

# **Fertilizers Recommendation System for Disease Prediction**

**PROJECT REPORT**

*Submitted by*

**Team ID: PNT2022TMID22653**

**G. ELAKIYA SRI (732919CSR024)**

**M.BRINDHA (732919CSR017)**

**R.P ARUN (732919CSR016)**

**R. JANANI SRI (732919CSR039)**

*In partial fulfilment for the award of the degree*

*Of*

**BACHELOR OF ENGINEERING**

**In**

**COMPUTER SCIENCE AND ENGINEERING**



**VELALAR COLLEGE OF ENGINEERING AND TECHNOLOGY,  
THINDAL.**

# **1.INTRODUCTION**

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

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

## **2.LITERATURE SURVEY**

2.1 Existing problem 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. Shiva reddy proposed an IoT based system for leaf disease detection and fertilizer recommendation which is based on Machine Learning techniques yields less 80 percentage accuracies.

2.2 Proposed solution In this project work, a deep learning based neural network is used to train the collected datasets and test the same. The deep learning based neural

2.3 network is CNN which gives more than 90% classification accuracies. By increasing

**3.** The more number of dense layers and by modifying hyperparameters such as number of epochs, batch size, the accuracy rate can be increased to 95% to 98%.

## 4.THEORITICAL ANALYSIS

### 4.1 Block diagram

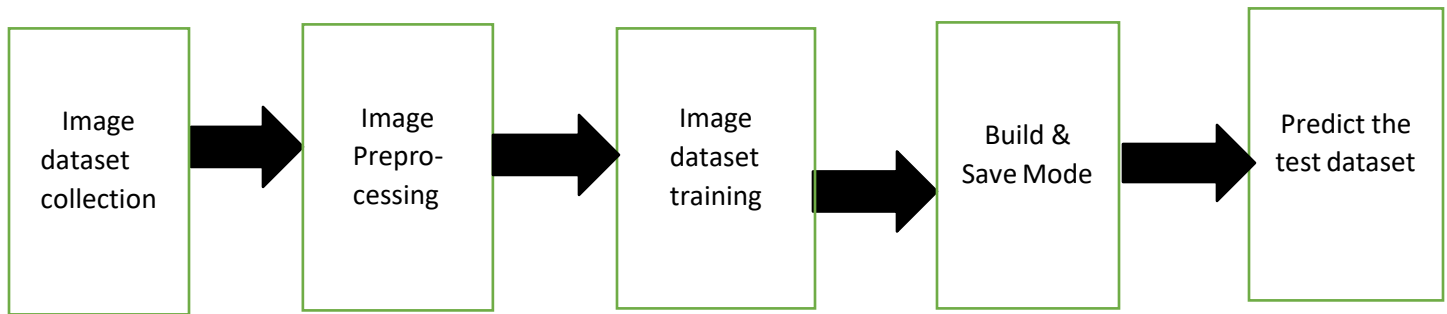


Figure.3.1. Block Diagram of the project

The block diagram of the entire project is shown in Fig.3.1. First step is the image dataset collection followed by image preprocessing. The third step is the training of image datasets with initializing different hyper parameters. Then build the model and save the model file with .h5 format. The final stage is the testing of existing or new datasets using the trained model.

### 4.2 Hardware/Software designing

The software used for training and testing the dataset is Python. The Jupyter notebook (Notebook of IBM cloud also) is used for python programming. The neural network used for training and testing the model is Convolutional Neural Network (CNN).

The CNN has following layers:

- Convolutional layer (32x32 kernel (3x3))

- Max-pool layer (kernel(2x2))
- Flatten layer
- Dense layer (different layers with different size)
- Drop out layer (optional)
- Final output dense layer(size 6x1 for fruit dataset and 9x1 for Vegetable dataset)

In the preprocessing step, images are normalized to 1 and then resized to 128x128. The images are arranged in different batch sizes. Then train set and test set are formed from the collected datasets. In order to do the above steps in Python, the following Python libraries must be imported before starting the process:

- NumPy
- TensorFlow
- Keras
- Matplotlib (optional for data visualization)

The following activation functions used in the CNN training:

