TEAM ID : PNT2022TMID18295

PROJECT TITLE : Fertilizers Recommendation System for Disease Prediction

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SONA COLLEGE OF TECHNOLOGY

(AN AUTONOMUS INSTITUTION)

Fertilizers Recommendation System for Disease Prediction

A PROJECT REPORT

Submitted by

Devdharshini M -1919102032

Dheepana K K -1919102037

Gayathiri S K - 1919102043

Guttula Lakshmi Vidhya - 1919102049

November 2022

INDEX PAGE

S.No	Title	Page.No
1	Introduction	3
2	Literature Survey	3
3	Ideation & Proposed Solution	4
4	Requirement Analysis	6
5	Project Design	7
6	Project Planning & Scheduling	10
7	Coding & Solution	11
8	Testing	12
9	Results	13
10	Advantages & Disadvantages	18
11	Conclusion	19
12	Future Scope	19
13	Appendix	19

1) INTRODUCTION

1.1) Overview

To Detect and recognize the plant diseases and to recommend fertilizer, it is necessary to provide symptoms in identifying the disease at its earliest. Hence the authors proposed and implemented new fertilizers Recommendation System for crop disease prediction.Two datasets, the fruit dataset and the vegetable dataset, are gathered for this project. Convolutional Neural Networks (CNN), a deep learning neural network, are used to train and evaluate the collected datasets.CNN is used to first train and then test the fruit dataset.There are six classes, and each one has been trained and tested.Second, the training and testing of the vegetable dataset take place.The product utilized for preparing and testing of datasets is Python.After being written in the Jupyter notebook that comes with Anaconda Python, all of the Python codes are tested in the IBM cloud.Finally, with the assistance of the Python library Flask, a web-based framework is created.Two HTML files, as well as the static folder files that are associated with them, are created in the templates folder.Spyder-Anaconda Python is used to write and test the Python program "app.py" that connects to these two websites**.**

1.2) Purpose

Agriculture is the main aspect of country development. Many people lead their life from agriculture field, which gives fully related to agricultural products. Plant disease, especially on leaves, is one of the major factors of reductions in both quality and quantity of the food crops. In agricultural aspects, if the plant is affected by leaf disease, then it reduces the growth of the agricultural level. Finding the leaf disease is an important role of agriculture preservation. After pre-processing using a median filter, segmentation is done by Guided Active Contour method and finally, the leaf disease is identified by using Support Vector Machine. The disease-based similarity measure is used for fertilizer recommendation.

2) LITERATURE SURVEY

2.1) Existing problems

Indumathi 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.Pandi selvi proposed a simple prediction method for soil based fertilizer recommendation system for predicted crop diseases. This method gives less accuracy and prediction.

2.2) References

1. R Indumathi.; N Saagari.; V Thejuswini.; R Swarnareka.," Leaf Disease Detection and Fertilizer Suggestion", IEEE International Conference on System, Computatio -n, References.

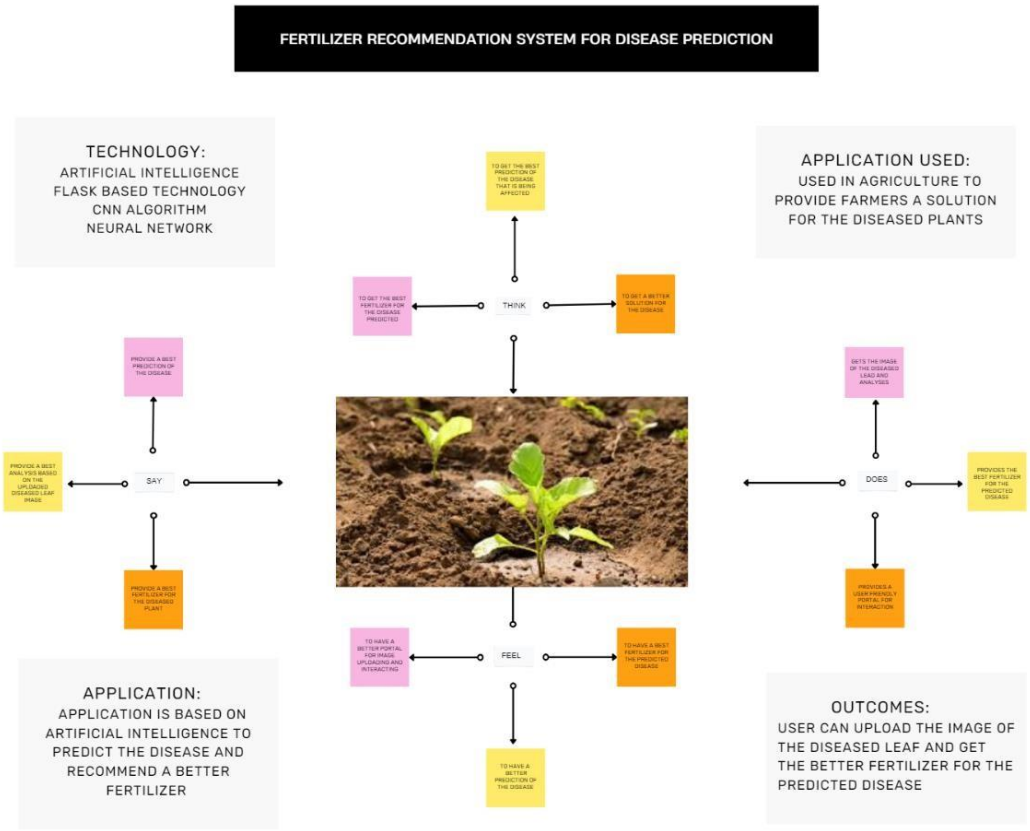
2. P. Pandi Selvi, P. Poornima, "Soil Based Fertilizer Recommendation System for Crop Disease Prediction System", International Journal of Engineering Trends a nd Applications (IJETA) – Volume 8 Issue 2, Mar-Apr 2021.

2.3) Problem Statement Definition

The most significant sector in modern life is agriculture.A wide range of bacterial and fungal diseases affect most plants.Plant diseases hampered production significantly and posed a significant threat to food security.As a result, in order to guarantee maximum quantity and quality, plant diseases must be identified promptly and accurately.The variety of pathogen varieties, shifts in cultivation practices, and inadequate plant protection methods have all contributed to an increase in the number and severity of plant diseases in recent years. By examining the symptoms that are displayed on the leaves of the plant, an automated system is introduced to identify various plant diseases.

