PNT2022TMID23699

FERTILIZERS RECOMMENDATION SYSTEM FOR DISEASE PREDICTION

IBM-Project-34192-1660232517

NALAIYATHIRAN PROJECT BASED LEARNING ON PROFESSIONAL READINESS FOR INNOVATION, EMPLOYING AND ENTERPRENEURSHIP

Project Report by:

GOLLAKARAM GANGA BHAVANI (612919101024)

JADAPALLI SHIVANI (612919101035)

AMUDALAPALLI AKSHAYA (612919101002)

S. JAYANANTHINI (612919101036)

Bachelor of Engineering

Computer Science and Engineering

Vivekanandha College of Engineering for Women

Namakkal-637205

INDEX

1. ABSTRACT:

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.

a. Purpose:

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.

2. INTRODUCTION:

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. Generally, the plant diseases are caused by the abnormal physiological functionalities of plants. Therefore, the characteristic symptoms are generated based on the differentiation between normal physiological functionalities and abnormal physiological functionalities of the plants. Mostly, the plant leaf diseases are caused by pathogens which are positioned on the stems of the plants. These different symptoms and diseases of leaves are predicted by different methods in image processing. These different methods include different fundamental processes like segmentation, feature extraction
and classification and so on. Mostly, the prediction and diagnosis of leaf diseases are depending on the segmentation such as segmenting the healthy tissues from diseased tissues of leaves

3. MATERIAL AND METHODS:

A digital camera or similar devices are used to take images of different types, and then those are used to identify the affected area in leaves. Then different types of image-processing techniques are applied to them, the process those images, to get different and useful features needed for the purpose of analyzing later-Plant leaf disease identification is especially needed to predict both the quality and quantity of the First segmentation step primarily based on a mild polygonal leaf model is first achieved and later used to guide the evolution of an energetic contour. Combining global shape descriptors given by the polygonal model with local curvaturebased features, the leaves are then classified overleaf datasets. In this research work introduce a method designed to deal with the obstacles raised by such complex images, for simple and plant leaves. A first segmentation step based on graph-cut approach is first performed and later used to guide the evolution of leaf boundaries, and implement classification algorithm to classify the diseases and recommend the fertilizers to affected leaves as shown in Figure 1.

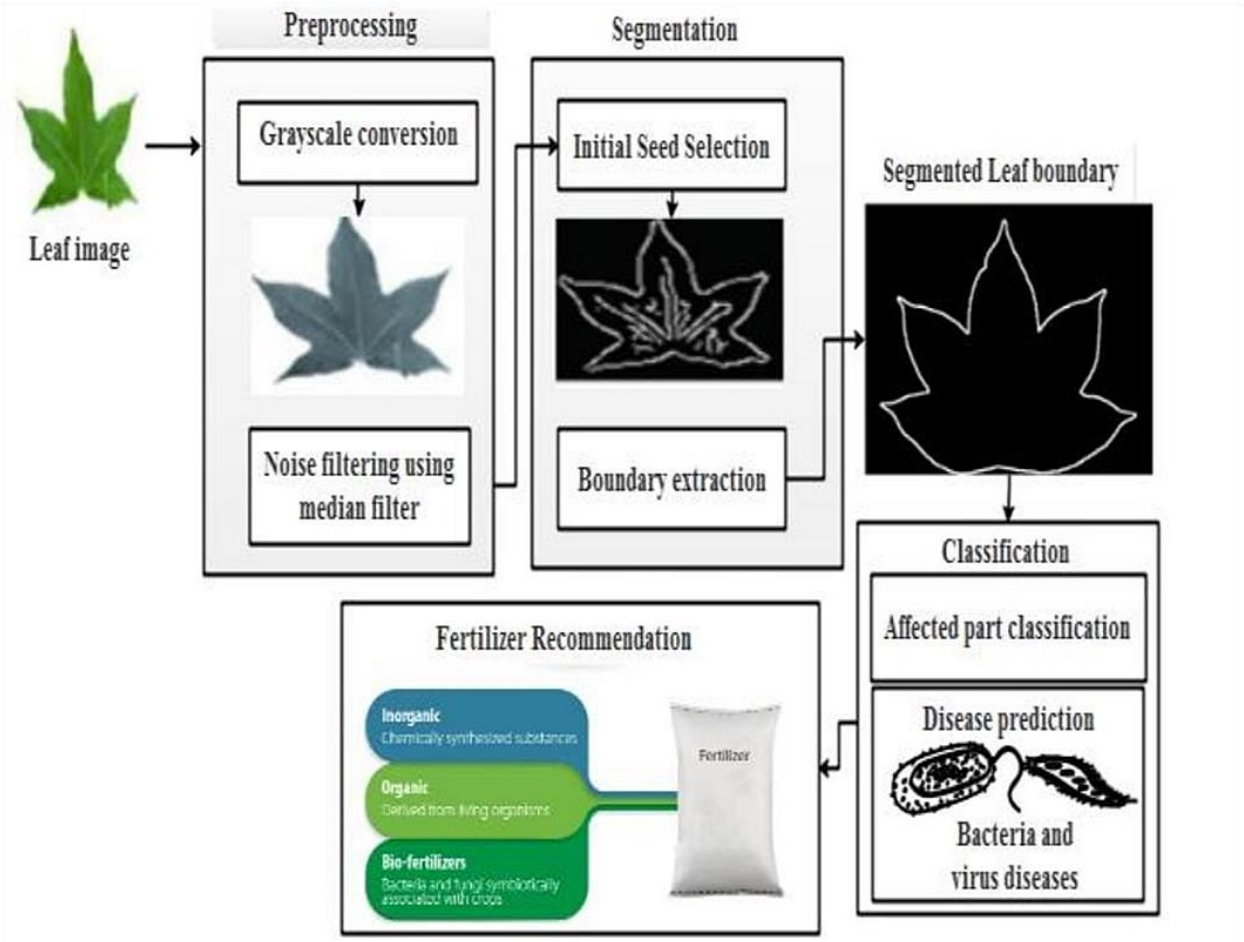


Figure.1 *Proposed Architecture*

3.1 Image Classification Steps:

The proposed image classification technique is divided into the following steps.

3.1.1 Image acquisition:

To get the image of a leaf so that evaluation in the direction of a class can be accomplished

3.1.2 Preprocessing:

The purpose of image preprocessing is improving image statistics so that undesired distortions are suppressed and image capabilities which are probably

relevant for similar processing are emphasized. The preprocessing receives an image as input and generates an output image as a grayscale, an invert and a smoothed one.

3.1.3 Segmentation:

Implements Guided active contour method. Unconstrained active contours applied to the difficult natural images. Dealing with unsatisfying contours, which would try and make their way through every possible grab cut in the border of the leaf. The proposed solution is used the polygonal model obtained after the first step not only as an initial leaf contour but also as a shape prior that will guide its evolution towards the real leaf boundary.

3.1.4 Disease Prediction:

Leaves are affected by bacteria, fungi, virus, and other insects. Support Vector Machine (SVM) algorithm classifies the leaf image as normal or affected. Vectors are constructed based on leaf features such as color, shape, textures. Then hyperplane constructed with conditions to categorize the preprocessed leaves and also implement multiclass classifier, to predict diseases in leaf image with improved accuracy.

3.1.5 Fertilizer Recommendation:

Recommend the fertilizer for affected leaves based on severity level. Fertilizers may be organic or inorganic. Admin can store the fertilizers based on disease categorization with severity levels. The measurements of fertilizers suggested based on disease severity.

3.2. SVM Classification Algorithm:

Support Vector Machine (SVM) SVM is a binary classifier to analyze the data and recognize the pattern for classification. The main goal is to design a hyperplane that classifies all the training vectors in different classes. The objective of SVM is to

identify a function Fx which obtain the hyper-plane. Hyperplane separates two classes of data sets. The linear classifier is defined as the optimal separating hyperplane. The data sets can be separated in two ways: linearly separated or nonlinearly separated. The vectors are said to be optimally separated if they are separated without error and the distance between the two closest vector points is maximum.

4. LITERATURE SURVEY:

a. Existing Problem:

Adequate mineral nutrition is central to crop production. However, it can also exert considerable Influence on disease development. Fertilizer application can increase or decrease development of diseases caused by different pathogens, and the mechanisms responsible are complex, including effects of nutrients on plant growth, plant resistance mechanisms and direct effects on the pathogen. The effects of mineral nutrition on plant disease and the mechanisms responsible for those effects have been dealt with comprehensively elsewhere. In India, around 40% of land is kept and grown using reliable irrigation technologies, while the rest relies on the monsoon environment for water. Irrigation decreases reliance on the monsoon, increases food security, and boosts agricultural production.Most research articles use humidity, moisture, and temperature sensors near the plant's root, with an external device handling all of the data provided by the sensors and transmitting it directly to an external display or an Android application. The application was created to measure the approximate values of temperature, humidity and moisture sensors that were programmed into a microcontroller to manage the amount of water.