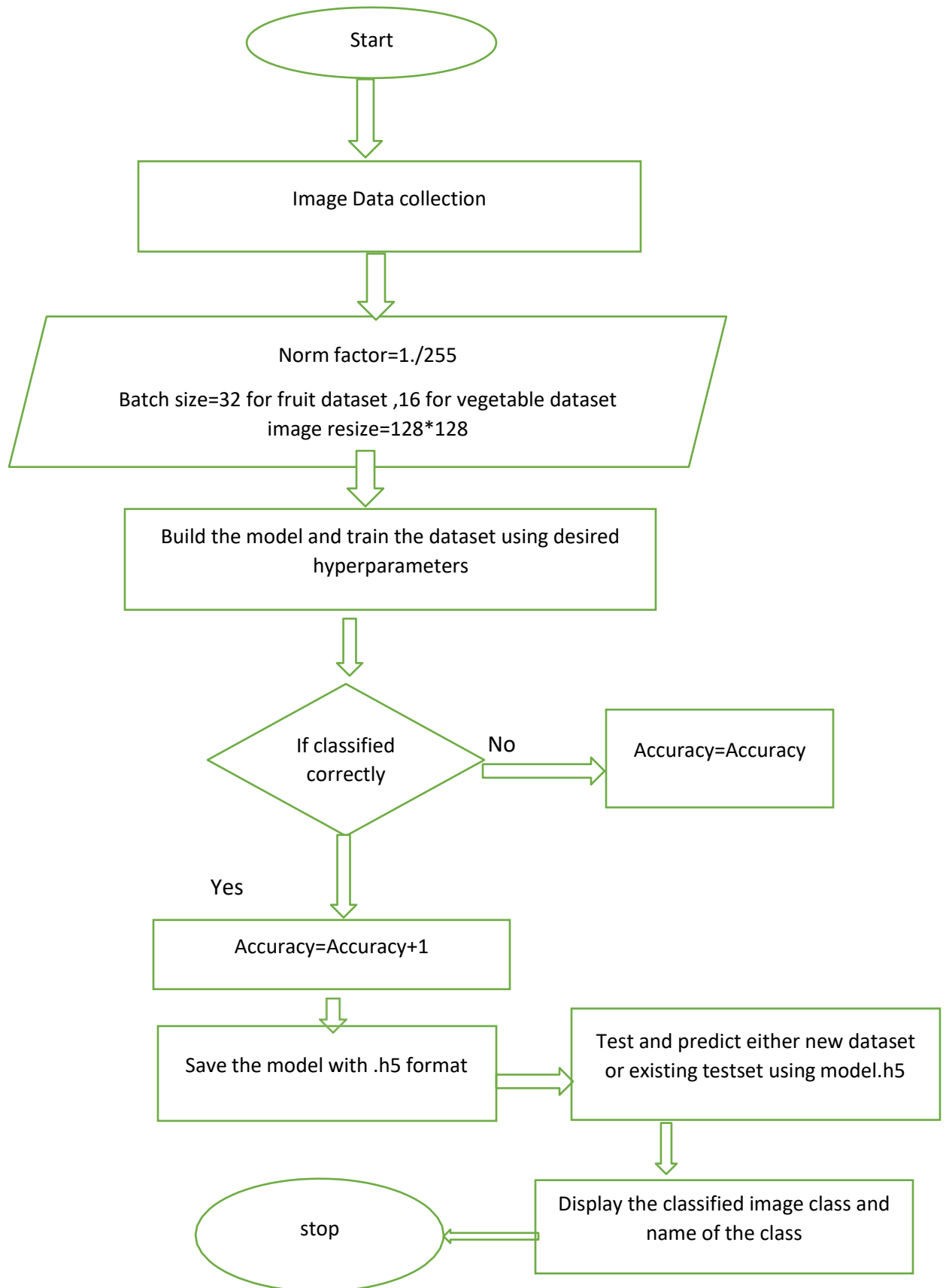
- RELU at the end of convolution layer and Max Pool layer
- SoftMax at the end of output dense layer
- For testing the dataset argmax is used, its an optional.

## **5.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.

## 6.FLOWCHARTS



## 7.RESULTS

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



```
In [40]: pred = model.predict_classes(x)

WARNING:tensorflow:From c:\python-input-40-f93bd030d5d\1: Sequential.predict_classes (from tensorflow.python.keras.engine.sequ
ential) is deprecated and will be removed after 2021-01-01.
Instructions for updating:
Please use Instead: "np.argmax(model.predict(x), axis=-1)", if your model does multi-class classification (e.g. if it uses
a 'softmax' last-layer activation), " (model.predict(x) > 0.5).astype("int32")", if your model does binary classification
(e.g. if it uses a 'sigmoid' last-layer activation).

In [41]: pred
Out[41]: array([1], dtype=int64)

In [42]: Index = ['Apple__Black_rot', 'Apple__healthy', 'Corn_(maize)__Northern_Leaf_Blight', 'Corn_(maize)__healthy', 'Peach__Bacterial_
spot', 'Peach__healthy']

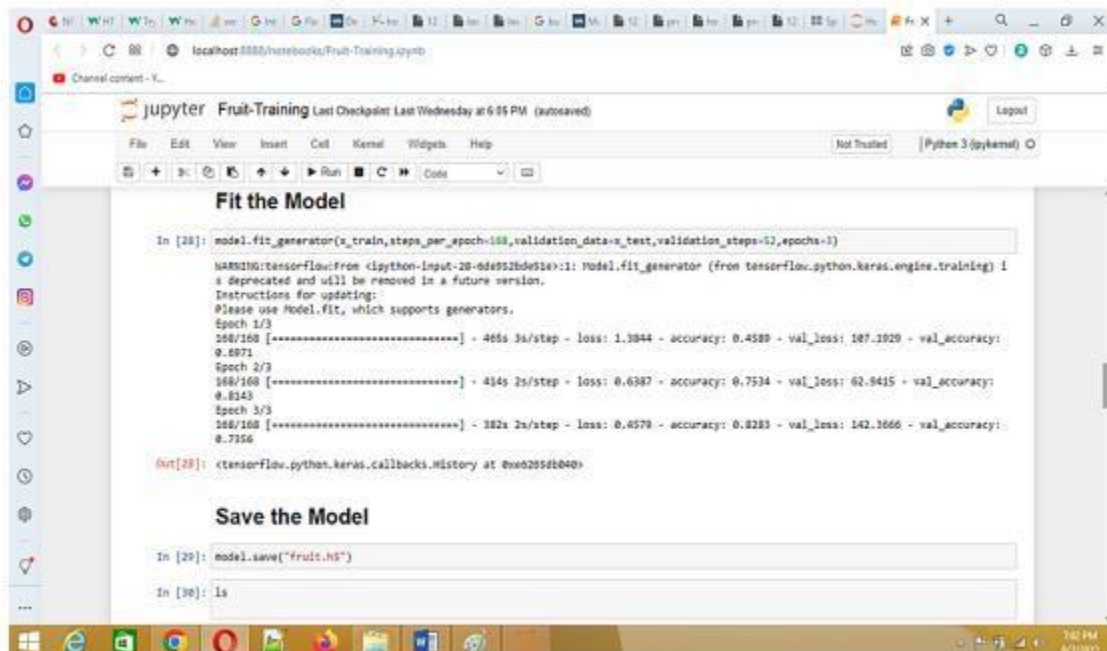
In [43]: print('the given image belongs to-', Index[pred[0]])
the given image belongs to- Apple__healthy

Test Apple Black Rot class images

In [54]: img = image.load_img('E:/IBM_COURSE/Project/Dataset Plant Disease/fruit-dataset/fruit-dataset/test/Apple__Black_rot/0f3d46f6
a.jpg')

In [55]: x_test = image_to_array(img) / 255
```