3) IDEATION & PROPOSED SOLUTION

3.1) Empathy map canvas

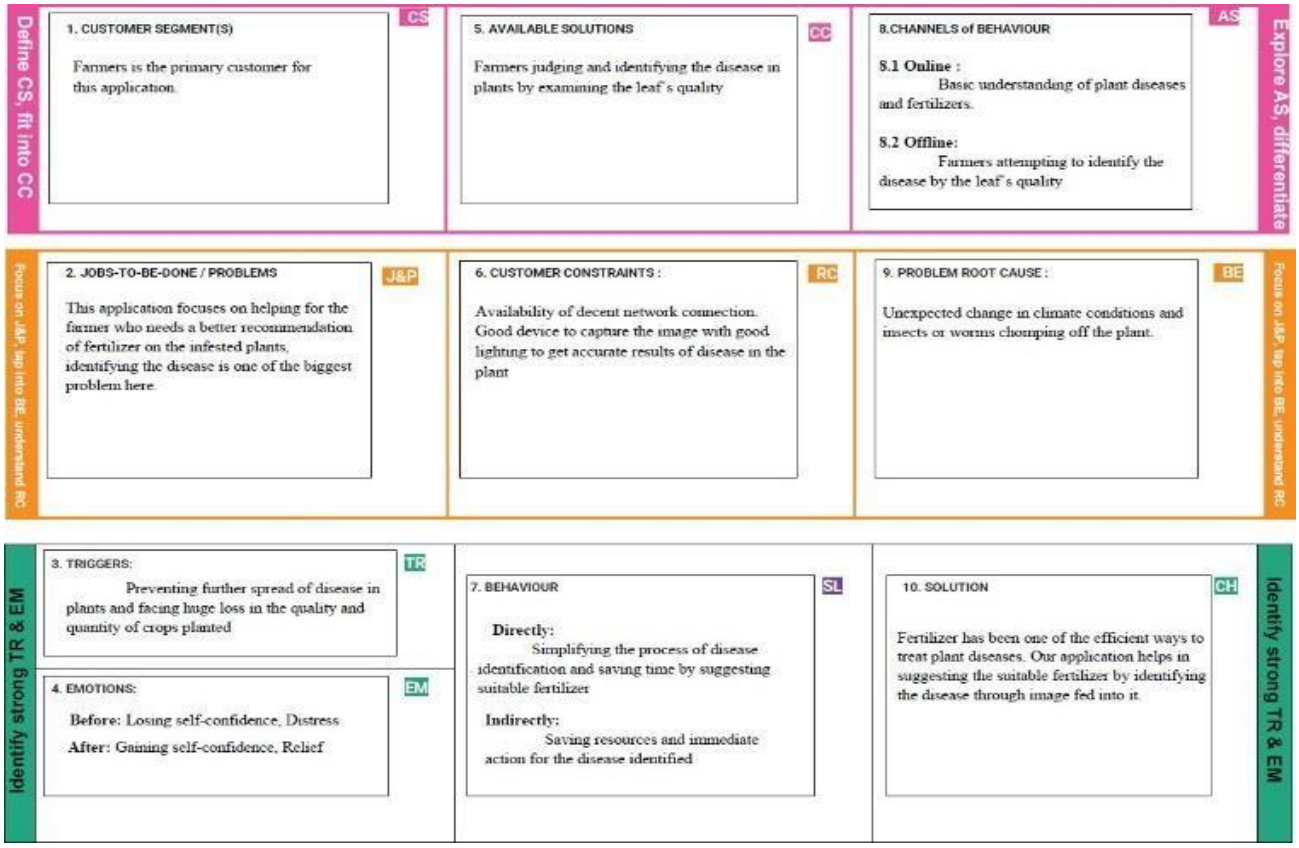


3.2) Ideation and Brainstorming

3.3) Proposed Solution

A deep learning-based neural network is used to train and test the collected datasets in this project.CNN, a neural network based on deep learning, achieves classification accuracy of more than 90%.The accuracy rate can be increased to 95% to 98% by increasing the number of dense layers and modifying hyper parameters like batch size and number of epoch.

3.4) Problem solution fit



4) REQUIREMENT ANALYSIS

4.1) Functional Requirements

Following are the functional requirements of the proposed solution

FR.No	Functional Requirements	Sub Requirements
FR-1	User Registration	• Registration through Form • Registration through Gmail • Registration through LinkedIN
FR-2	User Confirmation	• Confirmation via OTP • Confirmation via Email
FR-3	Capturing Image	Capture the image of the leaf and the parameter of the captured image.
FR-4	Image Processing	Upload the image for the prediction of the disease in the leaf.
FR-5	Leaf Identification	Identify the leaf and predict the disease in the leaf.

FR-6	Image Description	Suggesting the best fertilizer for the disease.