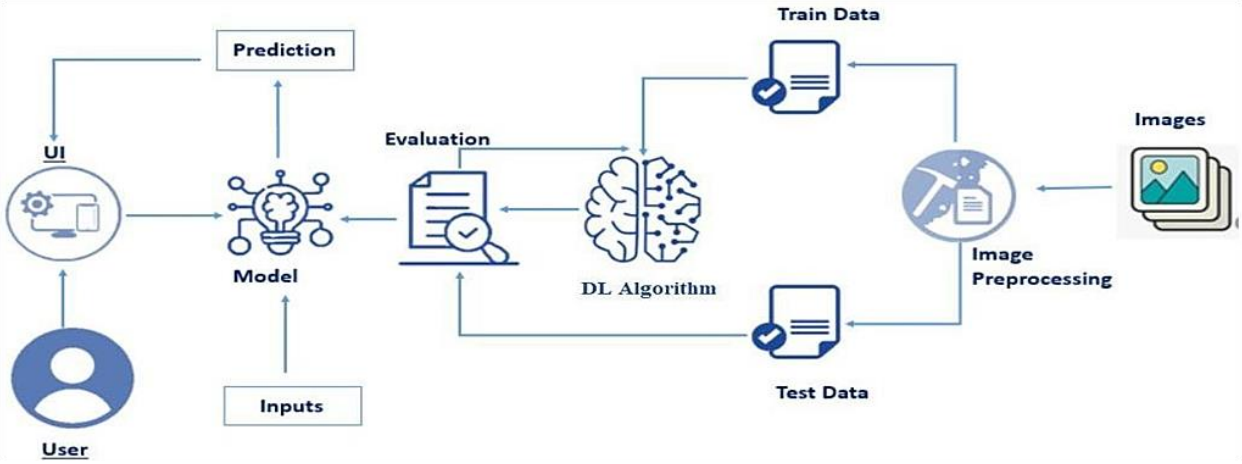
b. Proposed Solution:

So, we have built Web Application where:

1. Farmers interact with the portal build
2.Interacts with the user interface to upload images of diseased leaf
3.Our model built analyses the Disease and suggests the farmer with fertilizers are to be used

- Detection and recognition of plant diseases using machine learning are very efficient in providing symptoms of identifying diseases at its earliest.
- It recommends the fertilizer for affected leaves based on severity level.
- This web application makes the farmers to take right decision in selecting the fertilizer for crop disease such that agricultural sector will be developed by innovative idea.

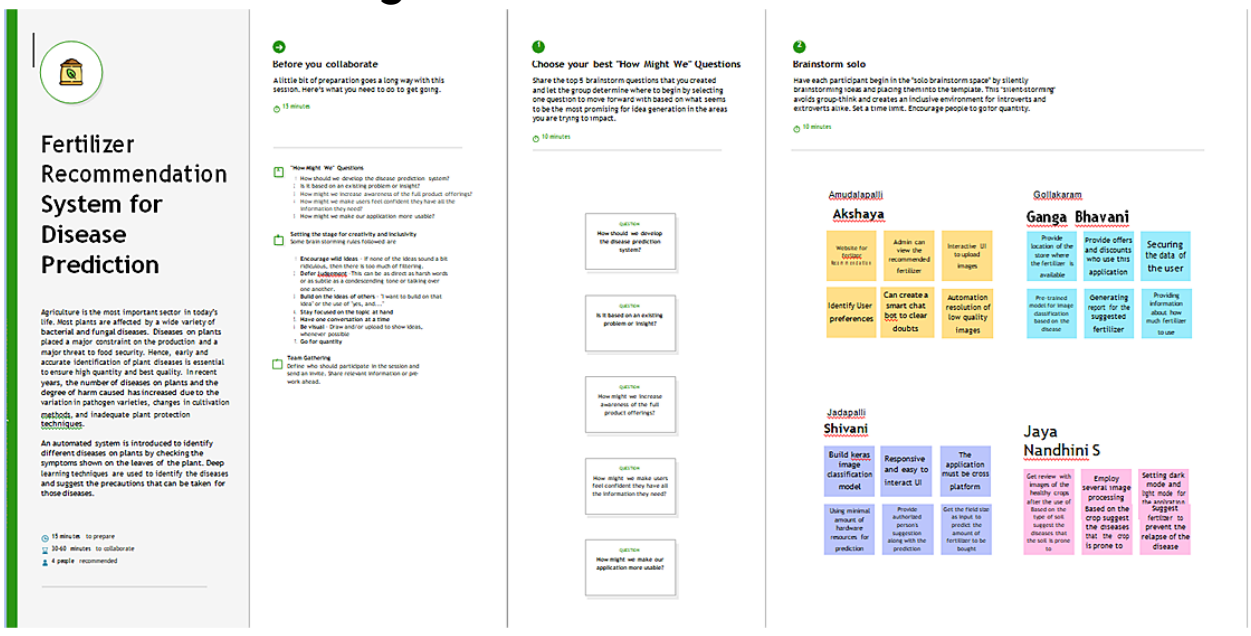
TECHNICAL ARCHITECTURE:

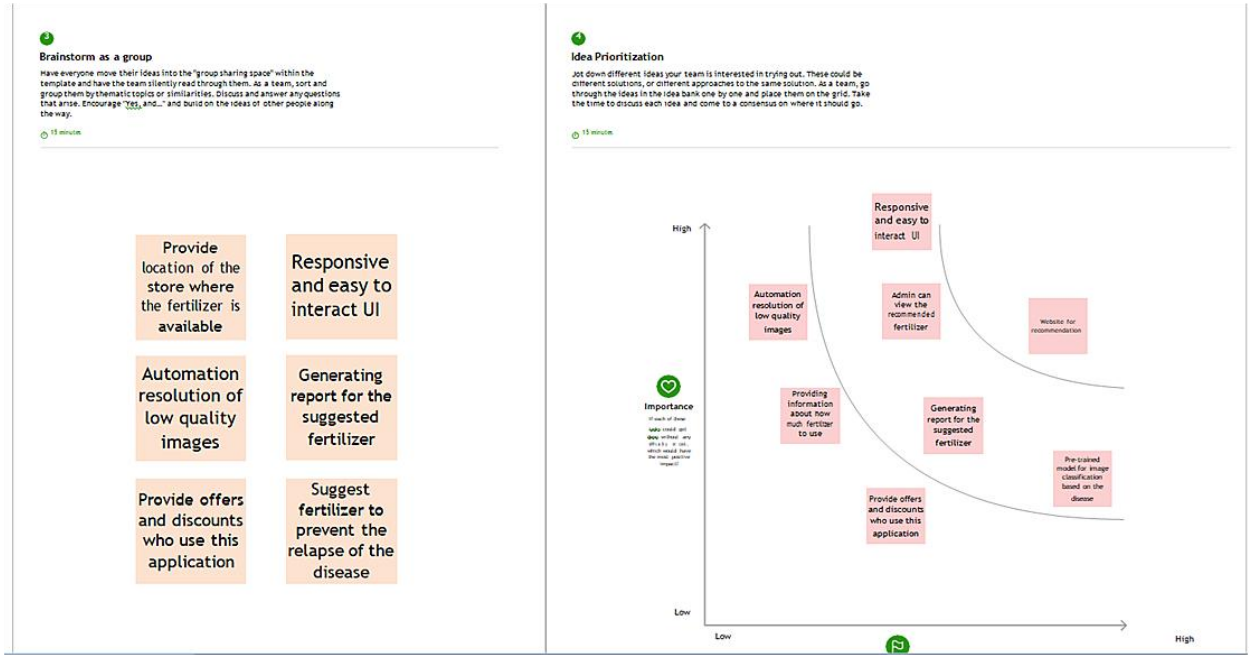


5. IDEATION & PROPOSED SOLUTION

a. Empathy Map Canvas

b. Brainstroming





c. Problem Solution Fit

PNT2022TMID23699	ProblemSolution-Fit	FertilizersRecommendationSystemforDiseasePrediction

1.CUSTOMERSEGMENT(S) — CS

- Farmersareourprimarycustomerstosolvetheirproblemchoosingrightfertilizers.
- Oursecondarycustomersaretheresearcherstomaketheirjobeasywithour AITechnology.
- Peoplewhocouldn'taffordforaConsultantforchoosingcropsandfertilizers.

6.CUSTOMERCONSTRAINTS — CC

- Thisisbasicallyawebapplication,WhichisSupportedinalmostalldevices.
- Theeasygraphicalrepresentationmakeaclearunderstandingforallpeople.
- TheResultsfortheirproblemwillbeinminute.

5. AVAILABLESOLUTIONS — AS

Which solutions are available to the customers when they face

- By using the AI will end up the existed problem , by provide results in low price.
- Its affordable by all people and the results are provided instantly
- Its Supports in Mobile, Desktop, etc (Almost all device support)

DefineCS,fitintoCC — ExploreAS,differentiate

2.JOBS-TO-BE-DONE/PROBLEMS — J&P

Which jobs-to-be-done (or problems) do you address for

- Itsprovidesagoodfertilizerrecommendationfortheircrops.
- Itsanalyzesthediseasewhichaffectstheirplants.
- Itsshowsasetofcropswhichsuitablefortheirsoilandtheirclimate.

9.PROBLEMROOTCAUSE — RC

What is the real reason that this problem

- Thetraditionalwayareexpensive.
- Farmerswantto getresultsinstantly.
- ToimproveProductioninlowcostandeasy.
- Traditionalwaynotcontainsaeasilyunderstandablegraphicalrepresentationofresults.