Figure.6.2 Test the Fruit dataset



```
Fit the Model

In [28]: model.fit_generator(x_train, steps_per_epoch=100, validation_data=x_test, validation_steps=10, epochs=1)

WARNING:tensorflow:From c:\python-input-28-d4d952bde51a\1: Model.fit_generator (from tensorflow.python.keras.engine.training) i
s deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.

Epoch 1/3
160/160 [=====] - 486s 3s/step - loss: 1.3844 - accuracy: 0.4589 - val_loss: 187.1929 - val_accuracy:
0.6871
Epoch 2/3
160/160 [=====] - 414s 2s/step - loss: 0.6387 - accuracy: 0.7534 - val_loss: 62.9415 - val_accuracy:
0.8543
Epoch 3/3
160/160 [=====] - 382s 2s/step - loss: 0.4579 - accuracy: 0.8283 - val_loss: 142.3666 - val_accuracy:
0.7356

Out[28]: <tensorflow.python.keras.callbacks.History at 0x0205d0e040>

Save the Model

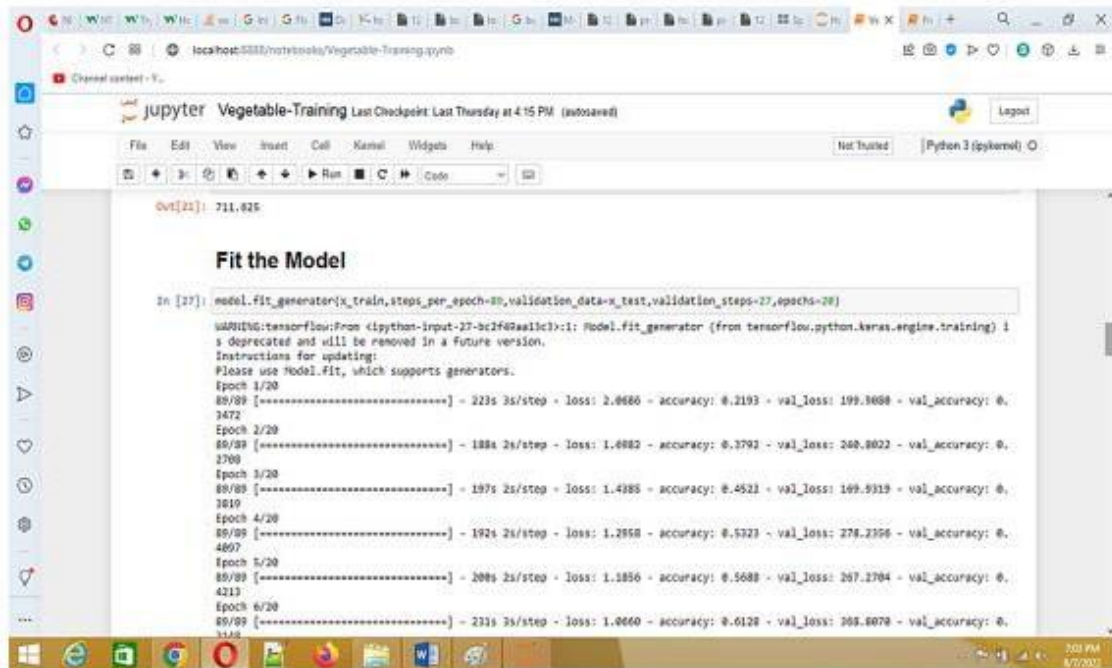
In [29]: model.save("fruit.h5")

In [30]: Is
```

Figure.6.1. Fit a model for Fruit dataset



## Train and Test Vegetable dataset



The screenshot shows a Jupyter Notebook titled "Vegetable-Training" with a Python 3 (ipykernel) environment. The notebook is running a code cell that trains a model. The output shows the model's performance over 20 epochs, with loss and accuracy metrics for both training and validation sets. The model is trained using the `fit_generator` method from Keras.

```
Out[21]: 711.825
```

### Fit the Model

```
In [27]: model.fit_generator(x_train, steps_per_epoch=80, validation_data=x_test, validation_steps=27, epochs=20)
```

WARNING:tensorflow:From <ipython-input-27-bc2f42aa1c1>:1: Model.fit\_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version. Instructions for updating: Please use Model.fit, which supports generators.