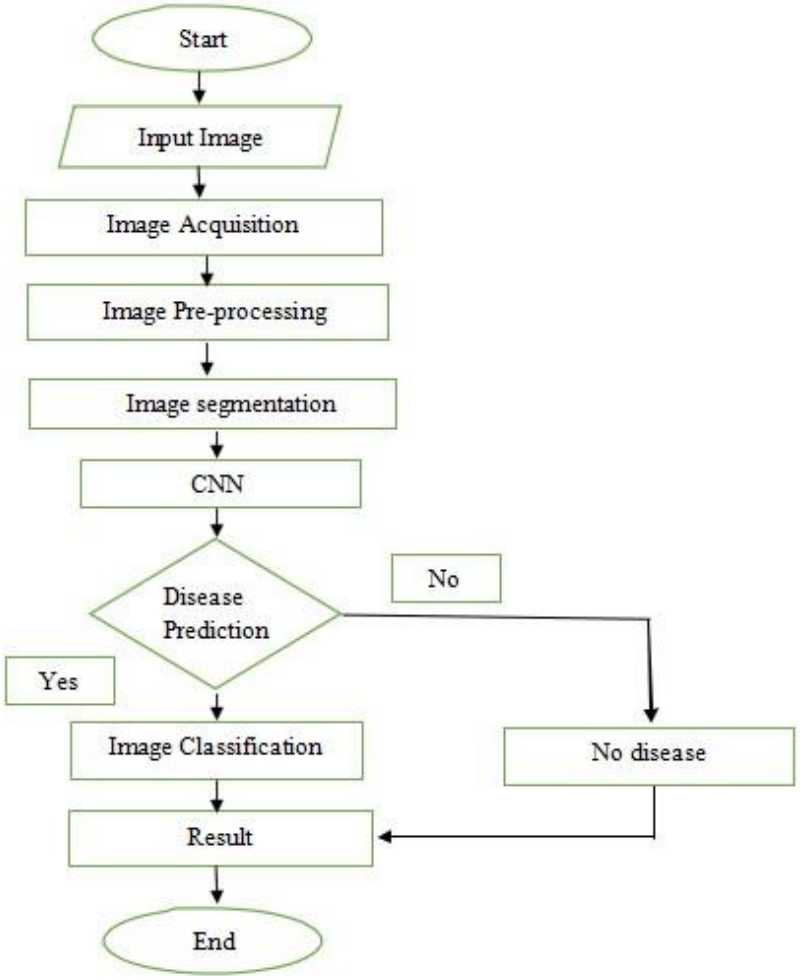
4.2 Non Functional Requirements

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

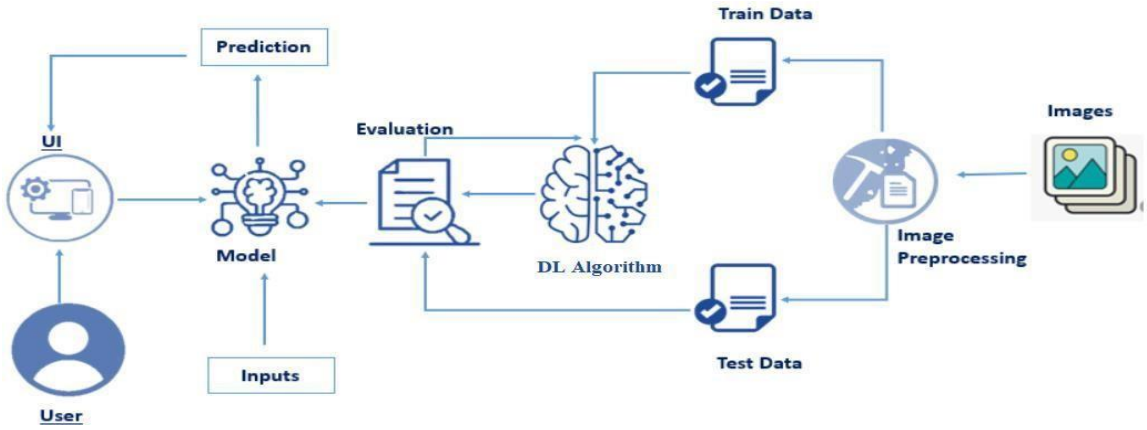
NFr.no	Non-Functional Requirement	Description
Nfr-1	Usability	Datasets of all the leaves are used to detect the disease that is present in the leaf.
Nfr-2	Security	The information belongs to the user and leaves are secured highly.
Nfr-3	Reliability	The leaf quality is important for predicting the disease in the leaf.
Nfr-4	Performance	The performance is based on the quality of the leaf used for disease prediction
Nfr-5	Availability	It is available for all users to predict the disease in the plant
Nfr-6	Scalability	Increasing the prediction of the disease in the leaf

5) PROJECT DESIGN

5.1) Data Flow Diagram



5.2 Solution & Technical Architecture



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Login	USN-1	As a user, I can log into the application by entering my email & password	I can access the UI and obtain the necessary results	Medium	Sprint-1
	Interactive UI	USN-2	As a user, I can easily understand and use the application	I can access the UI and obtain the necessary results	High	Sprint-1
	Fruit Disease	USN-3	As a user, I can separately upload the image for fruit disease prediction	I can upload and access the application through a separate UI	High	Sprint-2
	Vegetable Disease	USN-4	As a user, I can separately upload the image for vegetable disease prediction	I can upload and access the application through a separate UI	High	Sprint-2
	Logout	USN-5	As a user, I can log out of the application	I can end my session in the application by logging off	Medium	Sprint-1

6) PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement	User story Number	User Story/ Task	Story Points	Priorty	Team Members
Sprint 1	Image processing		Process the images for model training	2	High	• Devdharshini • Dheepana • Gayathiri • G.Lakshmi Vidhya

	Data Collection		Gather the data set from kaggle	2	High	• Devdharshini • Dheepana • Gayathiri • G.Lakshmi Vidhya

Sprint	Functional Requirement	User story Number	User Story/ Task	Story Point	Priory	Team Members
Sprint-2	model building and training(fruit)		Create a model which can classify diseased fruit	8	High	• Devdharshini • Dheepana • Gayathiri
			plants from given images.			• G.Lakshmi Vidhya
	Model building and training(Vegetable)		Create a model which can classify diseased vegetable plants from given images.	8	High	• • Devdharshini • Dheepana • Gayathiri G.Lakshmi Vidhya

Sprint	Functional Requirement	User story Number	User Story/ Task	Story Points	Priority	Team Members
sprint-3	Testing the model (Fruit)		Test the Fruit model built and train it on IBM Cloud	8	High	• Devdharshini • Dheepana • Gayathiri • G.Lakshmi Vidhya
	Testing the model (Vegetable)		Test the Vegetable model built and Train it on IBM Cloud	8	High	• Devdharshini • Dheepana • Gayathiri • G.Lakshmi Vidhya

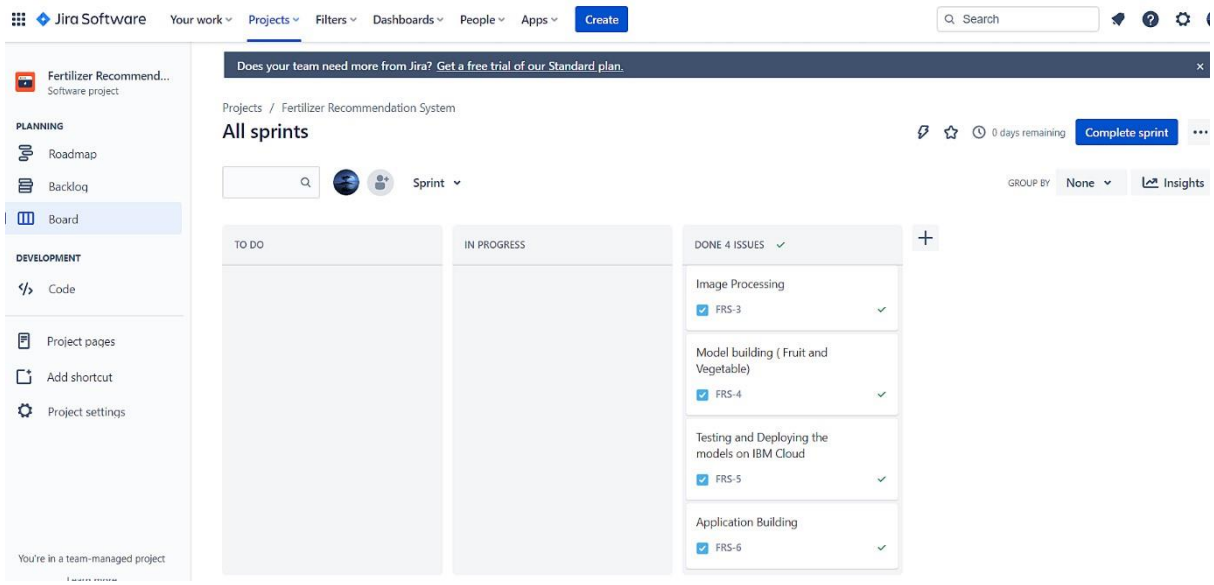
Sprint	Functional Requirement	User story Number	User Story/ Task	Story Points	Priorty	Team Members
Sprint -4	Application Building		Develop the Web pages. Home.html and predict.html	8	HIgh	• Devdharshini • Dheepana • Gayathiri • G.Lakshmi Vidhya

	Flask Application		Integrate the HTML pages with Flask and build web application	8	High	• Devdharshini • Dheepana • Gayathiri • G.Lakshmi Vidhya