7.BEHAVIOUR — BE

- By using our product , they able to saves a lot ofmoneyspend fora expert.
- Itssavesatimeandmakes theirprocessfaster.
- Itimprovestheirfieldgrowthwithourproduct.
- Itensuresthecausespreviouslyandprovidesolutionsbeforethe damagehappens.

FocusonJ&P,tapintoBE,understandRC — FocusonJ&P,tapintoBE,understandRC

3.TRIGGERS — TR

- People will feel that our provides abunchofvaluableserviceaffordable.

4.EMOTIONS:BEFORE/AFTER — EM

- Itsreducesthefarmersunwanted Workload,stress,money,time,etc...

10. YOURSOLUTION — SL

- ByBuildingaAI,MLbasedwebapplicationmaketheirissues resolvedinseconds.
- Maketheirexpensiveprocessaffordable.
- MinimizetheTimeforanalyzetheirproblemandprovideresults inseconds .
- EasyGraphicalrepresentationmakesabetterunderstandingby everyone.

8.CHANNELSofBEHAVIOUR — CH

ONLINE

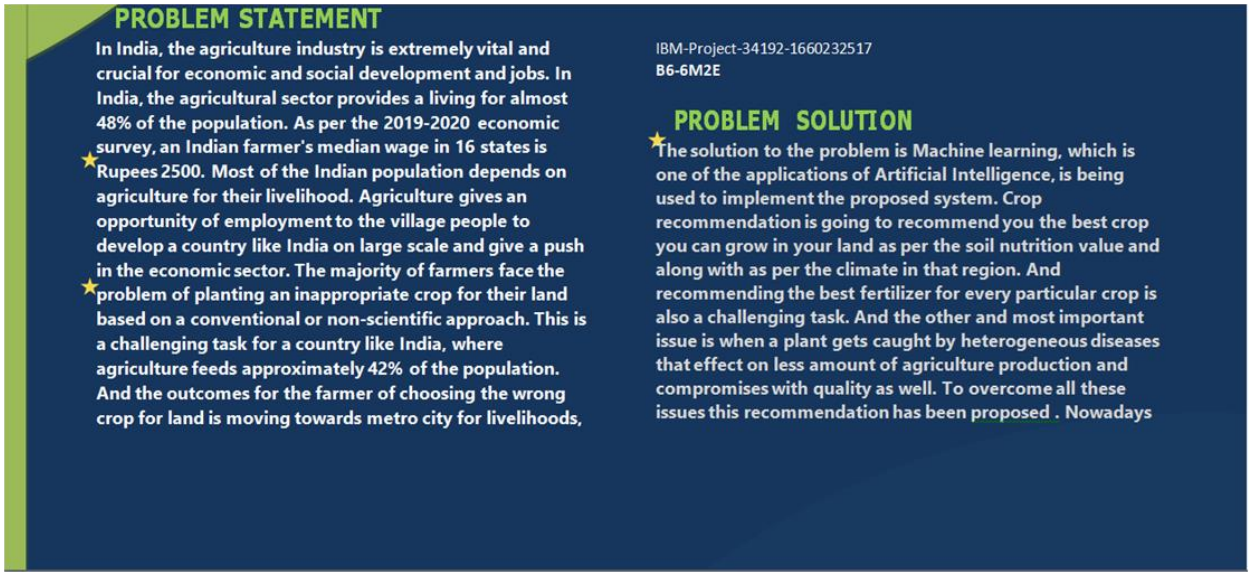
- Their Data analyzed early with help of cloud rendering

OFFLINE

- Its improves their crops production and reduces the losses.

Identifystrong TR&EM — Extractonline& offlineCHof BE

d. Proposed Solution



6. HARDWARE / SOFTWARE REQUIREMENTS:

To complete this project, you should have the following software and packages.

Software's:

- Anaconda Navigator
- pycharm
- Visual studio code
- Jupyter notebook
- IBM Watson studio

Packages:

- Tensor flow
- Keras
- Flask
- Numpy
- Pandas

By using the above listed software's and packages, we built this application to take the input as image from the Farmer and detects whether the plant is infected or not. Here we use Deep learning techniques and give the output to the user as Farmer.

7. 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

8. Flowchart:

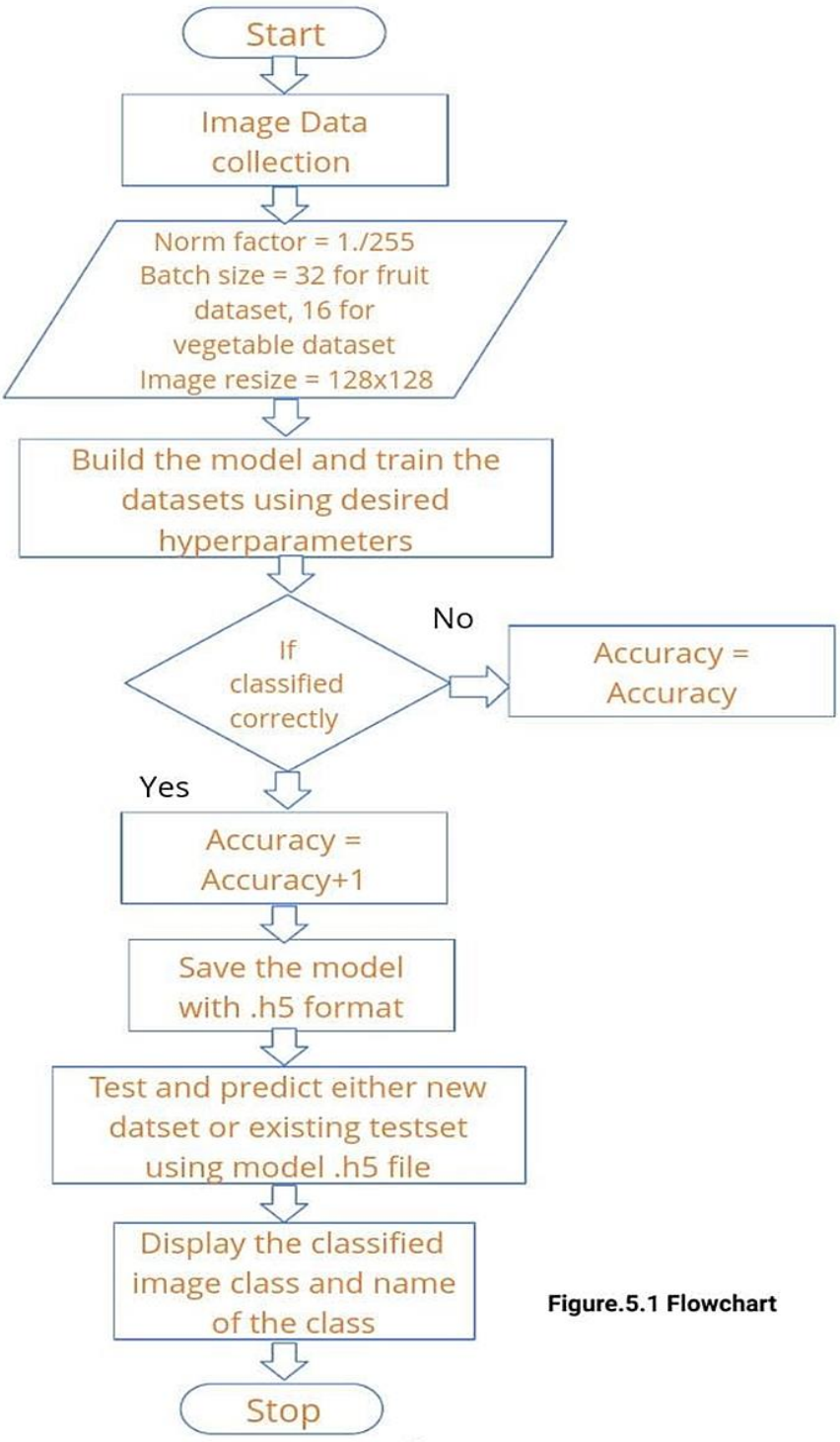
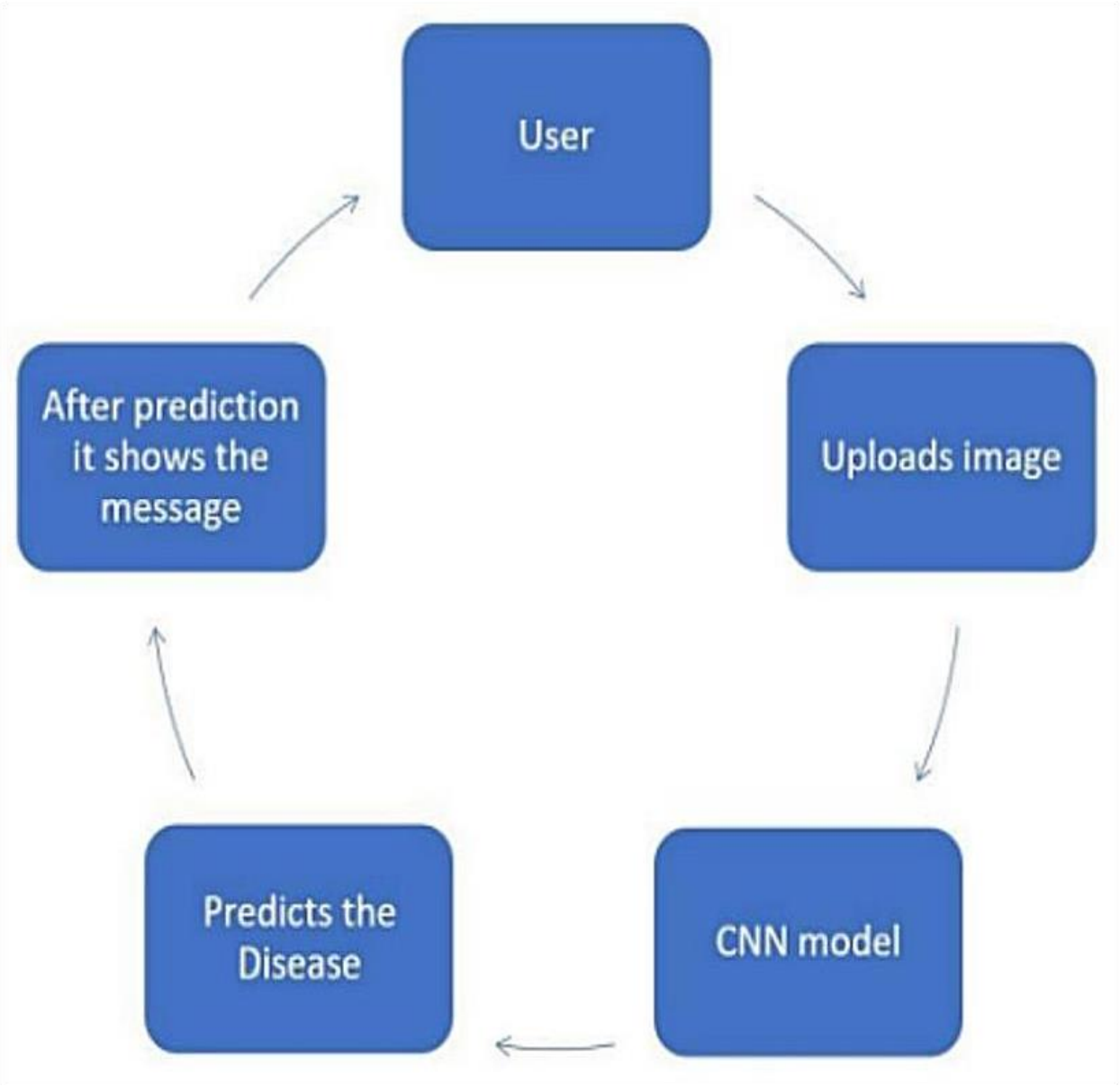