Epoch 1/20  
80/80 [=====] - 223s 3s/step - loss: 2.0660 - accuracy: 0.2193 - val\_loss: 199.8608 - val\_accuracy: 0.3472

Epoch 2/20  
80/80 [=====] - 188s 2s/step - loss: 1.6882 - accuracy: 0.3792 - val\_loss: 240.8022 - val\_accuracy: 0.2708

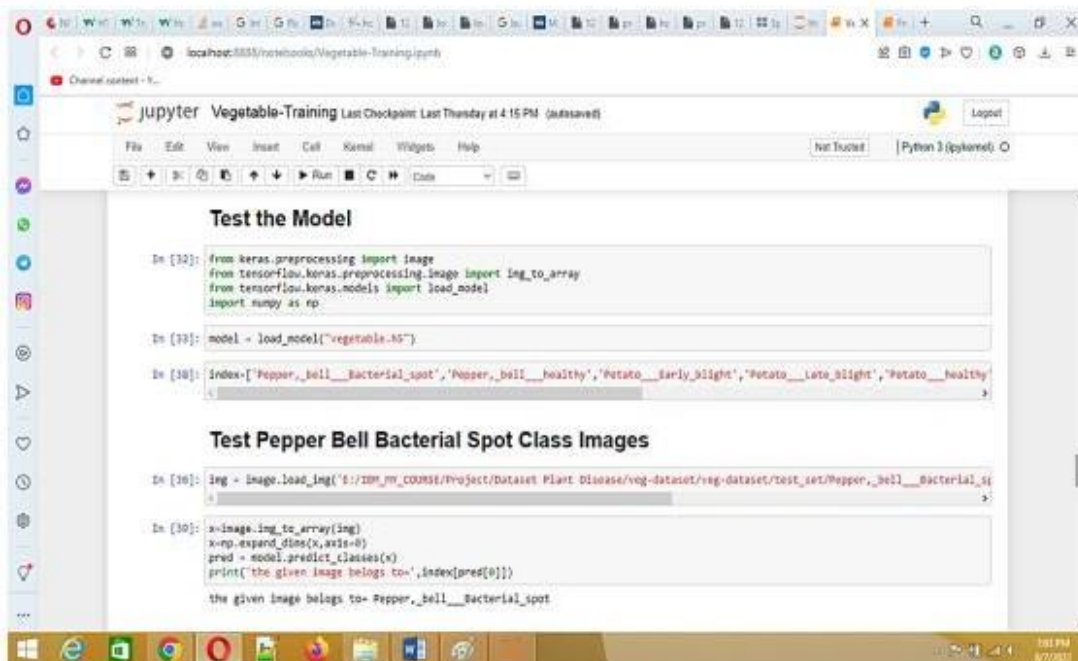
Epoch 3/20  
80/80 [=====] - 197s 2s/step - loss: 1.4085 - accuracy: 0.4521 - val\_loss: 169.9319 - val\_accuracy: 0.3010

Epoch 4/20  
80/80 [=====] - 192s 2s/step - loss: 1.2858 - accuracy: 0.5329 - val\_loss: 278.2356 - val\_accuracy: 0.4807

Epoch 5/20  
80/80 [=====] - 208s 2s/step - loss: 1.1856 - accuracy: 0.5688 - val\_loss: 267.2784 - val\_accuracy: 0.4213

Epoch 6/20  
80/80 [=====] - 231s 3s/step - loss: 1.0660 - accuracy: 0.6128 - val\_loss: 265.6070 - val\_accuracy: 0.3318

Figure.6.3. Train the Vegetable dataset



The screenshot shows a Jupyter Notebook titled "Vegetable-Training" with a Python 3 (ipykernel) environment. The notebook is running a code cell that tests the model. The output shows the model's performance on a test set, with loss and accuracy metrics for both training and validation sets. The model is tested using the `load_model` method from Keras.

### Test the Model

```
In [32]: from keras.preprocessing import image  
from tensorflow.keras.preprocessing.image import img_to_array  
from tensorflow.keras.models import load_model  
import numpy as np
```

```
In [33]: model = load_model("vegetable.h5")
```

```
In [38]: index=['Pepper_bell__Bacterial_spot', 'Pepper_bell__healthy', 'Potato__Early_blight', 'Potato__Late_blight', 'Potato__healthy']
```

### Test Pepper Bell Bacterial Spot Class Images

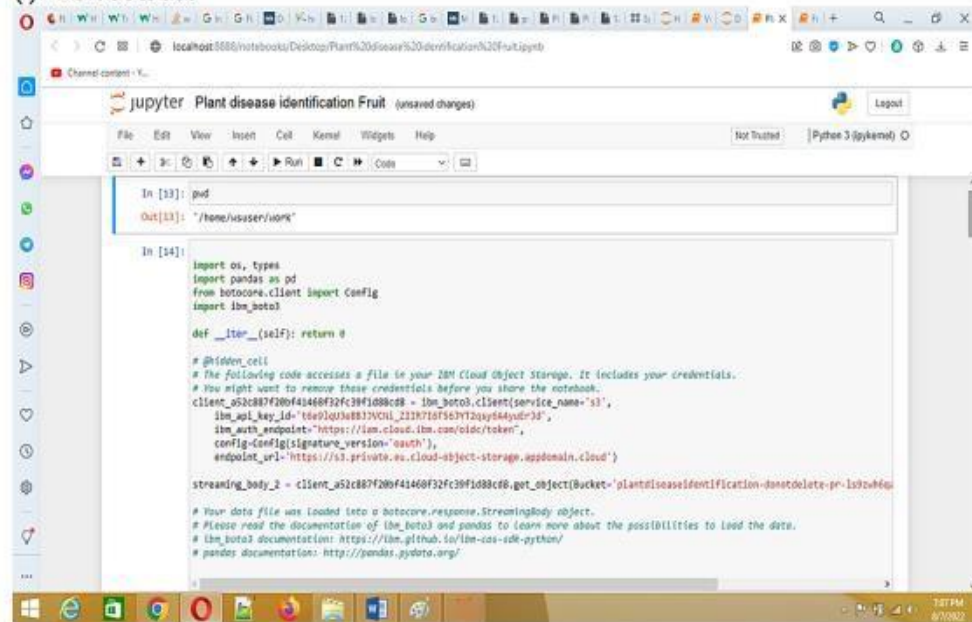
```
In [36]: img = image.load_img('F:/DTH_MY_COURSE/Project/Dataset Plant Disease/veg-dataset/veg-dataset/test_set/Pepper_bell__Bacterial_spot')  
x = image.img_to_array(img)  
x = np.expand_dims(x, axis=0)  
pred = model.predict_classes(x)  
print('the given image belongs to-', index[pred[0]])  
the given image belongs to- Pepper_bell__Bacterial_spot
```