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Sprint Release Date (Actual)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022	30 Oct 2022
Sprint-2	15	6 Days	31 Oct 2022	05 Nov 2022	06 Nov 2022
Sprint-3	15	6 Days	07 Nov 2022	12 Nov 2022	13 Nov 2022
Sprint-4	12	6 Days	14 Nov 2022	19 Nov 2022	20 Nov 2022

6.3 Reports from JIRA



7) CODING & SOLUTIONING

7.1) Feature-1

The fruit,vegetable disease prediction is developed using the CNN algorithm to predict the diseases. The model is trained over six classes and 5384 images and is tested over six classes with 1686 images to get better accuracy. We added a single convolution layer and max pooling with size 2 for better accuracy more convolution layers can be added, with higher max pooling. Then the model is trained over two epochs, this can be increased for better accuracy rates.

```
In [5]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
        train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
        test_datagen=ImageDataGenerator(rescale=1)
        x_train=train_datagen.flow_from_directory(r'C:\Users\krishna\OneDrive\Desktop\Fertilizer recommendation system for disease predic

        x_test=test_datagen.flow_from_directory(r'C:\Users\krishna\OneDrive\Desktop\Fertilizer recommendation system for disease predicti

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

In [6]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense,Convolution2D,MaxPooling2D,Flatten

In [7]: 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()
```

7.2) Feature 2

The models are then deployed on IBM Cloud deployment space and are trained using IBM machine learning services. The trained models are downloaded and tested for application development.

```
In [24]: model_details

Out[24]: {'entity': {'hybrid_pipeline_software_specs': [],
           'software_spec': {'id': 'acd9c798-6974-5d2f-a657-ce06e986df4d',
            'name': 'tensorflow_rt22.1-py3.9'},
           'type': 'tensorflow_2.7'},
          'metadata': {'created_at': '2022-11-16T18:37:37.349Z',
           'id': 'd0685141-b035-4830-80e4-abaebe5c2a40',
           'modified_at': '2022-11-16T18:39:57.388Z',
           'name': 'FRUIT CNN model',
           'owner': 'IBMid-668000FC50',
           'resource_key': 'c0e5746e-6f32-4a2a-ae0b-a628051952a2',
           'space_id': '7ab0dcb4-0291-4cd2-a251-ed4c2cc7f5ac'},
          'system': {'warnings': []}}

In [25]: model_id=client.repository.get_model_id(model_details)
         model_id

Out[25]: 'd0685141-b035-4830-80e4-abaebe5c2a40'

In [26]: client.repository.download(model_id,'fruit_model_ibm.tar.gn')

         Successfully saved model content to file: 'fruit_model_ibm.tar.gn'

Out[26]: 'C:\\Users\\krishna\\OneDrive\\Desktop\\Fertilizer recommendation system for disease prediction\\IBM TRAINING/fruit_model_ibm.t
         ar.gn'
```

8) TESTING

8.1) Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	20	0	1	19

Security	2	0	0	2
Outsource Shipping	0	0	0	0
Exception Reporting	10	0	0	10
Final Report Output	1	0	0	1
Version Control	1	0	0	1

8.2) User Acceptance Testing

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	20	0	1	19
Security	2	0	0	2
Outsource Shipping	0	0	0	0
Exception Reporting	10	0	0	10
Final Report Output	1	0	0	1
Version Control	1	0	0	1

9) RESULTS

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

```
In [11]: model.add(Dense(300,activation='relu'))
         model.add(Dense(150,activation='relu'))
         model.add(Dense(6,activation='softmax'))
         model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
         model.fit(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=3)

         Epoch 1/3
         2692/2692 [==============================] - 819s 304ms/step - loss: 1.6518 - accuracy: 0.3338 - val_loss: 1.6453 - val_accurac
         y: 0.2924
         Epoch 2/3
         2692/2692 [==============================] - 840s 312ms/step - loss: 1.6482 - accuracy: 0.3351 - val_loss: 1.6483 - val_accurac
         y: 0.2924
         Epoch 3/3
         2692/2692 [==============================] - 7306s 3s/step - loss: 1.6463 - accuracy: 0.3351 - val_loss: 1.6422 - val_accuracy:
         0.2924

Out[11]: <keras.callbacks.History at 0x24fc2426a90>
```

The above images shows the code and output for model building of fruit and vegetable.

Testing vegetable model :

```
In [29]: model.save('vegetabledata.h5')

In [30]: import numpy as np
         from tensorflow.keras.models import load_model
         from tensorflow.keras.preprocessing import image

In [31]: model=load_model('vegetabledata.h5')

In [32]: img=image.load_img(r"C:\Users\krishna\OneDrive\Desktop\Fertilizer recommendation system for disease prediction\Dataset Plant Dise

In [33]: img

Out[33]:
```

Testing fruit model :

```
In [18]: model.save('fruitdata.h5')

In [13]: import numpy as np
         from tensorflow.keras.models import load_model
         from tensorflow.keras.preprocessing import image

In [14]: model=load_model('fruitdata.h5')

In [15]: img=image.load_img(r"C:\Users\krishna\OneDrive\Desktop\Fertilizer recommendation system for disease prediction\Dataset Plant Dise

In [16]: img
```

Out[16]:

Testing of Fruit and vegetables models to find the accuracy.

Deploy in cloud for vegetable :

```
In [37]: model_details

Out[37]: {'entity': {'hybrid_pipeline_software_specs': [],
            'software_spec': {'id': 'acd9c798-6974-5d2f-a657-ce06e986df4d',
             'name': 'tensorflow_rt22.1-py3.9'},
            'type': 'tensorflow_2.7'},
           'metadata': {'created_at': '2022-11-16T17:31:05.821Z',
            'id': 'e03e92d6-9a4c-420b-bd91-e8c22b536741',
            'modified_at': '2022-11-16T17:33:12.979Z',
            'name': 'VEG CNN model',
            'owner': 'IBMid-668000FC50',
            'resource_key': '4f5d6cf7-4dd8-4c74-aecb-fa3edc25c6db',
            'space_id': '7ab0dcb4-0291-4cd2-a251-ed4c2cc7f5ac'},
           'system': {'warnings': []}}

In [38]: model_id=client.repository.get_model_id(model_details)
         model_id

Out[38]: 'e03e92d6-9a4c-420b-bd91-e8c22b536741'

In [39]: client.repository.download(model_id,'veg_model_ibm.tar.gn')

         Successfully saved model content to file: 'veg_model_ibm.tar.gn'

Out[39]: 'C:\\Users\\krishna\\OneDrive\\Desktop\\Fertilizer recommendation system for disease prediction\\IBM TRAINING/veg_model_ibm.ta
         r.gn'
```