Figure.5.1 Flowchart



To accomplish the above task, you must complete the below activities and tasks:

- Download the dataset.
- Classify the dataset into train and test sets.
- Add the neural network layers.
- Load the trained images and fit the model.
- Test the model.
- Save the model and its dependencies.
- Build a Web application using a flask that integrates with the model built.

9. Coding & Solutioning:

Plant disease detection

```
import numpy as np
import pickle
import cv2
  from os import listdir
from sklearn.preprocessing import LabelBinarizer
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
  import tensorflow
EPOCHS = 25
INIT_LR = 1e-3
BS = 32
default_image_size = tuple((256, 256))
image_size = 0
directory_root = '../input/plantvillage/'
width=256
height=256
```

```
depth=3

def convert_image_to_array(image_dir):
    try:
        image = cv2.imread(image_dir)
        if image is not None :
            image = cv2.resize(image, default_image_size)
            return img_to_array(image)
        else :
return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None

image_list, label_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)
    for directory in root_dir :
        # remove .DS_Store from list
if directory == ".DS_Store" :
            root_dir.remove(directory)

    for plant_folder in root_dir :
        plant_disease_folder_list = listdir(f"{directory_root}/{plant_folder}")

        for disease_folder in plant_disease_folder_list :
            # remove .DS_Store from list
            if disease_folder == ".DS_Store" :
                plant_disease_folder_list.remove(disease_folder)
```

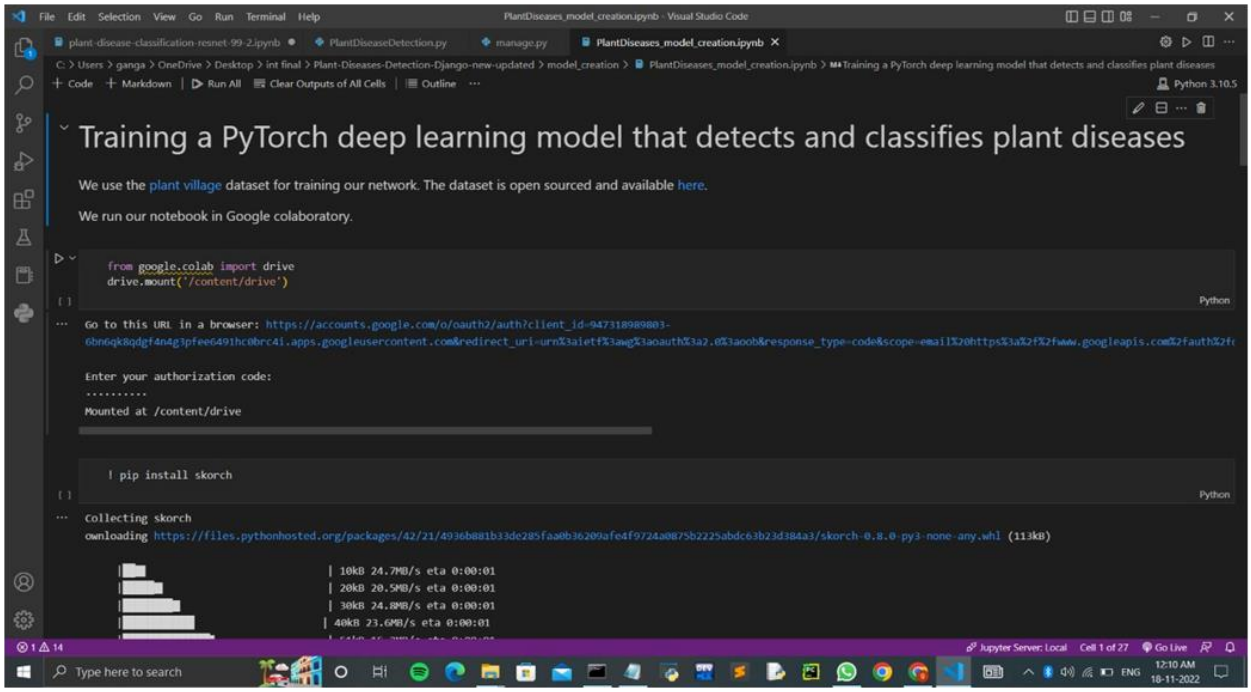
```
        for plant_disease_folder in plant_disease_folder_list:
            print(f"[INFO] Processing {plant_disease_folder} ...")
            plant_disease_image_list =
listdir(f"{directory_root}/{plant_folder}/{plant_disease_folder}/")

            for single_plant_disease_image in plant_disease_image_list :
                if single_plant_disease_image == ".DS_Store" :
                    plant_disease_image_list.remove(single_plant_disease_image)

            for image in plant_disease_image_list[:200]:
                image_directory =
f"{directory_root}/{plant_folder}/{plant_disease_folder}/{image}"
                if image_directory.endswith(".jpg") == True or
image_directory.endswith(".JPG") == True:
                    image_list.append(convert_image_to_array(image_directory))
                    label_list.append(plant_disease_folder)
    print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")

image_size = len(image_list)
label_binarizer = LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)
pickle.dump(label_binarizer,open('label_transform.pkl', 'wb'))
n_classes = len(label_binarizer.classes_)

print(label_binarizer.classes_)

np_image_list = np.array(image_list, dtype=np.float16) / 225.0
```