Figure.6.4. Test the Vegetable dataset

## Train and Test Vegetable dataset IBM Cloud

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

### (i). Fruit dataset



```
In [13]: pd
Out[13]: "/home/user/work"

In [14]:
import os, types
import pandas as pd
from botocore.client import Config
import ibm_botocore

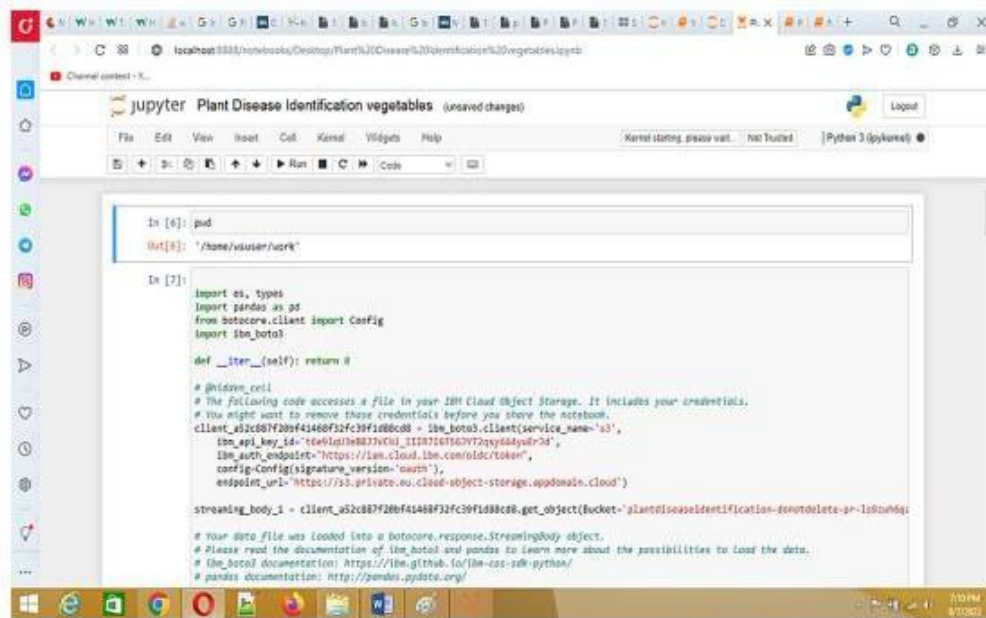
def __iter__(self): return 0

# @iiden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove these credentials before you share the notebook.
client = boto3.client('s3',
    aws_access_key_id='tde9303e8832vchi_233k7167563712ay64yudr3d',
    aws_secret_access_key='tde9303e8832vchi_233k7167563712ay64yudr3d',
    config=Config(signature_version='s3v4'),
    endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')

streaming_body_2 = client.get_object(Bucket='plantdiseaseidentification-donotdelete-pr-1s9zwh6u',
    Key='fruit_dataset.csv')

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_botocore and pandas to learn more about the possibilities to load the data.
# ibm_botocore documentation: https://ibm.github.io/ibm-cas-ide-python/
# pandas documentation: http://pandas.pydata.org/
```

Figure.6.5. Training Fruit Dataset in IBM Cloud



```
In [6]: pwd
Out[6]: '/home/vsuser/work'

In [7]: import os, types
import pandas as pd
from botocore.client import Config
import boto3

def __iter__(self): return 0

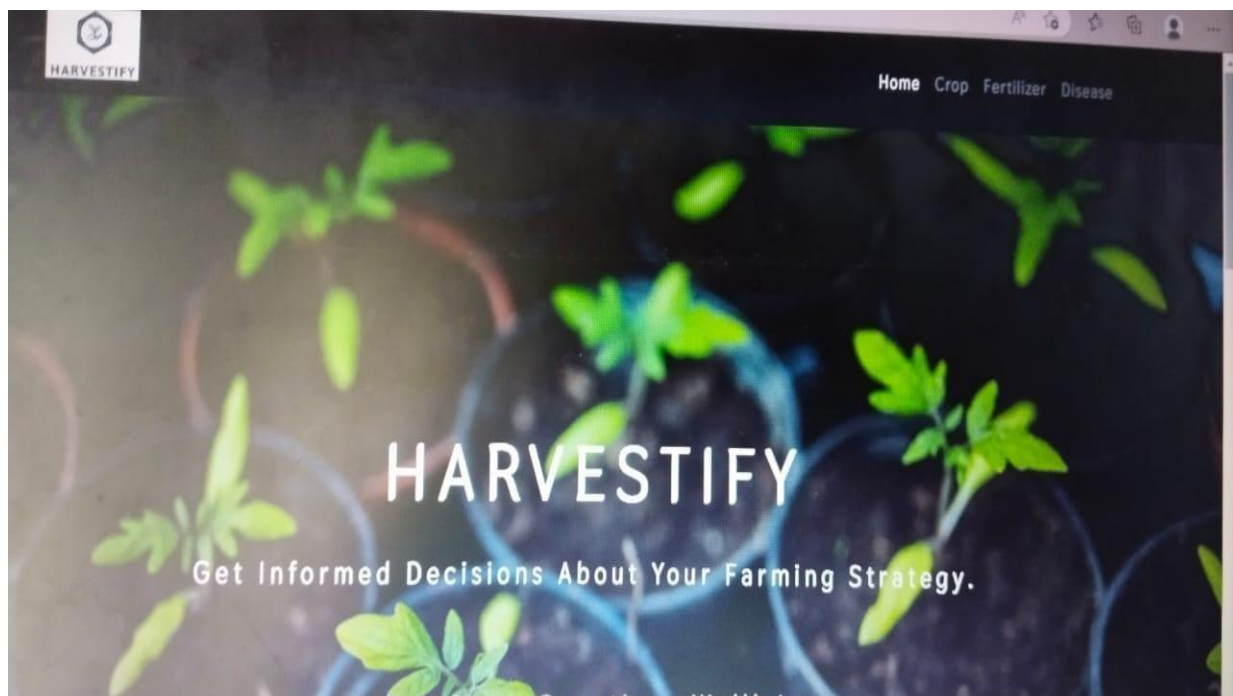
# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove these credentials before you share the notebook.
client = boto3.client(service_name='s3',
    aws_access_key_id='IAW45JH837XCUJ_I7R1D5T56V72eydldy6r0d',
    aws_secret_access_key='https://iam.cloud.ibm.com/oidc/token',
    config=Config(signature_version='auth'),
    endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')

streaming_body_1 = client.get_object(Bucket='plantdiseaseidentification-donotdelete-pr-158uh6q',
    Key='data/train.csv')

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of boto3 and pandas to learn more about the possibilities to load the data.
# boto3 documentation: https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html#python
# pandas documentation: http://pandas.pydata.org/
```