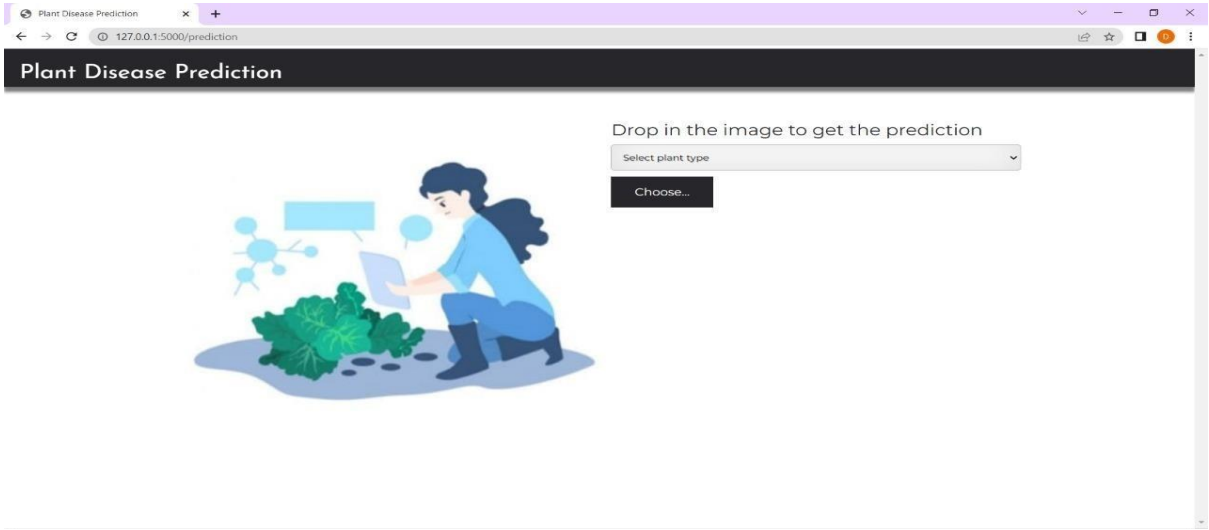
Deploy in cloud for fruit:

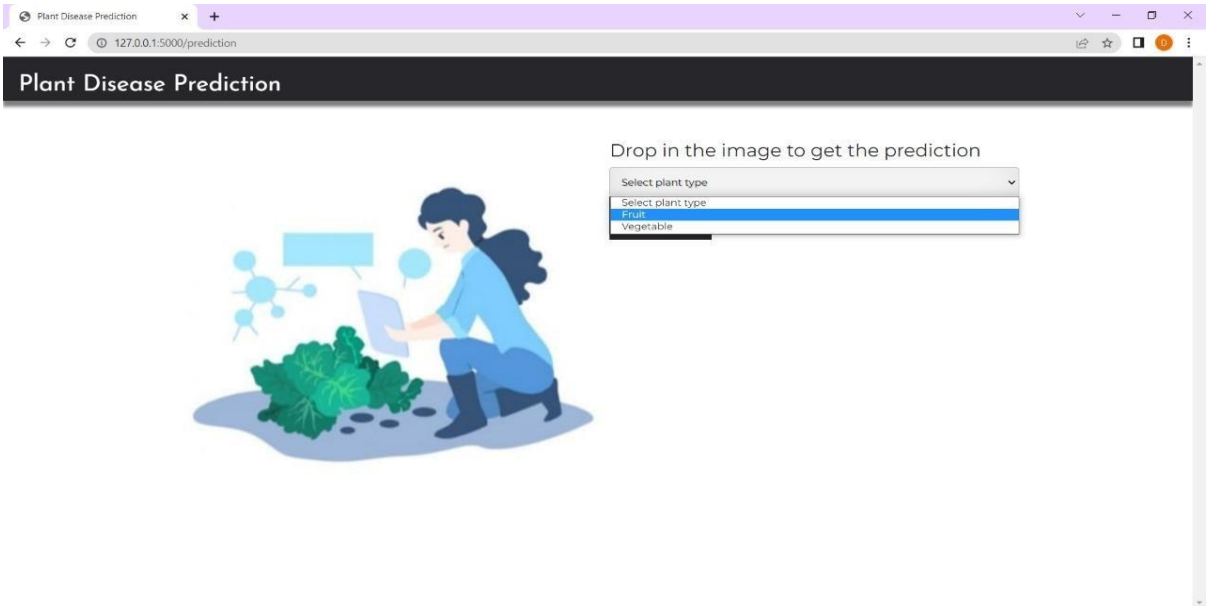
```
In [24]: model_details

Out[24]: {'entity': {'hybrid_pipeline_software_specs': [],
            'software_spec': {'id': 'acd9c798-6974-5d2f-a657-ce06e986df4d',
             'name': 'tensorflow_rt22.1-py3.9'},
            'type': 'tensorflow_2.7'},
           'metadata': {'created_at': '2022-11-16T18:37:37.349Z',
            'id': 'd0685141-b035-4830-80e4-abaebe5c2a40',
            'modified_at': '2022-11-16T18:39:57.388Z',
            'name': 'FRUIT CNN model',
            'owner': 'IBMid-668000FC50',
            'resource_key': 'c0e5746e-6f32-4a2a-ae0b-a628051952a2',
            'space_id': '7ab0dcb4-0291-4cd2-a251-ed4c2cc7f5ac'},
           'system': {'warnings': []}}

In [25]: model_id=client.repository.get_model_id(model_details)
         model_id

Out[25]: 'd0685141-b035-4830-80e4-abaebe5c2a40'

In [26]: client.repository.download(model_id,'fruit_model_ibm.tar.gn')

         Successfully saved model content to file: 'fruit_model_ibm.tar.gn'

Out[26]: 'C:\\Users\\krishna\\OneDrive\\Desktop\\Fertilizer recommendation system for disease prediction\\IBM TRAINING/fruit_model_ibm.t
         ar.gn'
```

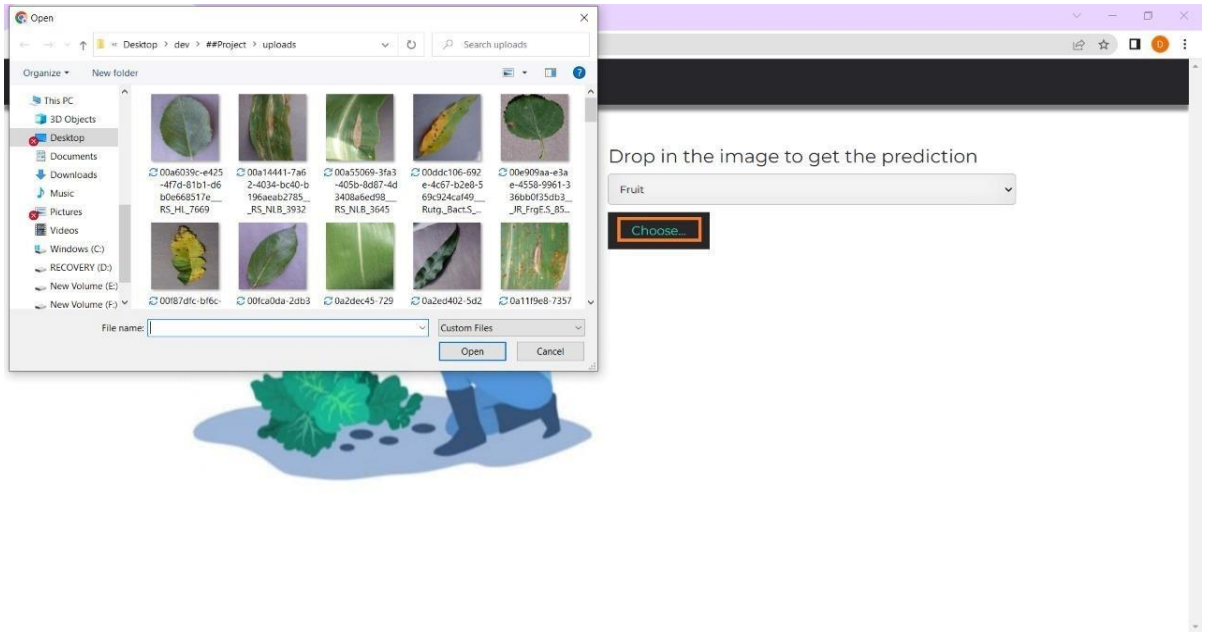
Deploying the fruit and vegetable models in IBM Cloud.

