FERTILIZERS RECOMMENDATION SYSTEM FOR DISEASE PREDICTION

1. INTRODUCTION

1.1 Overview

Agriculture is the most important and plays a major role in producing the food and impacts the economy in the country and contribute 17%of the GDP and from the overall population 60%of the people gets employment.in olden days' agriculture is the main economy for all the families. the formers like to farming various types of crops in their land and dedicatedly works to produce the good quality of products.

The good crop is depending on the various components like soil quality, best crop selection, rice disease prediction and natural disaster prediction [1]. Now days, it is very difficult to produce best quality crops due to various issues in farming like insufficient nutrients in land, un healthy seeding and etc., lot of diseases are affected to the plants.

We can improve the quality of farming by early prediction of diseases by soil sample testing [3], predicting from the leaf of the plant for various vegetables and fruits [2]. we can use the deep learning models to predict the diseases on plants like CNN model as it works well for image classification [4], Can apply "convnents" to identify the diseases[5].

1.2 Purpose

The main objective is developing a web application where the farmers can find the diseases form the leaf. we use CNN on the dataset and for predicting we use Deep learning techniques. After predicting the disease the farmer can understand the type of disease and get the suggestions on the submitted crop.

2. LITERATURE SURVEY

2.1 Deep learning model:

The proposing of deep learning model for accurate classification from plant leaf pictures, tomato leaf pictures were taken from the data set plant village and trained and find the accuracy[5], apply convnets for plant disease detection and classification[4].to identify and monitor a detection system is implemented by using tensorflow[6].In paper[8] says with the KNN the prediction is 98.56% accuracy for detecting the disease from the leaf.

Jetson Nano kit help full for faster and efficient processing in detecting the disease in the plant[9], in[10] review the deep learning methods and transfer learning and CNN for diseases identification in crops.

The segmentation and feature extraction and classification methods were used by applying with the healthy and diseased leaves [11].

2.2Existing System

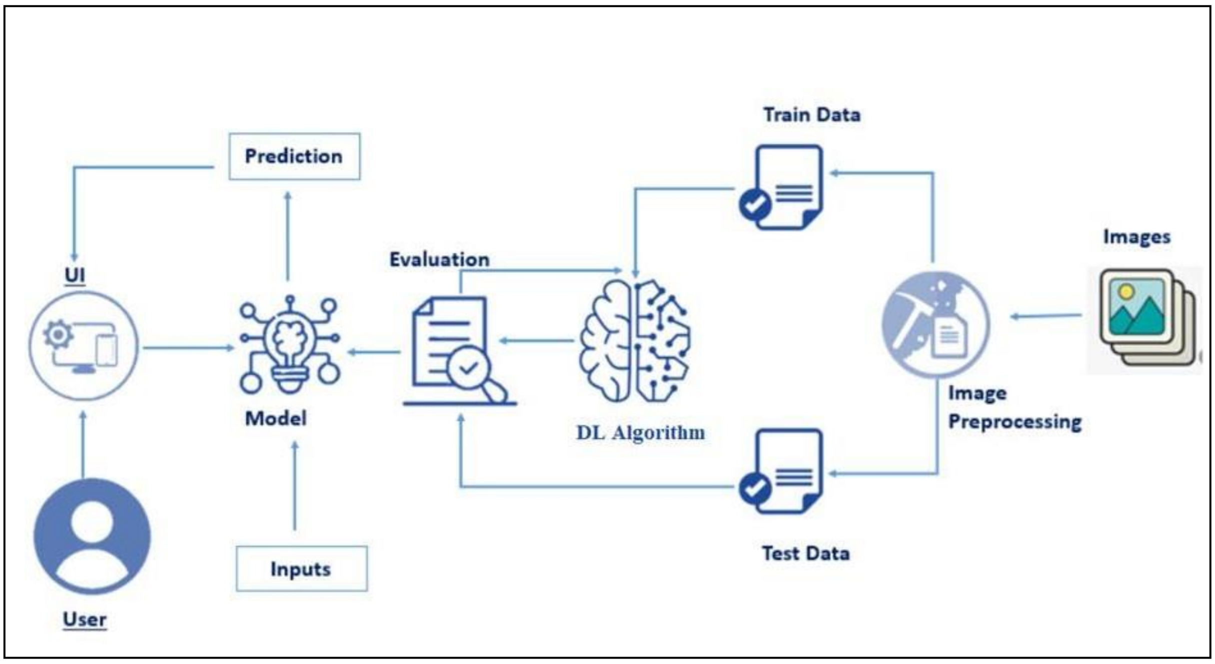
In most of the existing methods, the process of finding the soil type, identifying the leaf disease and preferring the fertilizer were all carried out manually. The method was prone to various disadvantages. Even when the framework was digitalized, it has certain problems as, predicting a diverse fertilizer for a soil type, certain files regarding the leaf disease or soil type or fertilizer may not be updated. In other situation the system may not provide the needed support. Hence to overcome some of these issues, this project proposed a new approach using AI.

2.3 Proposed System

An automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant. Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases.

3 THEORITICAL ANALYSIS

3.1 Technical Architecture



3.2 Hardware / Software designing

Hardware Requirements:

Processor: Intel i3 10 Gen or higher

Ram: 8 GB

HDD: 10 GB

GPU: Intel Graphics or NVIDIA or AMD based

Software Requirements:

OS: Windows 7 or Higher

Application: Anaconda Navigator, Jupyter Notebook, Spyder

4. IMPLEMENTATION

4.1 Project Flow

A web Application is built where

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

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.

4.2 Data Collection

The first step is to download the dataset. Create Train and Test folders with each folder having subfolders with leaf images of different plant diseases. You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc. From the dataset link you can download datasets that can be used for training. Two datasets will be used, we will be creating two models one to detect vegetable leaf diseases like tomato, potato, and pepper plants and the second model would be for fruits diseases like corn, peach, and apple.

4.3 Image Pre-processing

Now that we have all the data collected, let us use this data to train the model . before training the model you have to preprocess the images and then feed them on to the model for training. we make use of Keras ImageDataGenerator class for image preprocessing.

Image Pre-processing includes the following main tasks

- Import ImageDataGenerator Library.
- Configure ImageDataGenerator Class.

- Applying ImageDataGenerator functionality to the trainset and test set.

Note: The ImageDataGenerator accepts the original data, randomly transforms it, and returns only the new, transformed data.