Figure.6.6. Training Vegetable Dataset in IBM Cloud

Out put



← ↻ <https://harvestify.herokuapp.com/crop-recommend>

Enter the value (example:50)

Phosphorous

Enter the value (example:50)

Pottasium

Enter the value (example:50)

ph level

Enter the value

Rainfall (in mm)

Enter the value


State

Select State ▾

City

▾

Predict

 [Home](#) [Crop](#) [Fertilizer](#) [Disease](#)

## Get informed advice on fertilizer based on soil

Nitrogen

Enter the value (example:50)

Phosphorous

Enter the value (example:50)

Pottasium

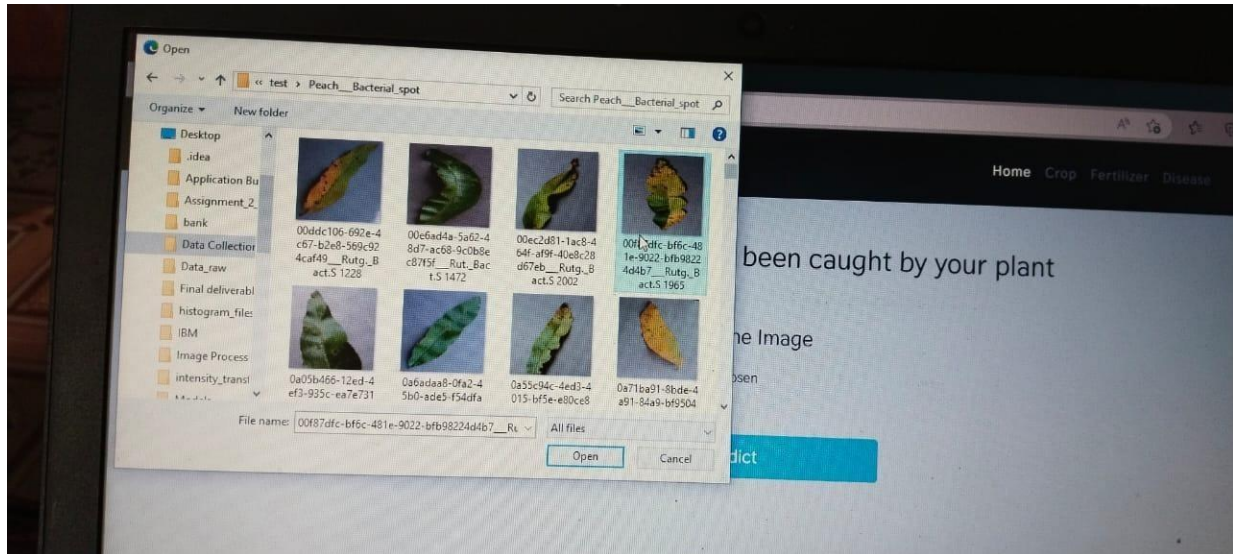
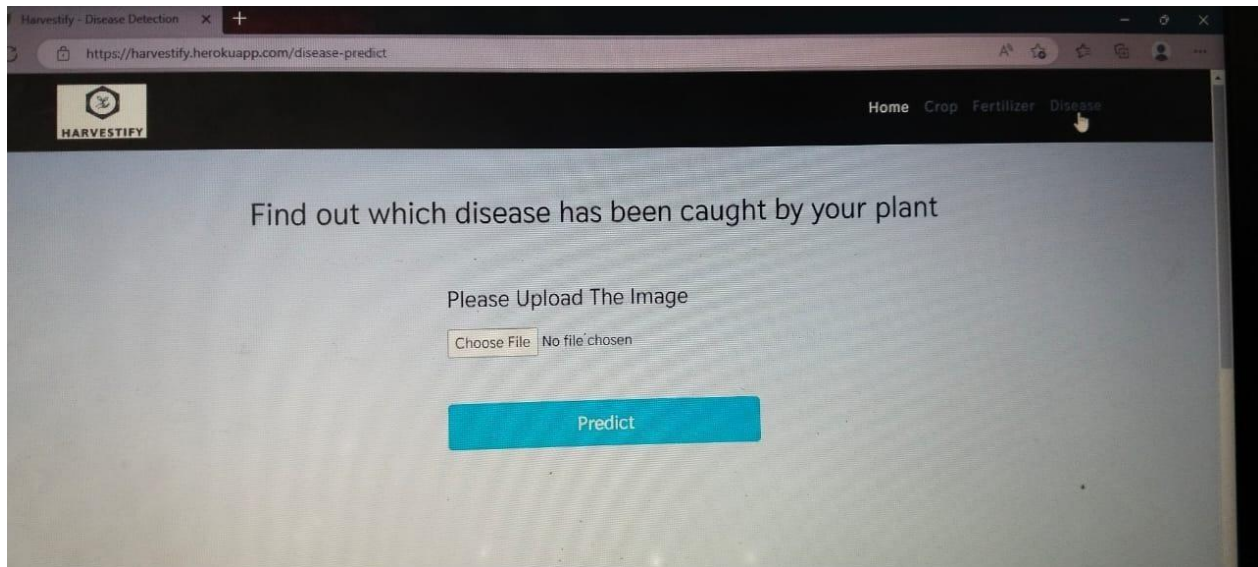
Enter the value (example:50)