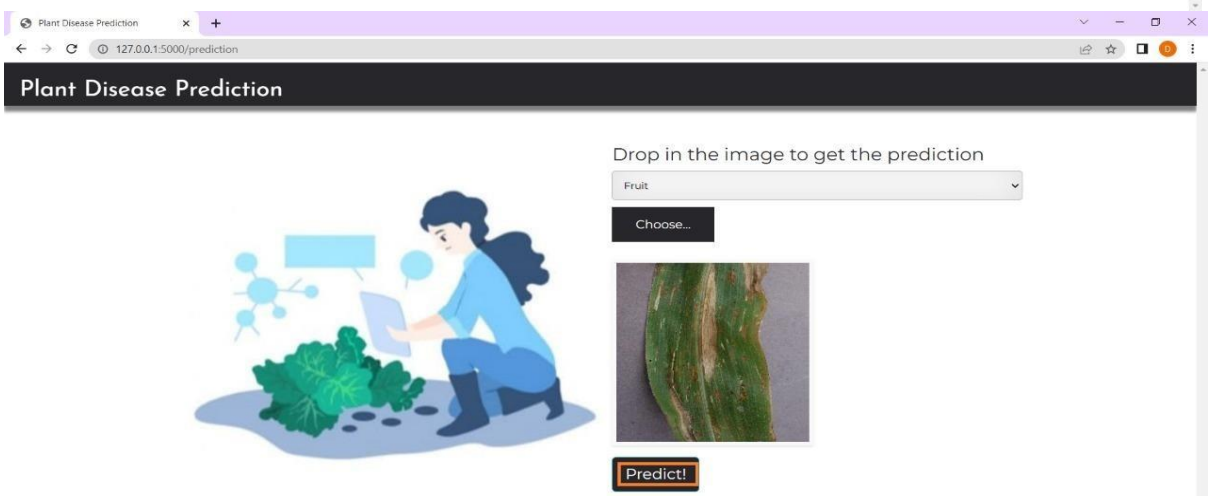
Demonstration Screenshots :

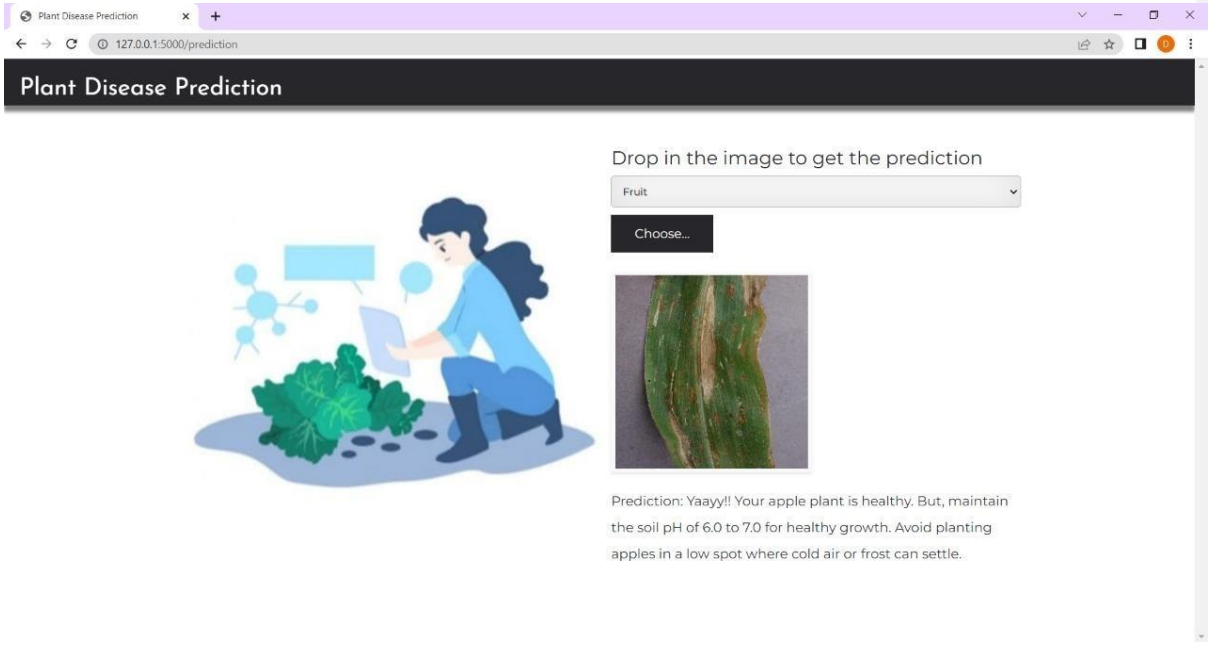
Plant Disease Prediction
127.0.0.1:5000
Plant Disease Prediction
Home
Predict
Detect if your plant is infected!!
Agriculture is one of the major sectors worls wide. Over the years it has developed and the use of new technologies and equipment replaced almost all the traditional methods of farming. The 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.

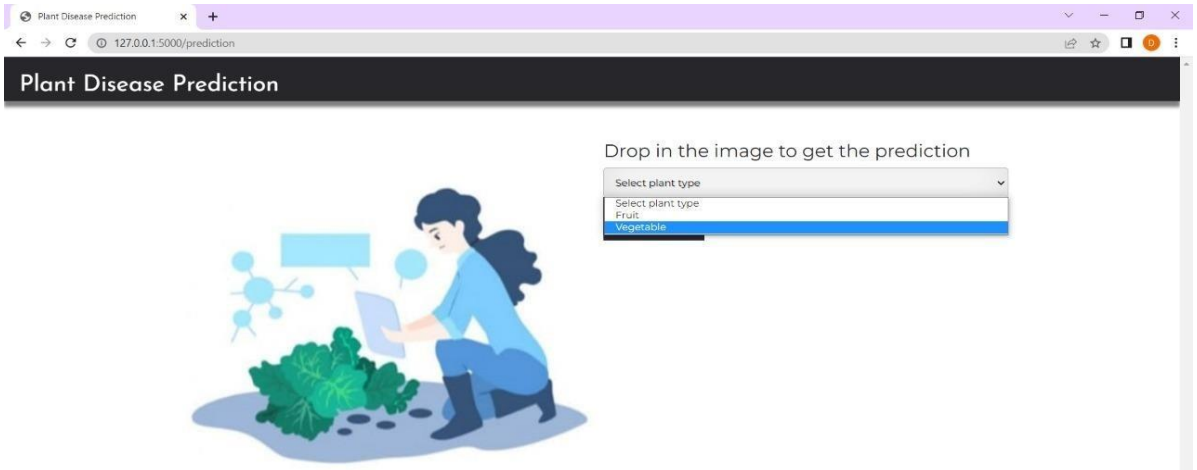

Plant Disease Prediction
127.0.0.1:5000/prediction
Plant Disease Prediction
Drop in the image to get the prediction
Select plant type
Choose...