Lets build model for fruit leaf disease detection

Open Jupyter notebook and create a new python file, name ii Fruit-Training.ipynb and save it in the project folder. To know more about the usage of the Jupyter notebook watch the video given in the pre-requisites section

Pre-process The Images

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.

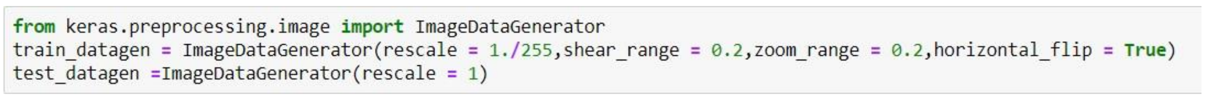
The first step is usually importing the libraries that will be needed in the program.

Import Keras library from that library import the ImageDataGenerator Library to your Python script:

Let us import the ImageDataGenerator class from Keras

Import ImageDataGenerator Library and Configure it

ImageDataGenerator class is used to load the images with different modifications like considering the zoomed image, flipping the image and rescaling the images to range of 0 and 1.



```
from 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)
```

There are five main types of data augmentation techniques for image data; specifically:

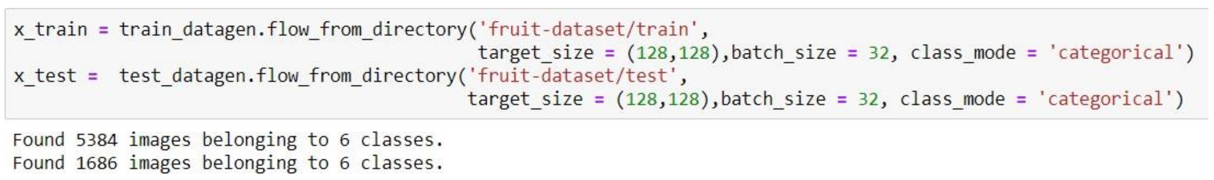
- Image shifts via the width_shift_range and height_shift_range arguments.

- The image flips via the horizontal_flip and vertical_flip arguments.
- The image rotates via the rotation_range argument
- Image brightness via the brightness_range argument.
- The image zooms via the zoom_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.

Apply ImageDataGenerator functionality to Train and Test set

Specify the path of both the folders in the flow_from_directory method. We are importing the images in 128*128 pixels.



```
x_train = train_datagen.flow_from_directory('fruit-dataset/train',
                                            target_size = (128,128),batch_size = 32, class_mode = 'categorical')
x_test =  test_datagen.flow_from_directory('fruit-dataset/test',
                                           target_size = (128,128),batch_size = 32, class_mode = 'categorical')
```

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

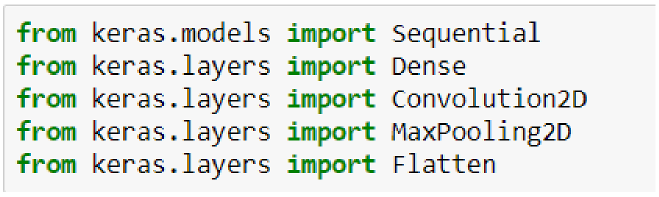
4.4 Model Building for Fruit Disease Prediction

We are ready with the augmented and pre-processed image data, Let’s begin our model building, this activity includes the following steps

- Import the model building Libraries
- Initializing the model
- Adding CNN Layers
- Adding Hidden Layer
- Adding Output Layer
- Configure the Learning Process
- Training and testing the model
- Saving the model

Import The Libraries

Import the libraries that are required to initialize the neural network layer and create and add different layers to the neural network model.



```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
```

Initializing The Model

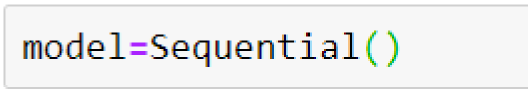
Keras has 2 ways to define a neural network:

- Sequential
- Function API

The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add () method.

Now, will initialize our model.

Initialize the neural network layer by creating a reference/object to the Sequential class.



```
model=Sequential()
```

ADD CNNLayers

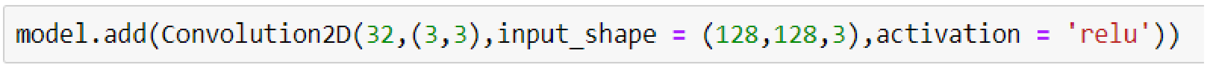
We will be adding three layers for CNN

- Convolution layer
- Pooling layer
- Flattening layer

Add Convolution Layer

The first layer of the neural network model, the convolution layer will be added. To create a convolution layer, Convolution2D class is used. It takes several feature detectors, feature detector size, expected input shape of the image, and activation function as arguments. This layer applies feature detectors on the input image and returns a feature map (features from the image).

Activation Function: These are the functions that help us to decide if we need to activate the node or not. These functions introduce non-linearity in the networks.



```
model.add(Convolution2D(32,(3,3),input_shape = (128,128,3),activation = 'relu'))
```

Add the pooling layer

Add Dense Layers

The name suggests that layers are fully connected (dense) by the neurons in a network layer. Each neuron in a layer receives input from all the neurons present in the previous layer. Dense is used to add the layers.

Adding Hidden layers

This step is to add a dense layer (hidden layer). We flatten the feature map and convert it into a vector or single dimensional array in the Flatten layer. This vector array is fed it as an input to the neural network and applies an activation function, such as sigmoid or other, and returns the output.

- **init** is the weight initialization; function which sets all the weights and biases of a network to values suitable as a starting point for training.
- units/ output_dim, which denote is the number of neurons in the hidden layer.
- The activation function basically decides to deactivate neurons or activate them to get the desired output. It also performs a nonlinear transformation on the input to get better results on a complex neural network.
- You can add many hidden layers, in our project we are added two hidden layers. The 1st hidden layer with 40 neurons and 2nd hidden layer with 20neurons.

Adding the output layer

This step is to add a dense layer (output layer) where you will be specifying the number of classes your dependent variable has, activation function, and weight initializer as the arguments. We use the add () method to add dense layers. the output dimensions here is 6

```
model.add(Dense(output_dim = 40 ,init = 'uniform',activation = 'relu'))
model.add(Dense(output_dim = 20 ,init = 'random_uniform',activation = 'relu'))
model.add(Dense(output_dim = 6,activation = 'softmax',init ='random_uniform'))
```

4.5 Train and Save The Model

Compile the model

After adding all the required layers, the model is to be compiled. For this step, loss function, optimizer and metrics for evaluation can be passed as arguments.

