x)
preds=np.argmax(preds)
print(preds)
df=pd.read_excel('precautions - veg.xlsx') print(df.iloc[preds]['caution'])
else:
preds = model1.predict(x)
preds=np.argmax(preds)
df=pd.read_excel('precautions - fruits.xlsx') print(df.iloc[preds]['caution']
return df.iloc[preds]['caution']
if _name_ == "_main_":
app.run(debug=False)
```

GitHub Link

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-2459-1658471981

VIDEO DEMO LINK:

https://drive.google.com/file/d/1_3XY-c5EXRVmBjKwRirTM0V68DsUuCpY/view?usp=drivesdk