Crop you want to grow

Select crop ▾

Predict





Find out which disease has been caught by your plant

Please Upload The Image

Choose File 00f87dfc-bf6c-...Bact.S 1965.JPG



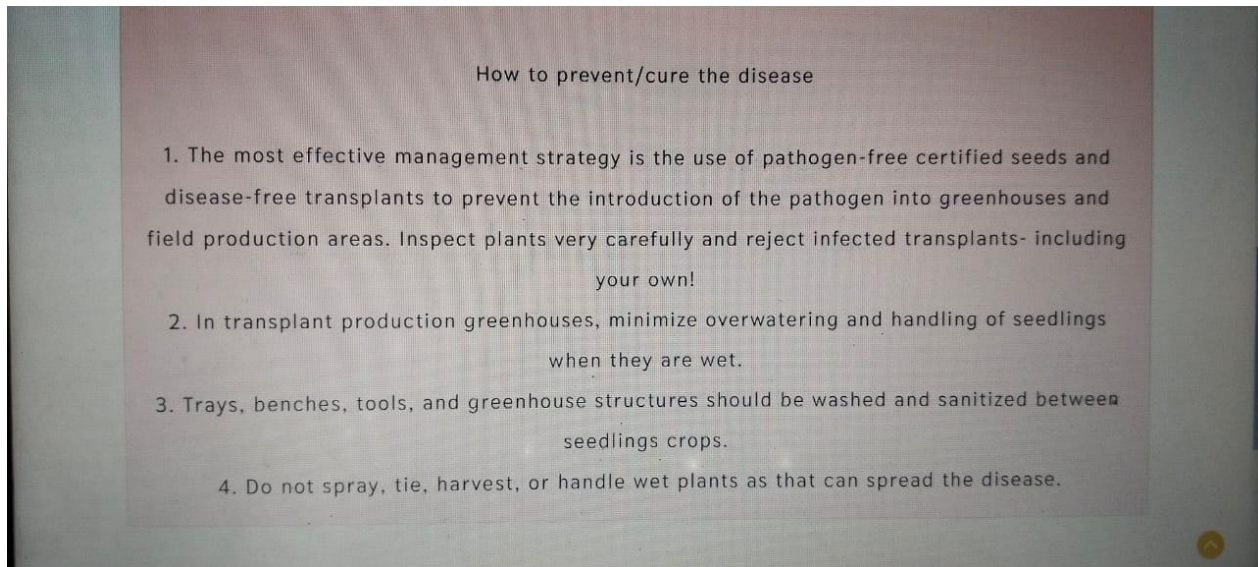
Predict

Crop: Peach

Disease: Bacterial Spot

Cause of disease:

1. The disease is caused by four species of *Xanthomonas* (*X. euvesicatoria*, *X. gardneri*, *X. perforans*, and *X. vesicatoria*). In North Carolina, *X. perforans* is the predominant species associated with bacterial spot on tomato and *X. euvesicatoria* is the predominant species associated with the disease on pepper.
2. All four bacteria are strictly aerobic, gram-negative rods with a long whip-like flagellum (tail) that allows them to move in water, which allows them to invade wet plant tissue and cause infection.



## **8.ADVANTAGES & DISADVANTAGES**

### **List of advantages**

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

### **List of disadvantages**

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

## **9.APPLICATIONS**

1. The trained network model used to classify the image patterns with high accuracy.
2. The proposed model not only used for plant disease classification but also for other image pattern classification such as animal classification.

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

## **9.CONCLUSIONS**

The model proposed here involves image classification of fruit datasets and vegetable datasets. The following points are observed during model testing and training:

- The accuracy of classification increased by increasing the number of epochs.
- For different batch sizes, different classification accuracies are obtained.
- The accuracies are increased by increasing more convolution layers.
- The accuracy of classification also increased by varying dense layers.
- Different accuracies are obtained by varying the size of kernel used in the convolution layer output.
- Accuracies are different while varying the size of the train and test datasets.

## **10.FUTURE SCOPE**

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



## 11.BIBILOGRAPHY

[1]. R Indumathi Leaf Disease Detection and Fertilizer Suggestion", IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), 29-30 March 2019, DOI: 10.1109/ICSCAN.2019.8878781.

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

[3]. H Shiva reddy, Ganesh hedge, Prof. DR Chinnaya3, "IoT based Leaf Disease Detection and Fertilizer Recommendation", International Research Journal of Engineering and Technology (IRJET), Volume: 06 Issue: 11, Nov 2019, e-ISSN: 2395- 0056.