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

Fit and save the model

Fit the neural network model with the train and test set, number of epochs and validation steps. Steps per epoch is determined by number of training images//batch size, for validation steps number of validation images//batch size.

```
model.fit_generator(x_train, steps_per_epoch = 168,epochs = 3,validation_data = x_test,validation_steps = 52)
```

Accuracy, Loss: Loss value implies how poorly or well a model behaves after each iteration of optimization. An accuracy metric is used to measure the algorithm's performance in an interpretable way. The accuracy of a model is usually determined after the model parameters and is calculated in the form of a percentage.

The weights are to be saved for future use. The weights are saved in as .h5 file using save().

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

model.summary() can be used to see all parameters and shapes in each layer in our models.

4.6 Model Building For Vegetable Disease Prediction

Create an other jupyter notebook file in the project folder and name it as vegitable_training. The same steps followed for the fruit disease prediction model are to be followed to train the tomato, potato, and pepper diseases

4.7 Test Both The Models

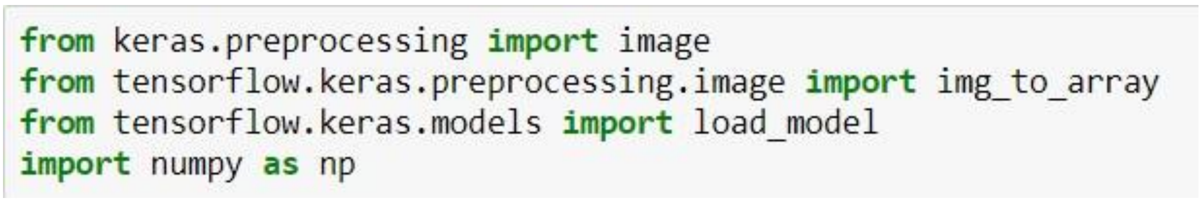
Now that we have trained both the models' let's test both the models by loading the saved models. let's create another notebook for testing

Test The Model

The model is to be tested with different images to know if it is working correctly.

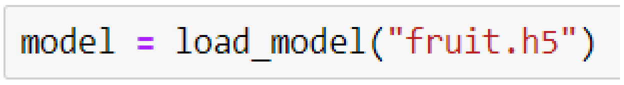
Import the packages and load the saved model

Import the required libraries



```python
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
```

Initially, we will be loading the fruit model. You can test it with the vegetable model in a similar way.



```python
model = load_model("fruit.h5")
```

Load the test image, pre-process it and predict

Pre-processing the image includes converting the image to array and resizing according to the model. Give the pre-processed image to the model to know to which class your model belongs to.

```
img = image.load_img('apple_healthy.JPG',target_size = (128,128))
```

```
x  = image.img_to_array(img)
x = np.expand_dims(x,axis = 0)
```

```
pred = model.predict_classes(x)
```

```
pred
```

```
[1]
```

The predicted class is 1

4.8 Application Building

After the model is built, we will be integrating it into a web application so that normal users can also use it. The new users need to initially register in the portal. After registration users can log in to browse the images to detect the disease.

In this section, you have to build

- HTML pages - front end
- Python script - Server-side script

Let's create a Python script using spyder IDE

Build Python Code

After the model is built, we will be integrating it into a web application so that normal users can also use it. The user needs to browse the images to detect the disease.

Activity 1: Build a flask application

Step 1: Load the required packages

Step 2: Initialize the flask app and load the model

An instance of Flask is created and the model is loaded using load_model from Keras.

Step 3: Configure the home page

Step 4: Pre-process the frame and run

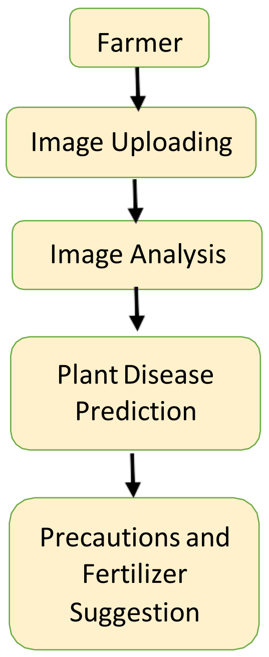
Pre-process the captured frame and give it to the model for prediction. Based on the prediction the output text is generated and sent to the HTML to display. We will be loading the precautions for fruits and vegetables excel file to get the precautions based on the output and return it to the HTML Page.

Run the flask application using the run method. By default, the flask runs on 5000 port. If the port is to be changed, an argument can be passed and the port can be modified.