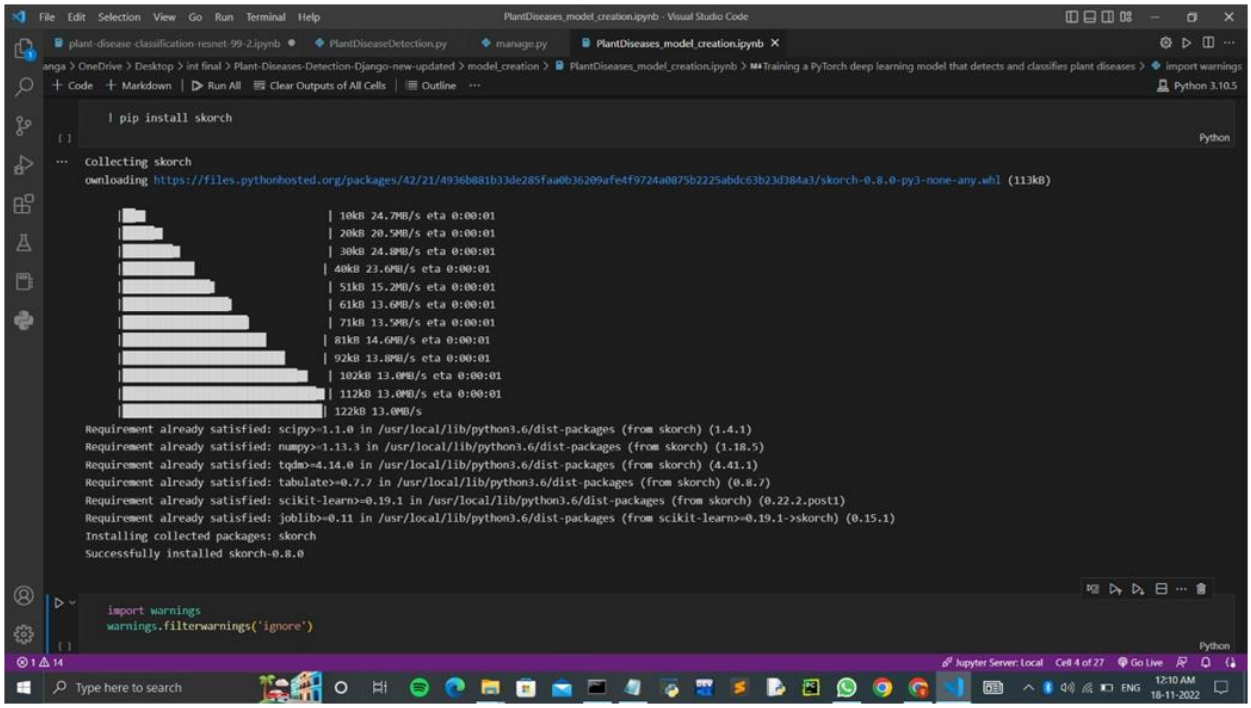
```
print("[INFO] Spliting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels,
test_size=0.2, random_state = 42)

aug = ImageDataGenerator(
    rotation_range=25, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2,
    zoom_range=0.2,horizontal_flip=True,
    fill_mode="nearest")

model = Sequential()
inputShape = (height, width, depth)
chanDim = -1
if K.image_data_format() == "channels_first":
    inputShape = (depth, height, width)
    chanDim = 1
model.add(Conv2D(32, (3, 3), padding="same",input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

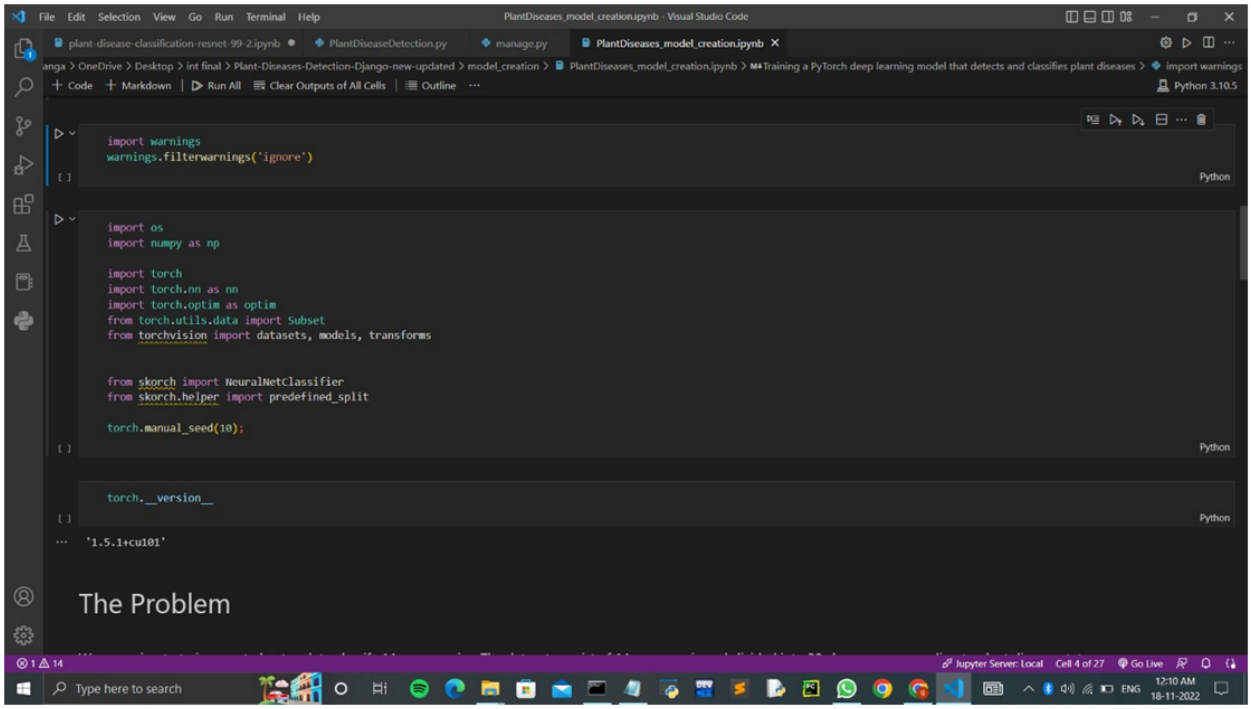
```
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(n_classes))
model.add(Activation("softmax"))
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
# distribution
model.compile(loss="binary_crossentropy", optimizer=opt,metrics=["accuracy"])
# train the network
print("[INFO] training network...")

history = model.fit_generator(
    aug.flow(x_train, y_train, batch_size=BS),
    validation_data=(x_test, y_test),
    steps_per_epoch=len(x_train) // BS,
    epochs=EPOCHS, verbose=1
    )

acc = history.history['acc']
val_acc = history.history['val_acc']
```

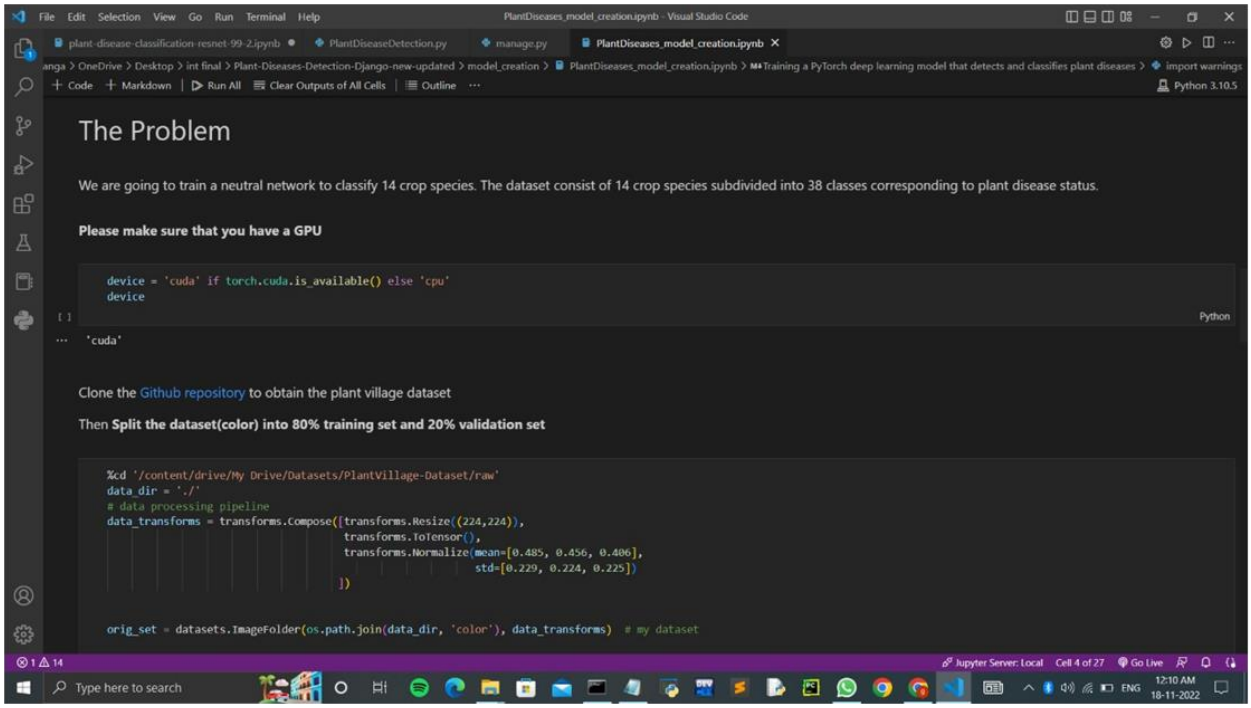
```
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
#Train and validation accuracy
plt.plot(epochs, acc, 'b', label='Training accurarcy')
plt.plot(epochs, val_acc, 'r', label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()

plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
print("[INFO] Calculating model accuracy")
scores = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")