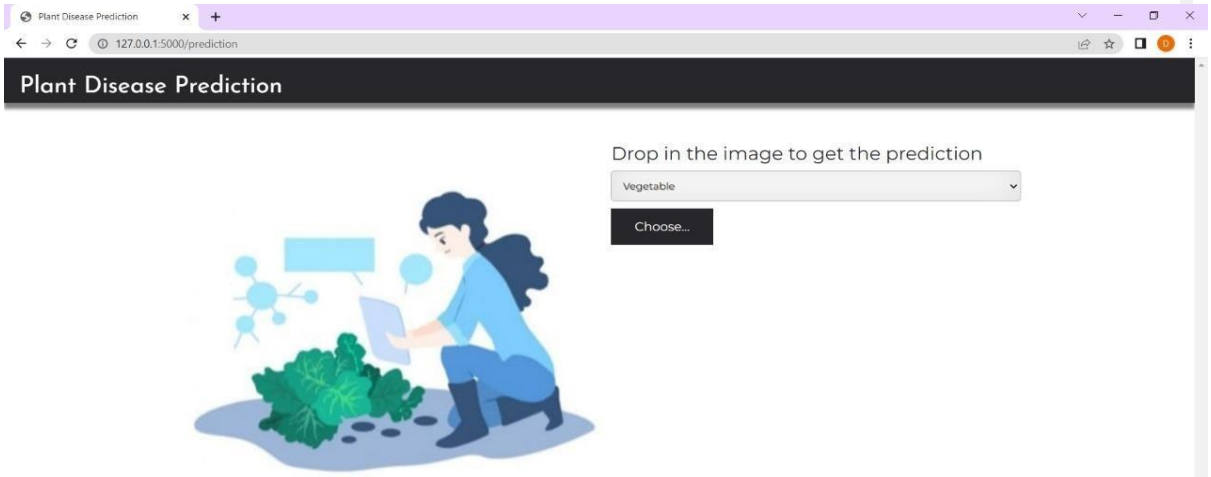

Plant Disease Prediction
127.0.0.1:5000/prediction
Plant Disease Prediction
Drop in the image to get the prediction
Select plant type
Select plant type
Fruit
Vegetable

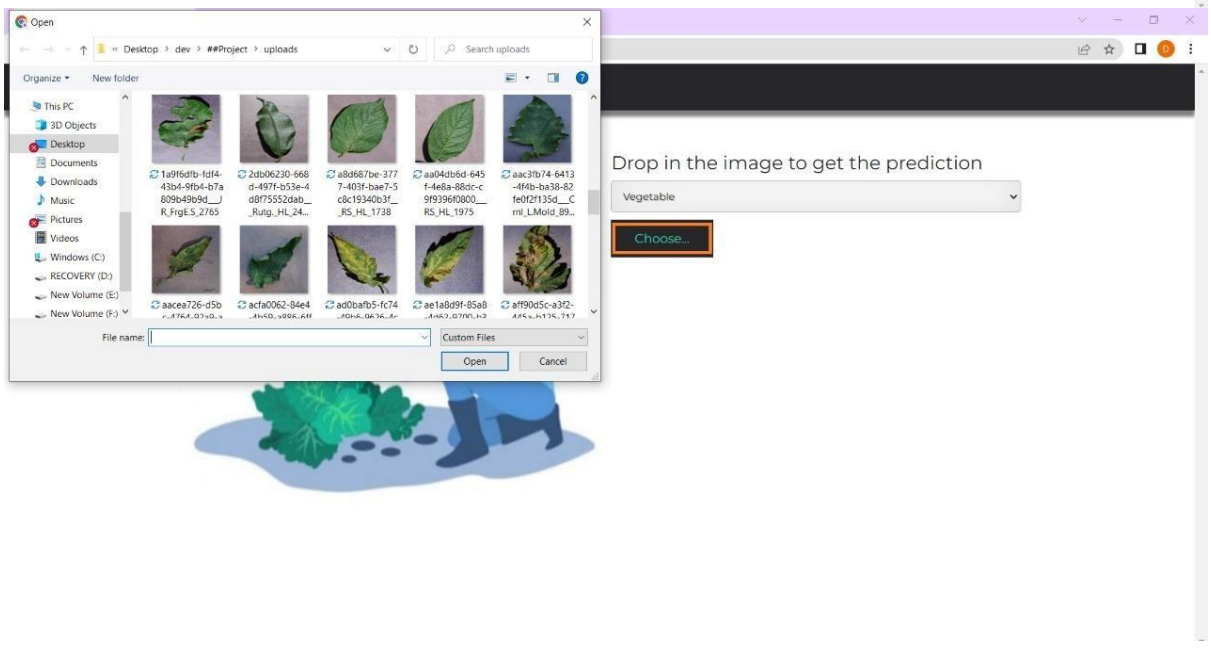

Open
Desktop > dev > ##Project > uploads
Search uploads
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
RECOVERY (D:)
New Volume (E:)
New Volume (F:)
File name:
Custom Files
Open
Cancel
Drop in the image to get the prediction
Fruit
Choose...