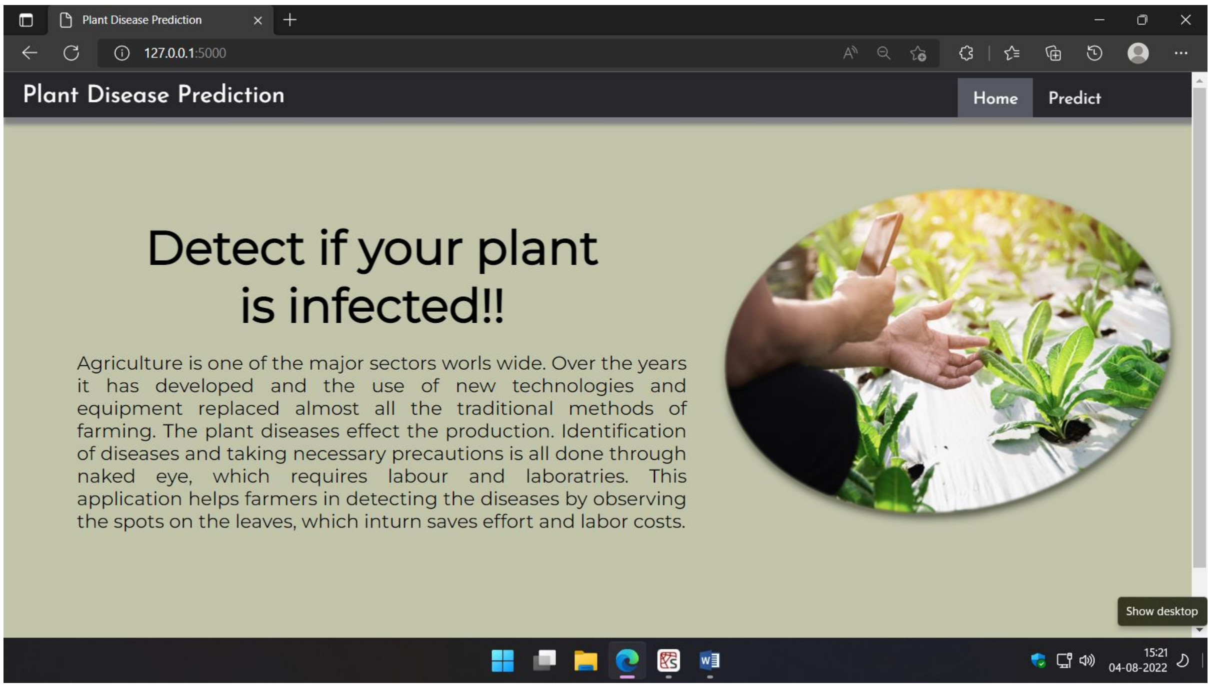
Build HTML Pages

Build the UI where a home page will have details about the application, a prediction page where a user is allowed to browse an image and get the predictions.

5. FLOWCHART



6. RESULTS



Plant Disease Prediction

127.0.0.1:5000

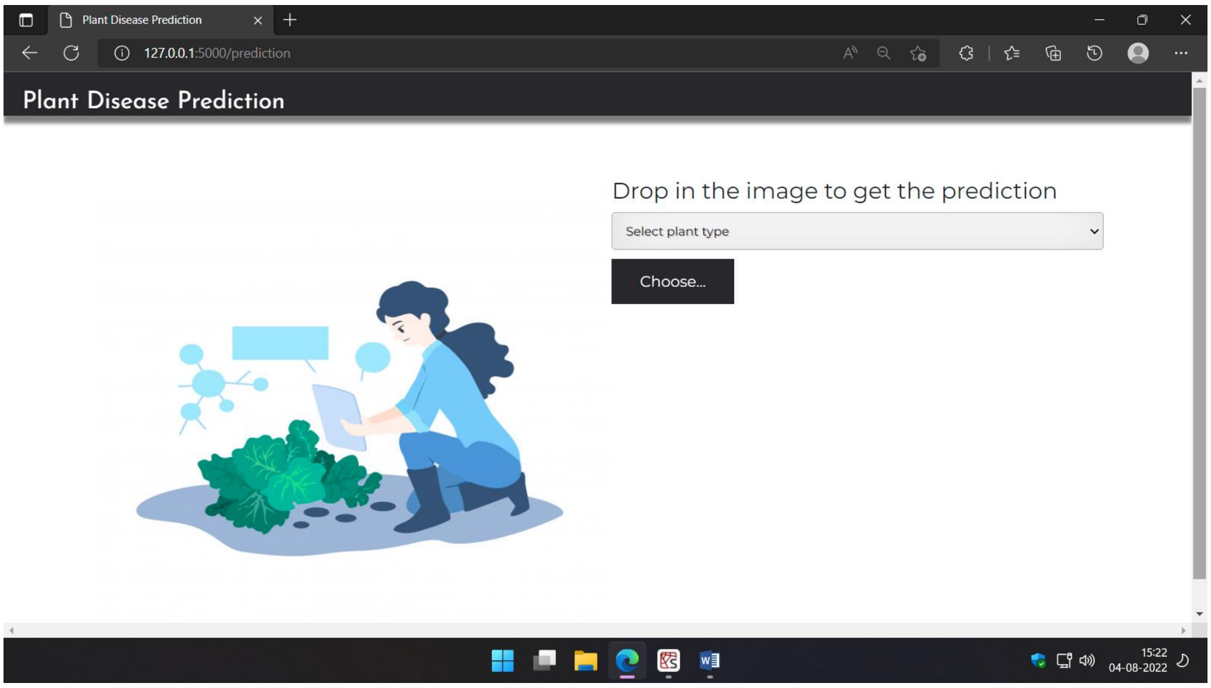
Plant Disease Prediction

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.

Fig 1: Home page



Plant Disease Prediction

127.0.0.1:5000/prediction

Plant Disease Prediction

Drop in the image to get the prediction

Select plant type

Choose...