# save the model to disk
print("[INFO] Saving model...")
pickle.dump(model,open('cnn_model.pkl', 'wb'))

#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

MANAGE
def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'plantvillage.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
```

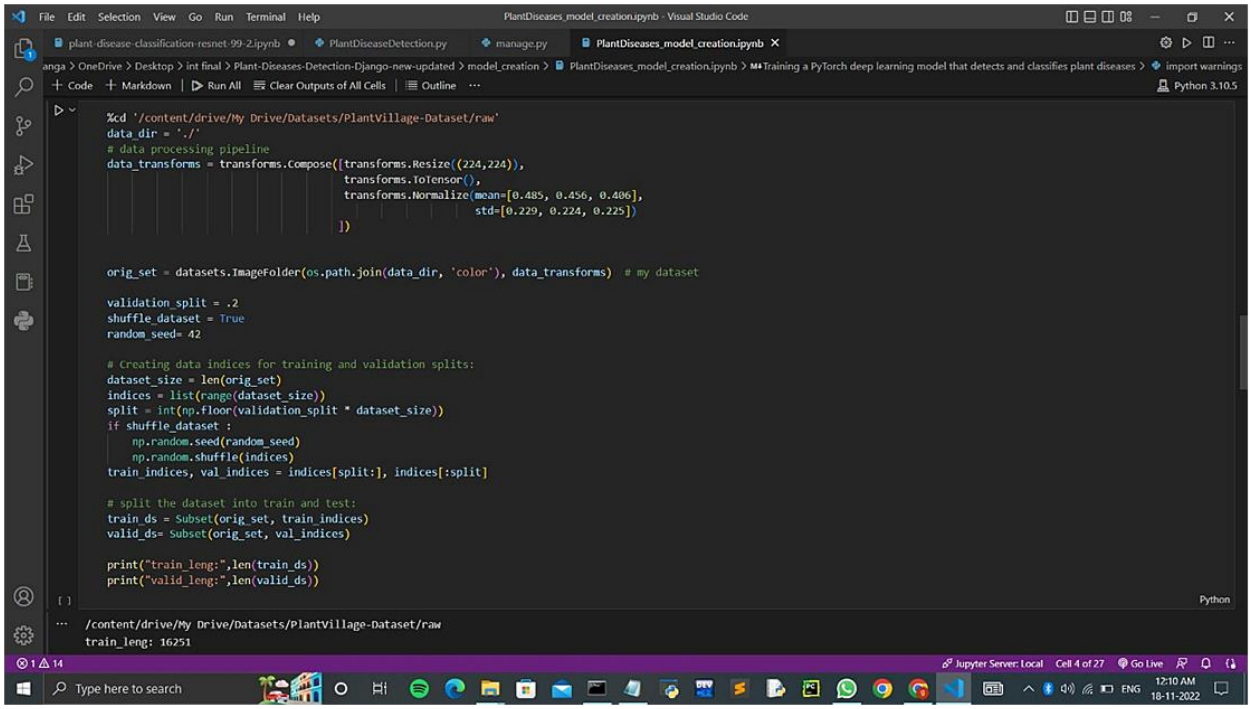
```
            "available on your PYTHONPATH environment variable? Did you "