[4].Dimitrovski, Ivica, GjorgjiMadjarov, DragiKoccev, and PetreLameski, "Maestra at LifeCLEF 2014 Plant Task: Plant Identification using Visual Data", In CLEF (Working Notes), pp. 705-714, 2014.

[5].

## APPENDIX

A. Source Code (Jupyter notebook python code)

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

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# In[1]: pwd
```

```
#In[2]:cdE:/IBM/Project/Dataset/PlantDisease/fruit-dataset/fruit-dataset
```

```
# # Apply ImageDataGenerator functionality to Train and Test set
```

```
# # Preprocessing #In[3]: from keras.preprocessing.imageimport
```

## Image DataGenerator

```
train_datagen=
```

```
ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True) test_datagen = ImageDataGenerator(rescale=1) # In[4]: pwd
```

```
#In[5]: x_train =
```

```
train_datagen.flow_from_directory('E:/IBM/Project/Dataset Plant
```

```
Disease/fruit-
```

```
dataset/fruitdataset/train', target_size=(128,128), batch_size=32, class_mode='categorical')
```

```
#In[6]:
```

```
x_test=test_datagen.flow_from_directory('E:/IBM/Project/Dataset Plant
```

```
Disease/fruit-dataset/fruit-dataset/test', target_size=(128,128),
```

```
batch_size=32, class_mode='categorical') ## Import the models
```

```
#In[7]: from tensorflow.keras.models import Sequential from
```

```
tensorflow.keras.layers import Dense, Convolution2D, MaxPool2D, Flatten
```

```
## Initializing the models 10
```

```
# In[8]: model=Sequential()
```

```
## Add CNN Layers
```

```
#In[9]:
```

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

```
# In[10]: x_train.class_indices
```

```
## Add Pooling layer
```

```
# In[11]: model.add(MaxPool2D(pool_size=(2,2)))
```

```
## Add Flatten layer # In[12]: model.add(Flatten())
```

```
## Add Dense Layer
```

```

#In[21]: model.add(Dense(40, kernel_initializer='uniform',activation='relu'))
model.add(Dense(20, kernel_initializer='random_uniform',activation='relu'))

# # Add Output Layer # In[24]: model.add(Dense(6,activation='softmax',
kernel_initializer='random_uniform'))

## Compile the model#

In[25]:
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accur
acy' ])

# In[26]: len(x_train)

# In[27]: 5384/32

# # Fit the Model

#In[28]:
model.fit_generator(x_train,steps_per_epoch=168,validation_data=x_test,validat
ion_steps=52,epochs=3)

# # Save the Model

# In[29]: model.save("fruit.h5")

# In[30]: ls

# # Test the Model

#In[32]:from keras.preprocessing import image from
tensorflow.keras.preprocessing.image import img_to_array from
tensorflow.keras.models import load_model import numpy as np

# In[33]: model = load_model("fruit.h5")

# # Test Apple_Healthy Class images

```

```

# In[37]: img = image.load_img('E:/IBM/Project/Dataset Plant
Disease/fruitdataset/fruit-dataset/test/Apple___healthy/00fca0da-2db3-481b-
b98a9b67bb7b105c___RS_HL_7708.JPG',target_size=(128,128)) 11

# In[39]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0)

# In[40]: pred = model.predict_classes(x)

# In[41]: pred

#In[45]:index
=['Apple___Black_rot','Apple___healthy','Corn_(maize)___Northern_Leaf_Blig
ht','Corn_(maize)___healthy','Peach___Bacterial_spot','Peach___healthy']

# In[46]: print('the given image belongs to=',index[pred[0]])

# # Test Apple Black Rot class images # In[54]: img =
image.load_img('E:/IBM/Project/Dataset Plant
Disease/fruitdataset/fruit-dataset/test/Apple___Black_rot/0f3d45f4-e121-42cd-
a5b6-be2f866a0574___JR_FrgE.S_2870.JPG',target_size=(128,128))

# In[55]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred =
model.predict_classes(x) print('the given image belongs to=',index[pred[0]])

# # Test Corn Northern leaf Blight class images

# In[56]: img = image.load_img('E:/IBM/Project/Dataset Plant
Disease/fruitdataset/fruit-
dataset/test/Corn_(maize)___Northern_Leaf_Blight/00a14441-7a62-4034-bc40-
b196aeab2785___RS_NLB_3932.JPG',target_size=(128,128))

#In[57]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred =
model.predict_classes(x) print('the given image belongs to=',index[pred[0]])

```

```

# # Test Corn Healthy class images

#In[58]: img =image.load_img('E:/IBM/Project/Dataset Plant
Disease/fruitdataset/fruit-dataset/test/Corn_(maize)___healthy/0a68ef5a-027c-
41ae-b227- 159dae77d3dd___R.S_HL 7969 copy.jpg',target_size=(128,128))

# In[59]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred =
model.predict_classes(x) print('the given image belongs to=',index[pred[0]]) # #
Test Peach Bacterial spot class images # In[60]: img =
image.load_img('E:/IBM/Project/Dataset Plant
Disease/fruitdataset/fruit-dataset/test/Peach___Bacterial_spot/00ddc106-692e-
4c67-b2e8- 569c924caf49___Rutg._Bact.S 1228.JPG',target_size=(128,128)) 12
# In[61]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred =
model.predict_classes(x) print('the given image belongs to=',index[pred[0]])

# # Test Peach Healthy class images

# In[62]: img = image.load_img('E:/IBM/Project/Dataset Plant
Disease/fruitdataset/fruit-dataset/test/Peach___healthy/1a07ce54-f4fd-41cf-
b088- 144f6bf71859___Rutg._HL 3543.JPG',target_size=(128,128))

#In[63]:x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred =
model.predict_classes(x) print('the given image belongs to=',index[pred[0]])

```