Fig 2: Plant disease prediction page

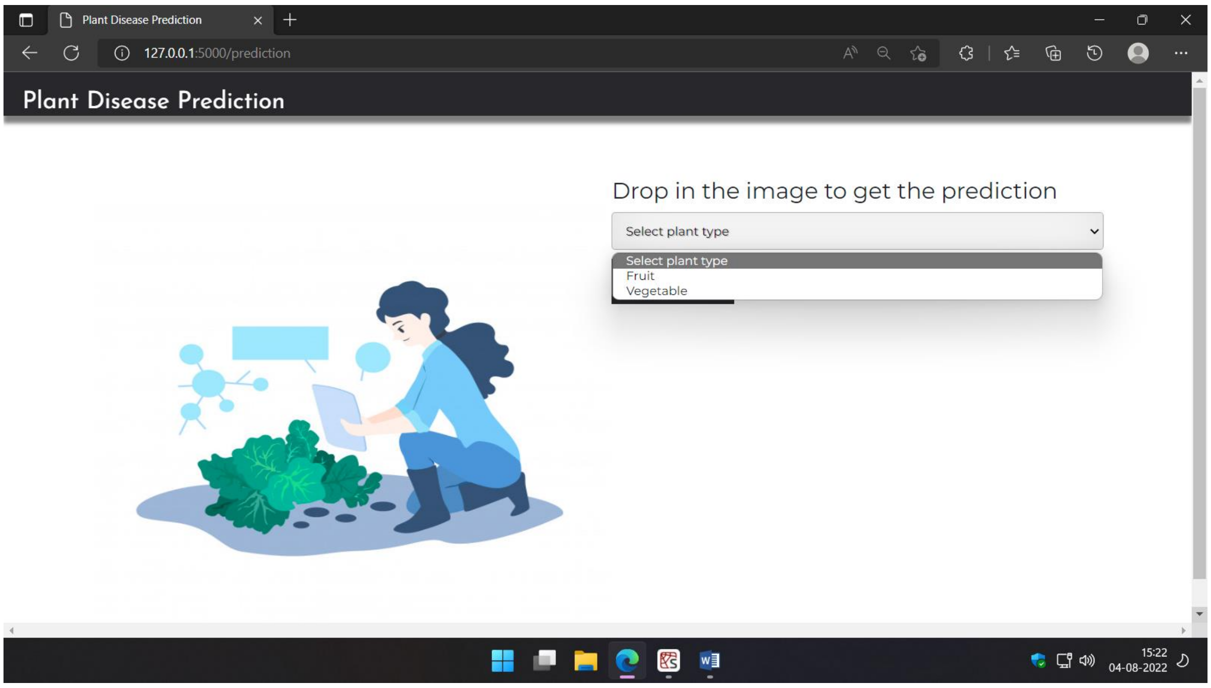
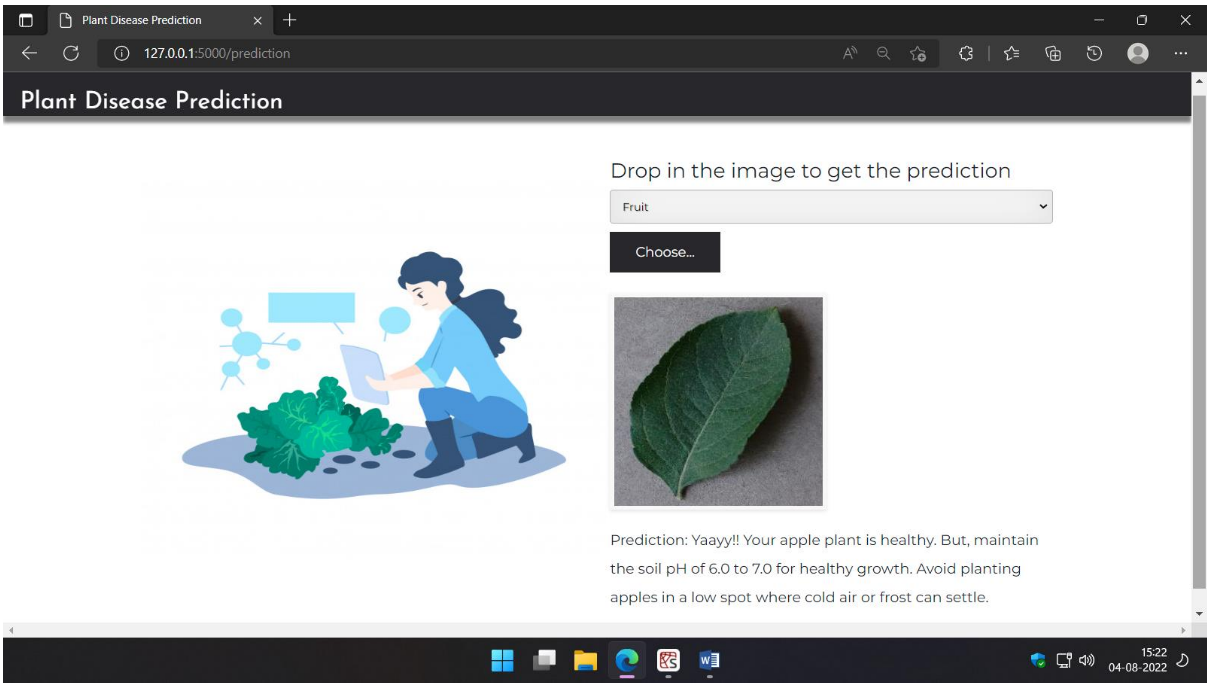
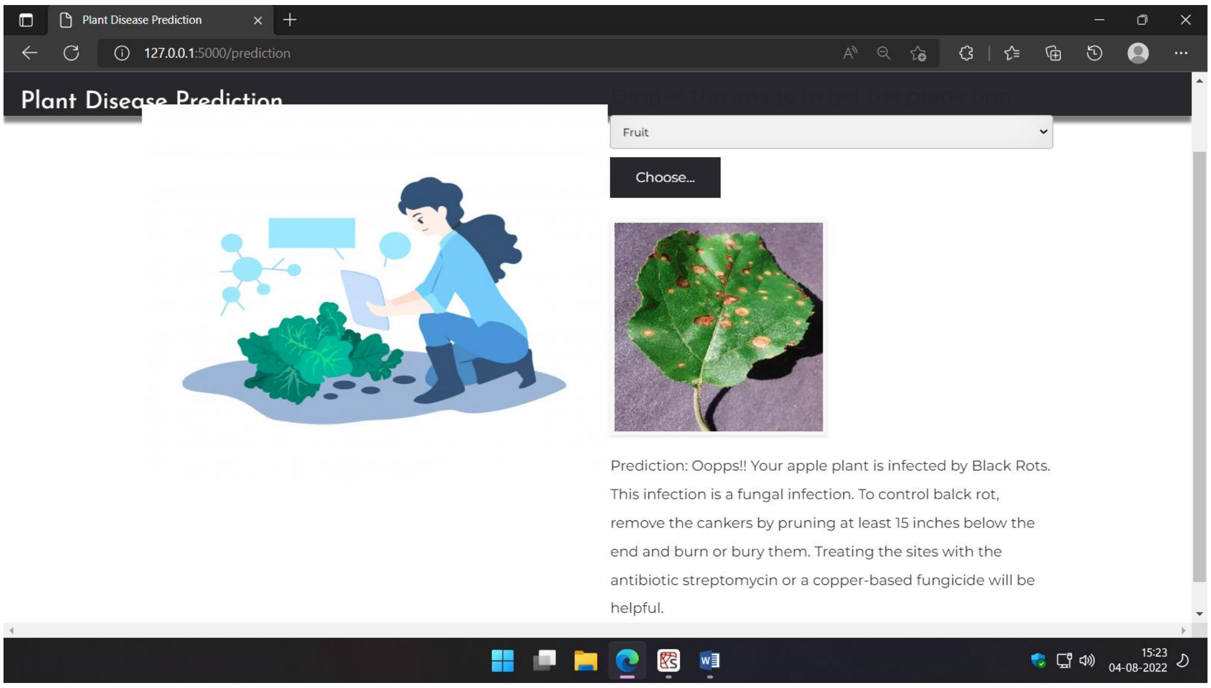