            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if _name_ == '_main_':
    main()
```




File Edit Selection View Go Run Terminal Help
PlantDiseases_model_creation.ipynb - Visual Studio Code
plant-disease-classification-resnet-99-2.ipynb
PlantDiseaseDetection.py
manage.py
PlantDiseases_model_creation.ipynb
Code Markdown Run All Clear Outputs of All Cells Outline
Python 3.10.5
! pip install skorch
Python
Collecting skorch
Downloading https://files.pythonhosted.org/packages/42/21/4936b881b33de285faa0b36209afe4f9724a0875b2225abdc63b23d384a3/skorch-0.8.0-py3-none-any.whl (113kB)
| 10kB 24.7MB/s eta 0:00:01
| 20kB 20.5MB/s eta 0:00:01
| 30kB 24.8MB/s eta 0:00:01
| 40kB 23.6MB/s eta 0:00:01
| 51kB 15.2MB/s eta 0:00:01
| 61kB 13.6MB/s eta 0:00:01
| 71kB 13.5MB/s eta 0:00:01
| 81kB 14.6MB/s eta 0:00:01
| 92kB 13.8MB/s eta 0:00:01
| 102kB 13.0MB/s eta 0:00:01
| 112kB 13.0MB/s eta 0:00:01
| 122kB 13.0MB/s
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.6/dist-packages (from skorch) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.6/dist-packages (from skorch) (1.18.5)
Requirement already satisfied: tqdm>=4.14.0 in /usr/local/lib/python3.6/dist-packages (from skorch) (4.41.1)
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.6/dist-packages (from skorch) (0.8.7)
Requirement already satisfied: scikit-learn>=0.19.1 in /usr/local/lib/python3.6/dist-packages (from skorch) (0.22.2.post1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn>=0.19.1->skorch) (0.15.1)
Installing collected packages: skorch
Successfully installed skorch-0.8.0
import warnings
warnings.filterwarnings('ignore')
Python
Jupyter Server: Local Cell 4 of 27 Go Live
Type here to search
ENG 12:10 AM 18-11-2022

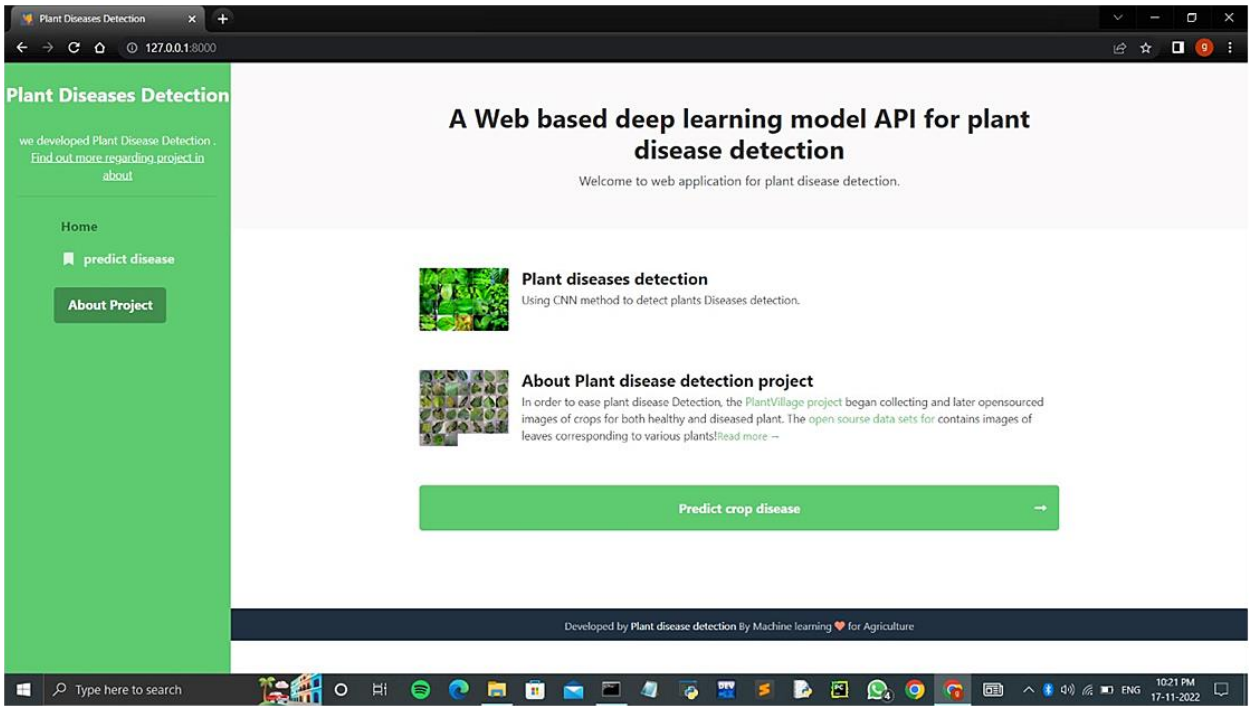

File Edit Selection View Go Run Terminal Help
PlantDiseases_model_creation.ipynb - Visual Studio Code
plant-disease-classification-resnet-99-2.ipynb
PlantDiseaseDetection.py
manage.py
PlantDiseases_model_creation.ipynb
Code Markdown Run All Clear Outputs of All Cells Outline
Python 3.10.5
import warnings
warnings.filterwarnings('ignore')
Python
import os
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Subset
from torchvision import datasets, models, transforms
from skorch import NeuralNetClassifier
from skorch.helper import predefined_split
torch.manual_seed(10);
Python
torch.__version__
Python
'1.5.1+cu101'
The Problem
Jupyter Server: Local Cell 4 of 27 Go Live
Type here to search
ENG 12:10 AM 18-11-2022

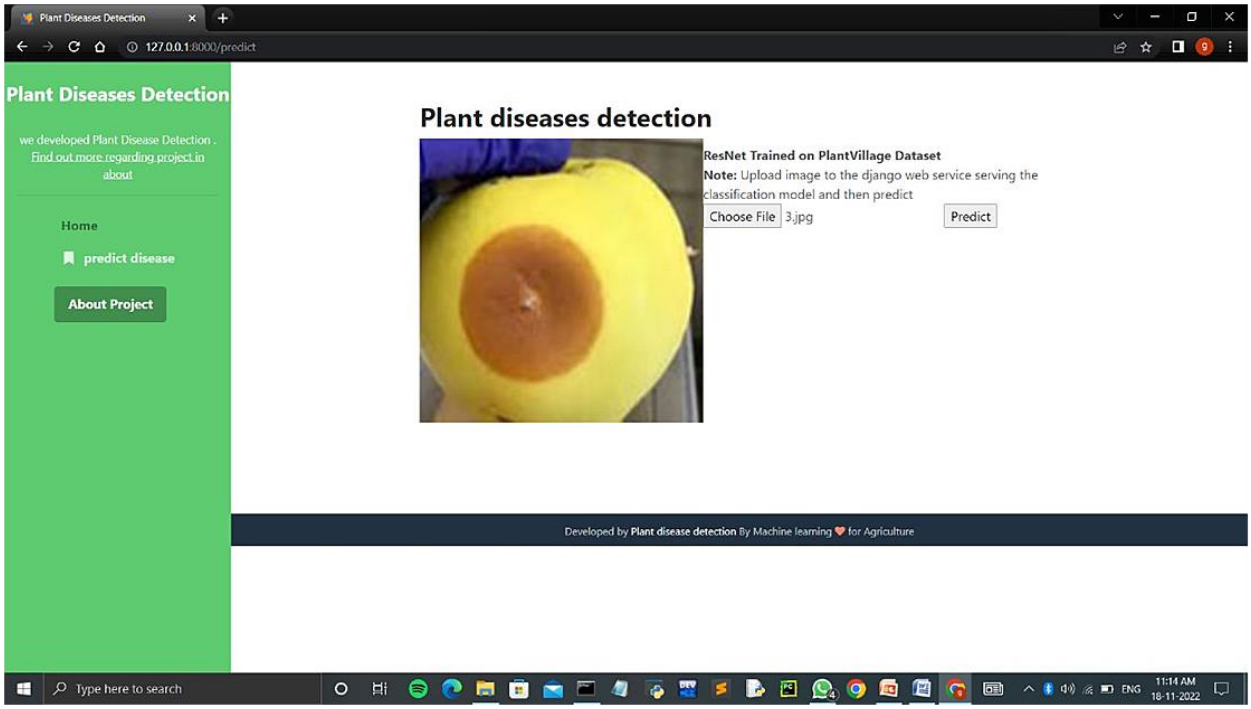

File Edit Selection View Go Run Terminal Help
PlantDiseases_model_creation.ipynb - Visual Studio Code
plant-disease-classification-resnet-99-2.ipynb
PlantDiseaseDetection.py
manage.py
PlantDiseases_model_creation.ipynb
Code Markdown Run All Clear Outputs of All Cells Outline
Python 3.10.5
The Problem
We are going to train a neutral network to classify 14 crop species. The dataset consist of 14 crop species subdivided into 38 classes corresponding to plant disease status.
Please make sure that you have a GPU
device = 'cuda' if torch.cuda.is_available() else 'cpu'
device
'cuda'
Clone the Github repository to obtain the plant village dataset
Then Split the dataset(color) into 80% training set and 20% validation set
%cd '/content/drive/My Drive/Datasets/PlantVillage-Dataset/raw'
data_dir = './'
# data processing pipeline
data_transforms = transforms.Compose([transforms.Resize((224,224)),
transforms.ToTensor(),
transforms.Normalize(mean=[0.485, 0.456, 0.406],
std=[0.229, 0.224, 0.225])
])
orig_set = datasets.ImageFolder(os.path.join(data_dir, 'color'), data_transforms) # my dataset
Jupyter Server: Local Cell 4 of 27 Go Live