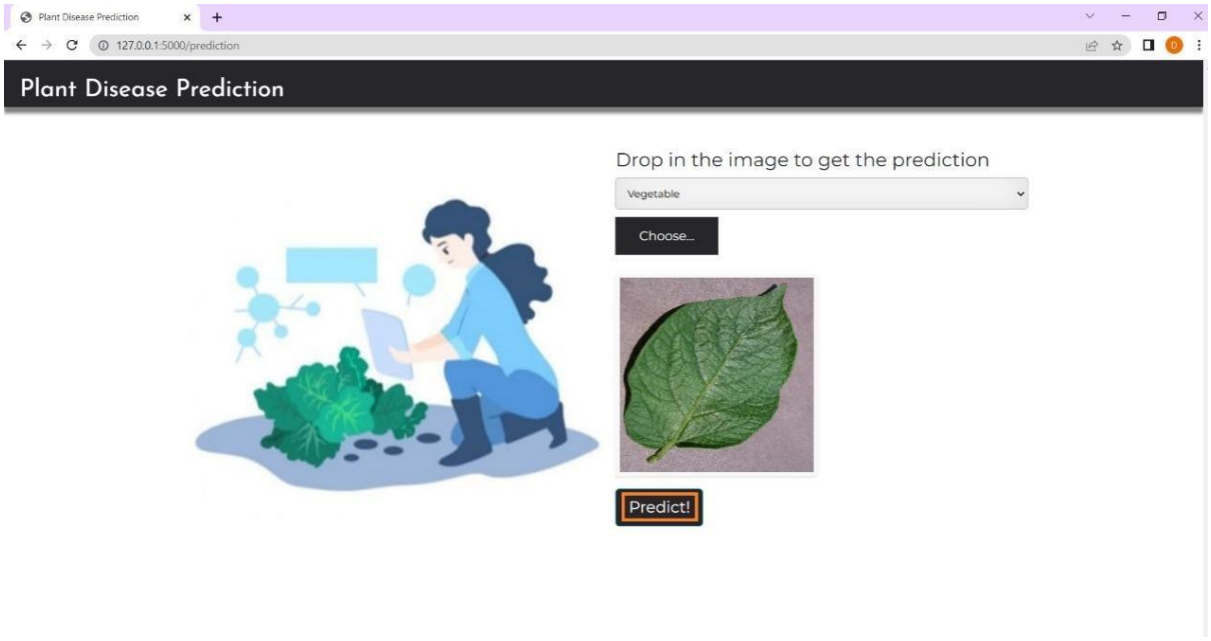

Plant Disease Prediction
127.0.0.1:5000/prediction
Plant Disease Prediction
Drop in the image to get the prediction
Fruit
Choose...
Predict!

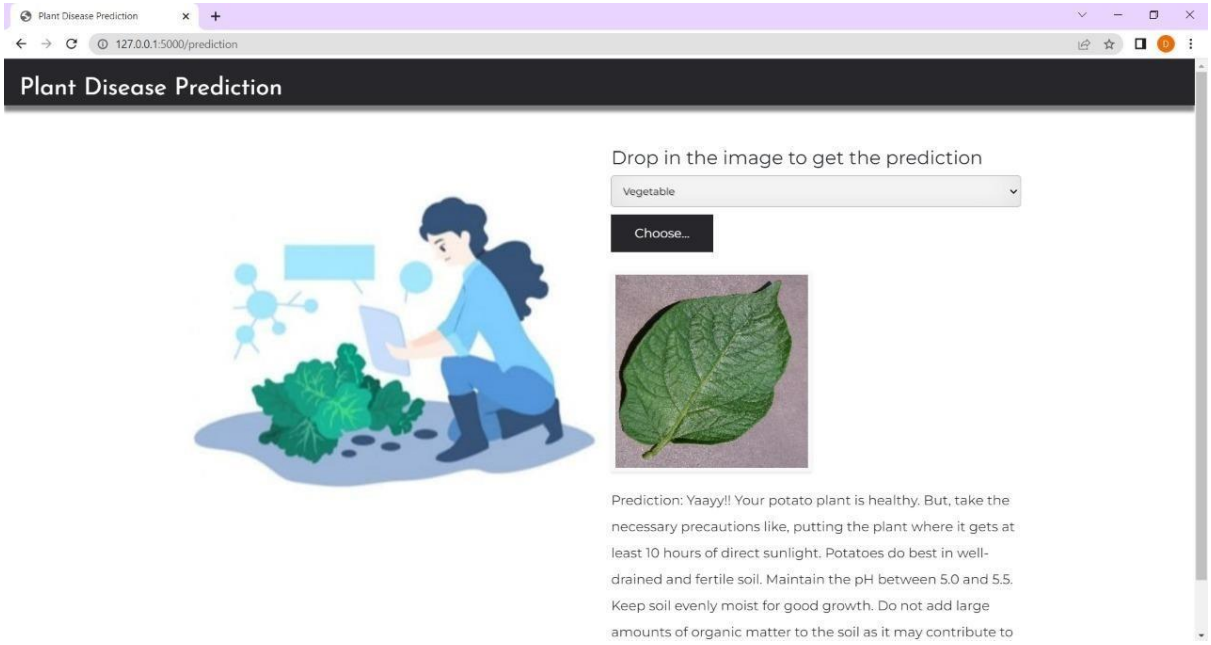

Plant Disease Prediction
127.0.0.1:5000/prediction
Plant Disease Prediction
Drop in the image to get the prediction
Fruit
Choose...
Prediction: Yaayy!! Your apple plant is healthy. But, maintain the soil pH of 6.0 to 7.0 for healthy growth. Avoid planting apples in a low spot where cold air or frost can settle.


Plant Disease Prediction
127.0.0.1:5000/prediction
Plant Disease Prediction
Drop in the image to get the prediction
Select plant type
Select plant type
Fruit
Vegetable


Plant Disease Prediction
127.0.0.1:5000/prediction
Plant Disease Prediction
Drop in the image to get the prediction
Vegetable
Choose...


Open
Desktop › dev › ##Project › uploads
Search uploads
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
RECOVERY (D:)
New Volume (E:)
New Volume (F:)
File name:
Custom Files
Open
Cancel
Drop in the image to get the prediction
Vegetable
Choose...





10) ADVANTAGES AND DISADVANTAGES

List of Advantages

- The proposed model here produces lofty accuracy of classification.
- Considerable datasets can also be trained and tested.
  Images of enormous can be resized within the proposed itself.

List of Disadvantages

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

11) CONCLUSION

The image classification of vegetable and fruit datasets is the focus of the proposed model.During the testing and training of the model, the following observations are made:

- Increasing the number of epochs led to an increase in classification accuracy.
- Different classification accuracies are obtained for various batch sizes.
- The number of convolution layers increases the accuracy.
- Variable dense layers also improved classification accuracy.
- By altering the size of the kernel that is utilized in the output of the convolution layer, various accuracies can be achieved.

When the train and test datasets are different sizes, the accuracy varies.

12) FUTURE SCOPE

This project's proposed model can be used for image recognition.Using Python to Executable Software, the entire model can be converted to application software.The OpenCV python library makes it possible to classify, recognize, and process images in real time.This project's work can be expanded to include login and signup to increase the authenticity of users.

13) APPENDIX

Github :
https://github.com/IBM-EPBL/IBM-Project-26340-1660025404

Demolink:
https://drive.google.com/file/d/1jFG36b5t1d8D7U5M1BN58UHkLqGvD45Q/view?usp=drivesdk

Source Code:
https://drive.google.com/drive/folders/10tZzrbRv60KxGByY-bpjizL3qslf2qGh

Source Code:

app.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 from

tensorflow.python.keras.backend import set_session
```

```
app = Flask(_name_)

#load both the vegetable and fruit models model
= load_model("vegetable.h5")
model1=load_model("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)
```

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'>
<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;
paddingtop: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;textalign: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;paddingright: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.jpeg')}}" style="maxheight: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>
```

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 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.jpeg')}}"
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>
```