Fig 3: image selection page



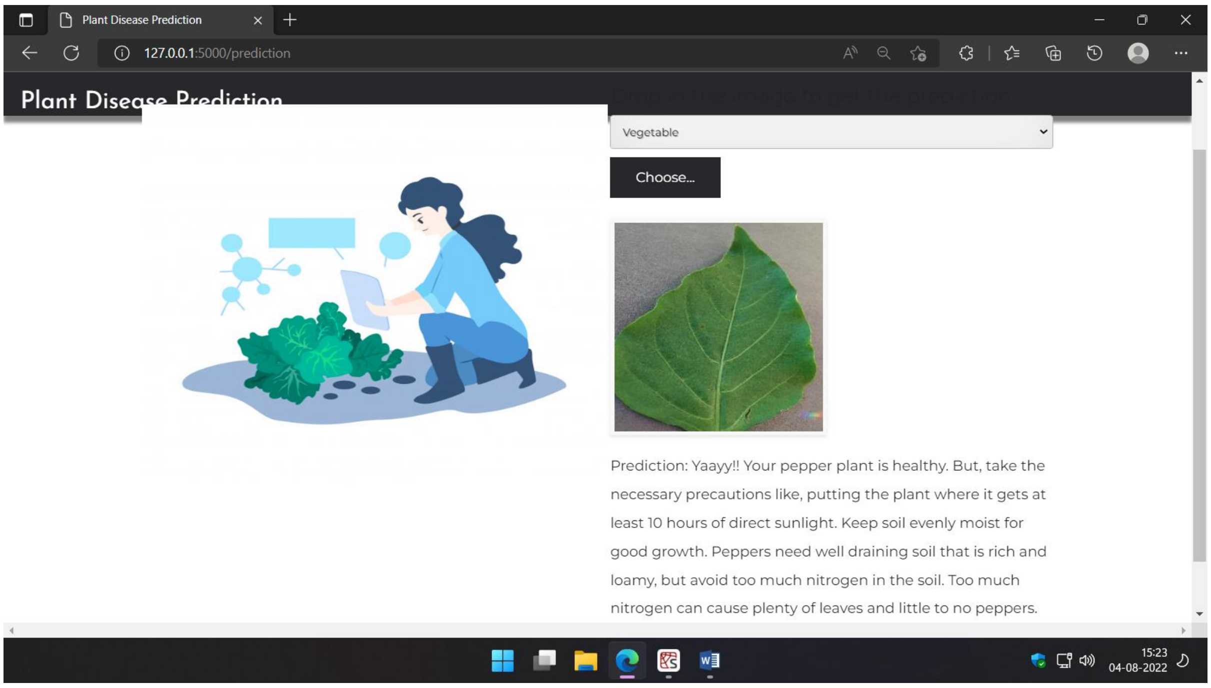
Prediction: Yaayy!! Your apple plant is healthy. But, maintain the soil pH of 6.0 to 7.0 for healthy growth. Avoid planting apples in a low spot where cold air or frost can settle.

Fig 4: Fruit plant Image selection page



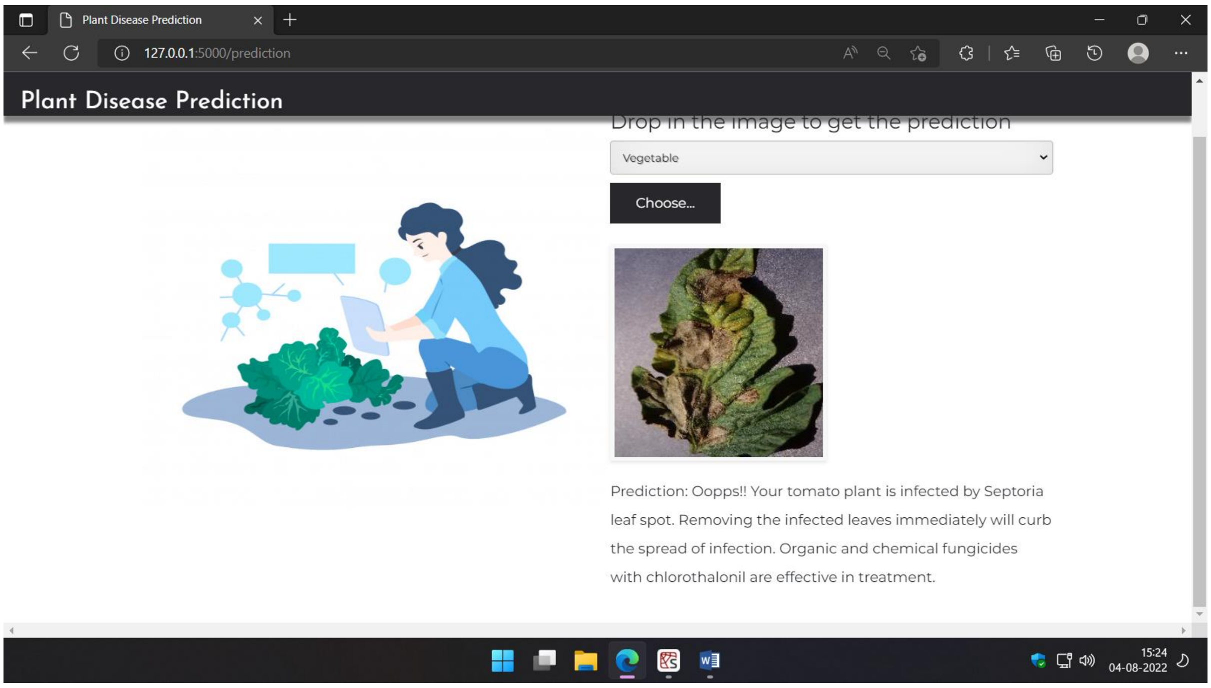
Prediction: Oopps!! Your apple plant is infected by Black Rots. This infection is a fungal infection. To control balck rot, remove the cankers by pruning at least 15 inches below the end and burn or bury them. Treating the sites with the antibiotic streptomycin or a copper-based fungicide will be helpful.

Fig 5: Prediction and Precaution



Prediction: Yaayy!! Your pepper plant is healthy. But, take the necessary precautions like, putting the plant where it gets at least 10 hours of direct sunlight. Keep soil evenly moist for good growth. Peppers need well draining soil that is rich and loamy, but avoid too much nitrogen in the soil. Too much nitrogen can cause plenty of leaves and little to no peppers.