Type here to search
ENG 12:10 AM 18-11-2022


File Edit Selection View Go Run Terminal Help
PlantDiseases_model_creation.ipynb - Visual Studio Code
plant-disease-classification-resnet-99-2.ipynb
PlantDiseaseDetection.py
manage.py
PlantDiseases_model_creation.ipynb
Code Markdown Run All Clear Outputs of All Cells Outline
Python 3.10.5
%cd '/content/drive/My Drive/Datasets/PlantVillage-Dataset/raw'
data_dir = './'
# data processing pipeline
data_transforms = transforms.Compose([transforms.Resize((224,224)),
transforms.ToTensor(),
transforms.Normalize(mean=[0.485, 0.456, 0.406],
std=[0.229, 0.224, 0.225])
])
orig_set = datasets.ImageFolder(os.path.join(data_dir, 'color'), data_transforms) # my dataset
validation_split = .2
shuffle_dataset = True
random_seed= 42
# Creating data indices for training and validation splits:
dataset_size = len(orig_set)
indices = list(range(dataset_size))
split = int(np.floor(validation_split * dataset_size))
if shuffle_dataset :
np.random.seed(random_seed)
np.random.shuffle(indices)
train_indices, val_indices = indices[split:], indices[:split]
# split the dataset into train and test:
train_ds = Subset(orig_set, train_indices)
valid_ds= Subset(orig_set, val_indices)
print("train_leng:",len(train_ds))
print("valid_leng:",len(valid_ds))
/content/drive/My Drive/Datasets/PlantVillage-Dataset/raw
train_leng: 16251
Jupyter Server: Local Cell 4 of 27 Go Live
Type here to search
ENG 12:10 AM 18-11-2022

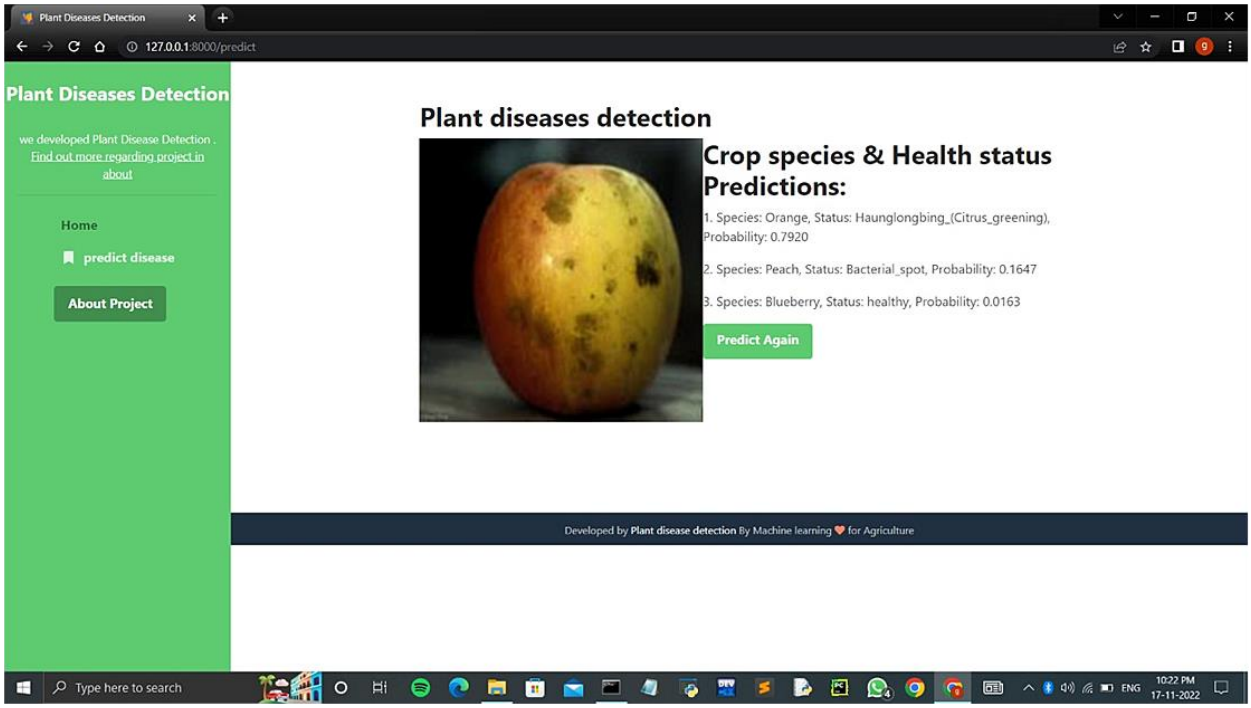
10. Result:

Home Page:

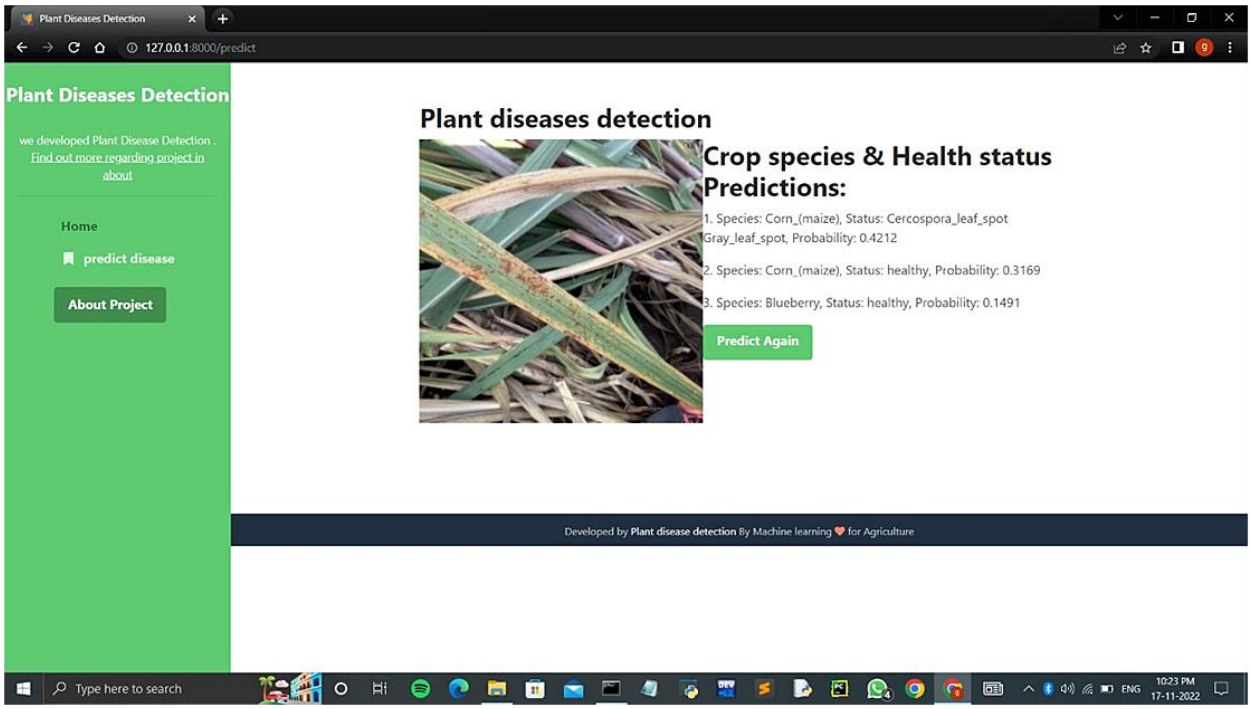




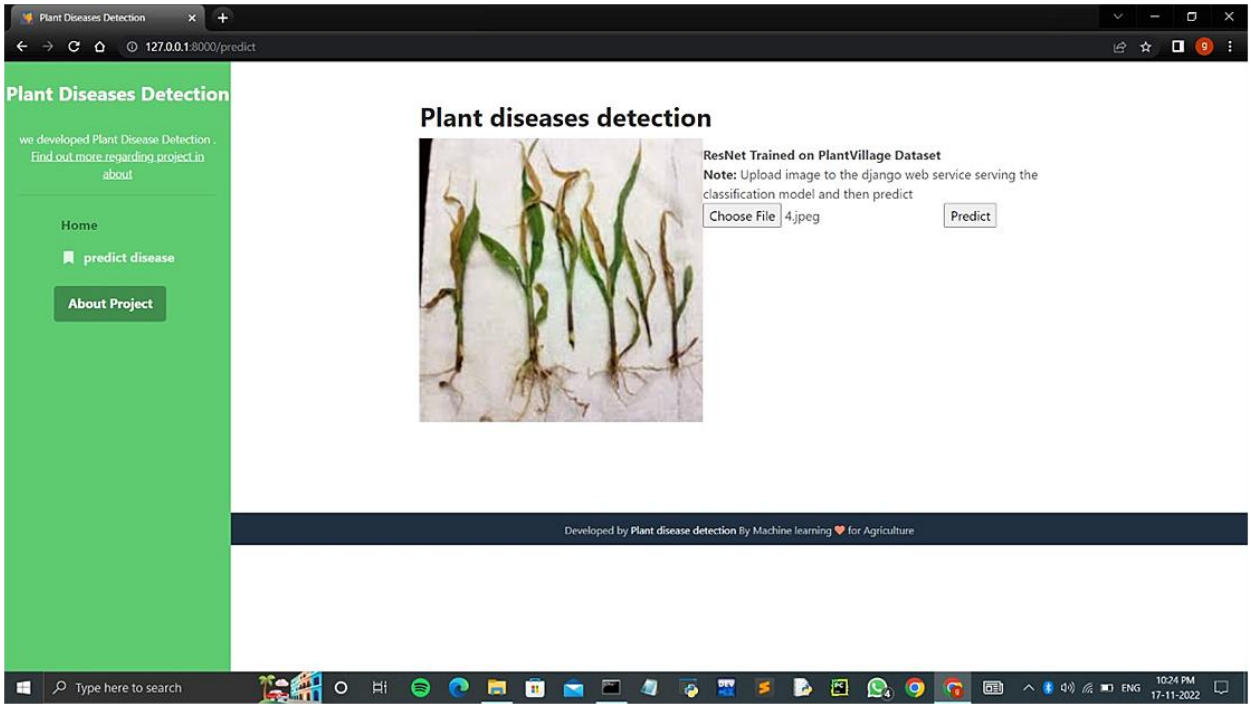
Train the fruit dataset



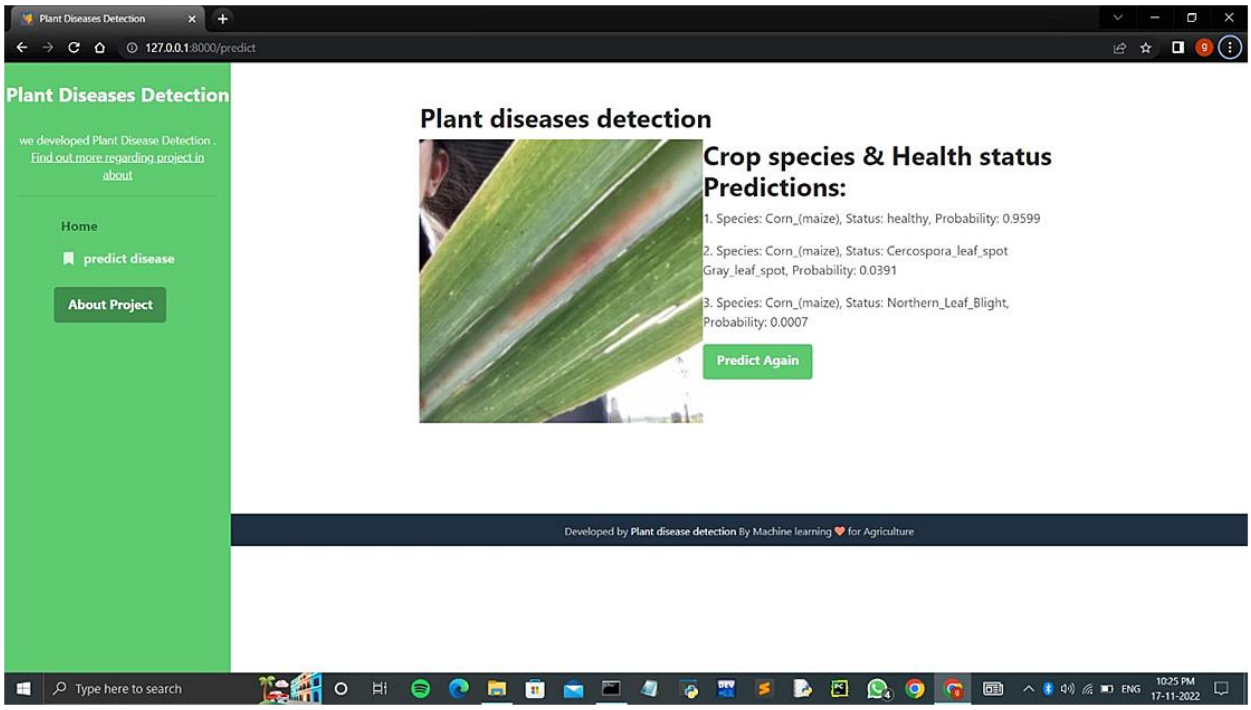
Test the fruit dataset



Test the leaf dataset



Train the leaf dataset



Test the leaf dataset

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 are very high can be resized within the proposed itself.

List of Disadvantages:

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

APPLICATIONS:

- This web application can be used by farmers or users to check whether their plant is infected or not and can also show the remedy so that the user can take necessary precautions.
- These kind of web applications can be used in the agricultural sector as well as for small house hold plants as well

11. CONCLUSION:

- 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. Usage of such applications could help the farmers to necessary precautions so that they don't face any loss as such.

12. FUTURE SCOPE:

- As of now we have just built the web application which apparently takes the input as an image and then predict the out in the near future we can develop an application which computer vision and AI techniques to predict the infection once you keep the camera near the plant or leaf this could make our project even more usable.

13. REFERENCE:

- Fertilizers Recommendation System For Disease Prediction In Tree Leave | Semantic Scholar
- Soil Based Fertilizer Recommendation System for Crop Disease Prediction System (ijetajournal.org)
- Leaf Disease Detection and Fertilizer Suggestion | IEEE Conference Publication | IEEE Xplore
- IRJET-V7I1004.pdf
- A nutrient recommendation system for soil fertilization based on evolutionary computation - ScienceDirect
- Fertilizers-Recommendation-System-For-Disease-Prediction-In-Tree-Leave.pdf (ijstr.org)
- 2204.11340.pdf (arxiv.org)
- 371-376,Tesma405,IJEAST.pdf
- CROFED - Crop and Fertilizer Recommendation and Disease diagnosis system using Machine Learning and Internet of Things. (ijirt.org)
- Prediction of Crop, Fertilizer and Disease Detection for Precision Agriculture by IJRASET - Issuu