Fig 6: Vegetable plant image Prediction and Precaution



Prediction: Ooops!! Your tomato plant is infected by Septoria leaf spot. Removing the infected leaves immediately will curb the spread of infection. Organic and chemical fungicides with chlorothalonil are effective in treatment.

Fig 7: Vegetable Image selection and Prediction

7. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

1. The proposed model could predict the disease just from the image of a particular plant
2. Easy to use UI Interface
3. Model with accurate prediction and high accuracy
4. Fertilizer and precaution suggestion based on plant disease

DISADVANTAGES:

1. Prediction is limited to few plants

8. APPLICATIONS

This web application can be used by farmers or any 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 household plants as well.

9. CONCLUSIONS

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. This project implements Deep learning techniques to identify the diseases and suggest the precautions that can be taken for those diseases. Usage of such applications could help the farmers to necessary precautions so that they don't face any loss as such.

10. FUTURE SCOPE

The application can be enhanced to predict the disease with live capture or can be used to monitor continuously in real time using live feed to detect the disease at the beginning itself and take necessary precautions.

11. BIBILOGRAPHY

1. D. S, S. M, N. P. R. N, R. W. A. I. M. N. Weerakkodi, A. Jayakody and N. Gamage, "Agro-Mate: A Virtual Assister to Maximize Crop Yield in Agriculture Sector," TENCON 2021 - 2021 IEEE Region 10 Conference (TENCON), 2021, pp. 387-392, doi: 10.1109/TENCON54134.2021.9707199.

2. Kishori Patil, Santosh Chobe, "Leaf Disease Detection using Deep Learning Algorithm",2020 International Journal of Engineering and Advanced Technology (IJEAT),2020, pp.3172-3175,doi: 10.35940/ijeat.C5965.029320.

3. C. P. Wickramasinghe, P. L. N. Lakshitha, H. P. H. S. Hemapriya, A. Jayakody and P. G. N. S. Ranasinghe, "Smart Crop and Fertilizer Prediction System," 2019 International Conference on Advancements in Computing (ICAC), 2019, pp. 487-492, doi: 10.1109/ICAC49085.2019.9103422.

4. A. KP and J. Anitha, "Plant disease classification using deep learning," 2021 3rd International Conference on Signal Processing and Communication (ICPSC), 2021, pp. 407-411, doi:

10.1109/ICSPC51351.2021.9451696.

5. A. Lakshmanarao, M. R. Babu and T. S. R. Kiran, "Plant Disease Prediction and classification using Deep Learning ConvNets," 2021 International Conference on Artificial Intelligence and Machine Vision (AIMV), 2021, pp. 1-6, doi: 10.1109/AIMV53313.2021.9670918.

6. P. Rubini and P. Kavitha, "Deep Learning model for early prediction of plant disease," 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 2021, pp. 1104-1107, doi: 10.1109/ICICV50876.2021.9388538.

7. S. Nandhini and K. Ashokkumar, "Analysis on Prediction of Plant Leaf diseases using Deep Learning," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021, pp. 165-169, doi: 10.1109/ICAIS50930.2021.9395751.

8. A. S. Tulshan and N. Raul, "Plant Leaf Disease Detection using Machine Learning," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2019, pp. 1-6, doi: 10.1109/ICCCNT45670.2019.8944556.

9. M. Asta Lakshmi and V. Gomathi, "Automatic Prediction of Plant Leaf Diseases Using Deep Learning Models: A Review," 2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT), 2021, pp. 569-574, doi: 10.1109/ICEECCOT52851.2021.9708043.

10. S. Bondre and A. K. Sharma, "Review on Leaf diseases detection using Deep learning," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), 2021, pp. 1455-1461, doi: 10.1109/ICESC51422.2021.9532697.

11. R. Neela, P. Nithya ,“Fertilizers Recommendation System For Disease Prediction In Tree Leave”,2019 INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH 2019,pp.3343-3346.

APPENDIX

Source Code:

Fruit-Training.ipynb

```
PREPROCESS THE IMAGES
Image Data Augmentation
Import ImageDataGenerator Library and Configure it
from 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)
Apply ImageDataGenerator functionality to Train and Test set
x_train = train_datagen.flow_from_directory('fruit-dataset/train',
                              target_size = (128,128),batch_size = 32,class_mode =
'categorical')
x_test = test_datagen.flow_from_directory('fruit-dataset/test',
                              target_size = (128,128),batch_size = 32,class_mode =
'categorical')
Found 5384 images belonging to 6 classes.
Found 1686 images belonging to 6 classes.
MODEL BUILDING FOR FRUIT DISEASE PREDICTION
Importing the Libraries
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
Initializing the Model
model=Sequential()
Add CNN Layers
#Activation Function
model.add(Convolution2D(32,(5,5),input_shape = (128,128,3),activation = 'relu'))
#Max Pooling
model.add(MaxPooling2D(pool_size = (2,2)))
#Add Flatten layer
model.add(Flatten())
Add Dense Layers
model.add(Dense(40,kernel_initializer='uniform', activation='relu'))
model.add(Dense(20,kernel_initializer='random_uniform', activation='relu'))
model.add(Dense(6,kernel_initializer='random_uniform', activation='softmax'))
TRAIN AND SAVE THE MODEL
Compile the model
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
Fit and save the model
model.fit(x_train,steps_per_epoch = 168, epochs = 3, validation_data = x_test,
validation_steps = 52)
Epoch 1/3
168/168 [==============================] - 201s 1s/step - loss: 0.1745 - accuracy:
0.9398 - val_loss: 151.6907 - val_accuracy: 0.7879
Epoch 2/3
168/168 [==============================] - 192s 1s/step - loss: 0.1048 - accuracy:
0.9638 - val_loss: 265.2658 - val_accuracy: 0.7825
Epoch 3/3
168/168 [==============================] - 194s 1s/step - loss: 0.1128 - accuracy:
0.9638 - val_loss: 130.0285 - val_accuracy: 0.8456
<keras.callbacks.History at 0x1df3bc28be0>
model.save('fruit.h5')
model.save('fruit.h5')
```

Vegetable-Training.ipynb

```
PREPROCESS THE IMAGES
Image Data Augmentation
Import ImageDataGenerator Library and Configure it
from 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)
Apply ImageDataGenerator functionality to Train and Test set
x_train = train_datagen.flow_from_directory('veg-dataset/train_set',
                                target_size = (128,128),batch_size = 16,class_mode =
'categorical')
x_test = test_datagen.flow_from_directory('veg-dataset/test_set',
                                target_size = (128,128),batch_size = 16,class_mode =
'categorical')
Found 11386 images belonging to 9 classes.
Found 3416 images belonging to 9 classes.
MODEL BUILDING FOR VEGETABLE DISEASE PREDICTION
Importing the Libraries
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
Initializing the Model
model=Sequential()
Add CNN Layers
#Activation Function
```

```
model.add(Convolution2D(32,(5,5),input_shape = (128,128,3),activation = 'relu'))
#Max Pooling
model.add(MaxPooling2D(pool_size = (2,2)))
#Add the flatten layer
model.add(Flatten())
Add Dense Layers
model.add(Dense(units = 300,kernel_initializer='uniform', activation='relu'))
model.add(Dense(units = 150,kernel_initializer='uniform', activation='relu'))
model.add(Dense(units = 75,kernel_initializer='uniform', activation='relu'))
model.add(Dense(9, kernel_initializer='uniform', activation='softmax'))
TRAIN AND SAVE THE MODEL
Compile the model
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
Fit and save the model
model.fit(x_train,steps_per_epoch = 150, epochs = 20, validation_data = x_test,
validation_steps = 40)
Epoch 1/20
150/150 [==============================] - 43s 282ms/step - loss: 0.5447 -
accuracy: 0.8133 - val_loss: 718.6071 - val_accuracy: 0.3703
Epoch 2/20
150/150 [==============================] - 44s 293ms/step - loss: 0.5246 -
accuracy: 0.8133 - val_loss: 447.5519 - val_accuracy: 0.4688
Epoch 3/20
150/150 [==============================] - 43s 284ms/step - loss: 0.5247 -
accuracy: 0.8138 - val_loss: 471.3345 - val_accuracy: 0.4234
Epoch 4/20
150/150 [==============================] - 42s 282ms/step - loss: 0.4690 -
accuracy: 0.8296 - val_loss: 530.0467 - val_accuracy: 0.4500
Epoch 5/20
150/150 [==============================] - 43s 285ms/step - loss: 0.4618 -
accuracy: 0.8317 - val_loss: 420.3822 - val_accuracy: 0.4000
Epoch 6/20
150/150 [==============================] - 43s 283ms/step - loss: 0.3974 -
accuracy: 0.8592 - val_loss: 551.9142 - val_accuracy: 0.4703
Epoch 7/20
150/150 [==============================] - 44s 291ms/step - loss: 0.4480 -
accuracy: 0.8462 - val_loss: 649.0431 - val_accuracy: 0.4500
Epoch 8/20
150/150 [==============================] - 43s 284ms/step - loss: 0.4053 -
accuracy: 0.8587 - val_loss: 549.8568 - val_accuracy: 0.4547
Epoch 9/20
150/150 [==============================] - 43s 288ms/step - loss: 0.4252 -
accuracy: 0.8504 - val_loss: 525.5988 - val_accuracy: 0.4531
Epoch 10/20
150/150 [==============================] - 44s 296ms/step - loss: 0.4395 -
accuracy: 0.8512 - val_loss: 523.9396 - val_accuracy: 0.4672
Epoch 11/20
150/150 [==============================] - 42s 277ms/step - loss: 0.3844 -
accuracy: 0.8662 - val_loss: 729.3231 - val_accuracy: 0.4344
Epoch 12/20
```

```
150/150 [==============================] - 41s 275ms/step - loss: 0.3824 -
accuracy: 0.8696 - val_loss: 835.8876 - val_accuracy: 0.3812
Epoch 13/20
150/150 [==============================] - 41s 272ms/step - loss: 0.3719 -
accuracy: 0.8759 - val_loss: 478.4791 - val_accuracy: 0.4406
Epoch 14/20
150/150 [==============================] - 41s 272ms/step - loss: 0.3587 -
accuracy: 0.8712 - val_loss: 686.9310 - val_accuracy: 0.3594
Epoch 15/20
150/150 [==============================] - 41s 274ms/step - loss: 0.3332 -
accuracy: 0.8850 - val_loss: 413.1308 - val_accuracy: 0.5609
Epoch 16/20
150/150 [==============================] - 41s 274ms/step - loss: 0.3558 -
accuracy: 0.8779 - val_loss: 620.4662 - val_accuracy: 0.4266
Epoch 17/20
150/150 [==============================] - 41s 273ms/step - loss: 0.3672 -
accuracy: 0.8754 - val_loss: 431.7836 - val_accuracy: 0.4922
Epoch 18/20
150/150 [==============================] - 41s 273ms/step - loss: 0.3082 -
accuracy: 0.8921 - val_loss: 793.1554 - val_accuracy: 0.3859
Epoch 19/20
150/150 [==============================] - 41s 272ms/step - loss: 0.3491 -
accuracy: 0.8763 - val_loss: 829.8880 - val_accuracy: 0.3844
Epoch 20/20
150/150 [==============================] - 42s 277ms/step - loss: 0.3043 -
accuracy: 0.9008 - val_loss: 1051.2703 - val_accuracy: 0.3406
<keras.callbacks.History at 0x2747f3c4fa0>
model.save('vegetable.h5')
```

Plant-Disease-Testing.ipynb

```
Import the packages and load the saved model
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
import keras
import tensorflow as tf
Loading Fruit Model
model = load_model('fruit.h5')
Load the test image, pre-process it and predict
img = keras.utils.load_img('apple_healthy.JPG', target_size = (128,128))
x = keras.utils.img_to_array(img)
x=np.expand_dims(x,axis = 0)
pred=model.predict(x)
pred = np.argmax(pred, axis = 1)[:5]
1/1 [==============================] - 0s 280ms/step
print(pred)
[1]
```

```
Loading Vegetable Model
model1 = load_model('vegetable.h5')
Load the test image, pre-process it and predict
img1 = keras.utils.load_img('potato_healthy.JPG', target_size = (128,128))
x = keras.utils.img_to_array(img1)
x=np.expand_dims(x,axis = 0)
pred=model.predict(x)
pred = np.argmax(pred, axis = 1)[:5]
1/1 [==============================] - 0s 21ms/step
print(pred)
[1]
```

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_)

#Loading both models veg and fruits
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 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)
```